ABSTRACT

Title of Document: EFFECTS OF ASSESSMENT FREQUENCY AND DATA-MANAGEMENT LAG ON FISHERY MANAGMEMENT PERFORMANCE: STRATEGIES FOR IMPROVEMENT

Andrea Lynn Sylvia, Master of Science, 2015

Directed By: Associate Professor, Dr. Michael J. Wilberg, Marine Estuarine Environmental Sciences

Some challenges of using stock assessments in management decisions are data availability and the allocation of resources to conduct stock assessments in a frequent and timely manner. We conducted a simulation evaluation that included the population dynamics, stock assessment, and management. Our objectives were to 1) determine effects of assessment interval and data-management lag and 2) test methods to reduce data lag by using partial data in the last year of the assessment. We found that increasing assessment interval and data-management lag caused a decrease in average catch and biomass across scenarios, with data-management lag having a larger effect compared to assessment interval. To reduce the effects of data-management lag, lag reduction methods that included some information about the age-composition of the catch and survey performed about as well as not having lag. Stock assessment interval, data-management lag and lag reduction methods should be considered when designing fishery management plans.

EFFECTS OF ASSESSMENT FREQUENCY AND DATA-MANAGEMENT LAG ON FISHERY MANAGMEMENT PERFORMANCE: STRATEGIES FOR IMPROVEMENT

By

Andrea Lynn Sylvia

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2015

Advisory Committee:
Associate Professor, Dr. Michael J. Wilberg, Chair
Dr. Thomas Miller
Dr. Lora Harris

Acknowledgements

First and foremost I would like to thank NOAA and the Mid-Atlantic fishery management council for funding this research. I would also like to thank Rick Methot and Rich Seagraves for helping this project come to fruition. The University of Maryland MEES program and especially the Chesapeake Biological Laboratory's professors, staff, and students have been paramount in my time here with their teaching, training and truly instilling a sense of community in me. The Wilberg Lab students: Emily Liljestrand, Cara Simpson, Jennifer Humphrey and Sarah Rains, it has been wonderful working with all you. Brian Gallagher and Suzan Shahrestani, thank you for being such great friends throughout this entire process and for never allowing me to doubt my abilities. To my committee members, Dr. Tom Miller and Dr. Lora Harris, your guidance and support has allowed me to excel as a student. I would also like to thank Dr. John Wiedenmann for working side by side with me on this project and saving me many times when I was struggling with code. For my advisor, Dr. Mike Wilberg, I feel truly fortunate to have had the opportunity to work with you these last two years. You have taught me what it means to be a scientist and I can only hope that one day I can become half the teacher and mentor that you are. Thank you again for all of your patience and guidance. I would also like to thank Bill Stewart and Dr. Eli Fenichel, you gave a young undergrad that wasn't sure what she wanted to do with her future a chance to explore opportunities and instill a passion for fisheries science and research. I would not be where I am at today if it wasn't for both of you. Most importantly, I would like to thank my family. I feel so incredibly lucky to have a family that has supported me in my life through thick and thin and has

always pushed me to pursue my dreams, even if that meant moving 3000 miles away. John-Michael, you are the greatest gift life has given me, to have a husband who has supported me, followed me around the country, and has never told me to get a "real job". I love you and our puppies (Rex, Duke, Bryan and Squirrel) with all my heart and I could not have done this without you.

Table of Contents

Acknowledgements ii
Table of Contents iv
List of Tables 1
List of Figures 2
Chapter 1: Introduction 4
..... 10
Chapter 2: Effects of assessment interval and data-management lag on Mid-Atlantic fishery management performance 12
Abstract 12
Introduction 13
Methods 17
Operating model 18
Assessment model 21
Scenarios 22
Performance metrics 24
Results 25
Discussion 28
Tables and Figures 34
Chapter 3: Performance of lag reduction methods for the Mid-Atlantic fishery management 48
Abstract 48
Introduction 49
Methods 53
Operating model 54
Lag reduction methods 57
Assessment model 59
Scenarios 61
Performance metrics 62
Results 63
Discussion 65
Tables and Figures 72
Appendices 86
Bibliography 283

List of Tables

Table 2.1. Equations governing the population and data-generating dynamics in the operating model……..…34-36

Table 2.2. Symbols, specified life history and time-varying parameters of model……..…37-38

Table 2.3. Median values for recruitment variability scenarios across performance metrics. SA refers to stock assessment intervals and DLM refers to data-management lag combinations……39-41

Table 3.1. Design and details of lag reduction methods. Lag reduction method 1 is a control with annual data lag. Lag reduction method 2 is a control with two year data lag……72

Table 3.2. Equations governing the population and data-generating dynamics in the operating model……..73-75

Table 3.3. Symbols, specified life history and time-varying parameters of model……..76-77

Table 3.4. 95 % confidence intervals for each life history and data scenarios, combinations are paired by stock assessment interval and lag reduction methods. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. Confidence intervals were calculated around the median with a Bonferonni correction to account for multiple comparisons……78-79

List of Figures

Figure 1.1. Timeline of data-management lag and assessment interval processes……10

Figure 1.2. Diagram of the management strategy evaluation (MSE) process……11

Figure 2.1. Flow diagram of management strategy evaluation model with both operating and estimation models……42

Figure 2.2. Mid-Atlantic P* approach showing decreasing probability of overfishing with a declining $B/B_{35\%}$ ratio……43

Figure 2.3. Box plots of the catch for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles……..44

Figure 2.4. Box plots of the biomass for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles……45

Figure 2.5. Box plots of the probability for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles……46

Figure 2.6. Box plots of the catch average annual variation (aav) for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles……..47

Figure 3.1. Flow diagram of management strategy evaluation model with both operating and estimation models……...80

Figure 3.2. Mid-Atlantic P* approach showing decreasing probability of overfishing with a declining $B/B_{35\%}$ ratio……81

Figure 3.3. Box plots of the catch for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles……82

Figure 3.4. Box plots of the biomass for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles……83

Figure 3.5. Box plots of the probability of overfishing for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LR1 is lag reduction method 1, LR2 is lag reduction method 2 and LR3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles……84

Figure 3.6. Box plots of the AAV of catch for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LR1 is lag reduction method 1, LR2 is lag reduction method 2 and LR3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles……85

Chapter 1: Introduction

Stock assessments are vital to modern fisheries management by providing valuable science to inform management decisions (NOAA 2012). The main goals of stock assessments are to estimate biomass, stock status, and reference points for management. Stock assessments have become more complex, with better quantification of uncertainty and improved modeling approaches (Caddy and Cochrane 2001). In fact, many regions throughout the world now require stock assessments in order to implement management regulations. For example, all U.S. federally managed fisheries are required to have catch limits which are presumably from some sort of assessment (NMFS 2006). Likewise, the International Council for Exploration of the Sea (ICES), which consists of 20 member countries, currently uses quotas informed by stock assessments to manage many stocks throughout European waters and in the North Atlantic (ICES 2012).

Although they provide estimates of parameters that are critical to wise and sustainable management of natural resources, assessments are not without their costs. A principal cost of adopting a stock assessment as a component of the management process is the additional time it takes to collect and prepare data used in stock assessments. The additional time required for an assessment is only one part of a series of time lags in the modern fisheries management process. Often management also involves extensive development and implementation of management actions that can take up to a year or more depending on the region and species. Procedures for implementing management usually consist of a peer review process, regulation development, and public comment periods; the process of implementing new

management measures can extend longer than a year if management is then delayed due to scientific uncertainty or negative short-term economic effects of proposed management actions (Brown et al. 2012). This time between data collection and implementation of management measures, or data-management lag (DML), has the potential to impact both the quality of assessment estimates and the performance of management.

In addition to DML, how often stock assessments are conducted, (stock assessment intervals) can affect management performance. The frequency with which a stock is assessed is often not determined by considerations of the "ideal" assessment interval for a particular species. Rather, in many regions of the world a single group may be tasked with assessing numerous stocks. This can result in an effort deficit due to personnel limitations. Under this situation, the assessment interval is established in part by the number of competing demands on the time of individual assessment scientists. Priorities to establishing assessment intervals can be based on economics, stock status, and ecosystem considerations. For example, in the U.S. new efforts to prioritize stock assessments are based on economic importance, biological significance, and status of the stock (NOAA 2014). While extended stock assessment intervals do not add to DML, they can cause similar problems to extended DML periods (Figure 1.1). In some cases catch limits or regulations that were set from the original stock assessment will remain constant for the inter-assessment periods. Alternatively, projections from the last stock assessment will be used to set catch limits until the next stock assessment is completed. During the period between assessments, the population may be changing over time in unexpected ways, and the

imposed fishing pressure based on previous population estimates may differ substantially from the target fishing mortality. These effects can pose problems for management, such as continuing to fish an already diminished stock and increasing the probability of stock collapse.

Few management bodies have developed strategies or adopted practices to determine appropriate stock assessment intervals or to reduce the effects of DMLs. Both stock assessment intervals and DML can vary widely depending on the region and management agency. Although some high priority stocks have annual assessments, others can experience a decade or more between assessments. Additionally, DML can extend up to ten years in some regions of the world. Studies that have explored the effects of stock assessment interval and DML have demonstrated decreased management performance, measured in a variety of ways, with increased assessment interval and DML (Mace et al. 2001, Shertzer and Prager 2007, Brown et al. 2012, ICES 2012, and Li et al., in review),. This decreased management performance included decreases in biomass and catch and increases in probability of overfishing or fishery collapse.

Hiring more scientists to mitigate DML and lengthy stock assessment interval effects is often not feasible in the short term. One approach might be to accept the time lags involved and use more biologically conservative targets to account for the effects of DML and assessment interval. But measures that are thought to be overly conservative are often rejected by stakeholders (Terceiro 2002, 2011). Decreasing the length of the management process may result in improved fishery performance (Brown et al. 2012). Additionally, reducing DML by better using the available data,

instead of waiting for more, is a potential solution that requires more investigation (Shertzer and Prager 2007).

Determining the expected performance of a management system can be challenging because it is often difficult or impossible to conduct controlled experiments at the level of the management system. Therefore, computer simulation-based approaches, i.e., management strategy evaluations (MSE), have been developed to better understand long-term management effects on the population and how those population changes ultimately affect future management outcomes (Butterworth 2007). MSEs are a widely used tool which involves evaluating the results of a variety of management options and determining trade-off decisions through the use of performance measures, which quantify how well management objectives are achieved (Smith et al., 1999). These evaluations combine several simulation models: a population model, which establishes the populations, an estimation model, which runs the stock assessments, and a management model, which simulates the management process (Figure 1.2). Using an MSE, we are able to compare assessment performance of alternative management options and their feedback controls.

Better understanding of the effects of assessment interval and DML is essential to improving management performance. The objectives of my thesis research were to 1) determine the effects of stock assessment interval and DML on fishery performance using a case study of fisheries managed by the Mid-Atlantic Fishery Management Council (MAFMC), and 2) test three methods to reduce data lag by using partial data in the last year of the assessment.

In Chapter 2, we used an MSE, which included the population dynamics and management processes, to address the first objective. The management models varied by length of assessment intervals, ranging from annual to decadal, and DMLs, ranging from a one to three year lag. We examined a range of performance measures to represent objectives of fishery management. Additionally, we evaluated how life-history, data quality and stock-recruitment dynamics interacted with DML and assessment interval to affect management performance including average catch, average biomass, probability of overfishing, and the average annual change in catch.

In Chapter 3, we used an MSE to evaluate the performance of three DML reduction methods. We modeled three different approaches which used partial data in the last year of the stock assessment in order to evaluate the effects of using some data in the terminal year of the stock assessment as opposed to using no data which would be equivalent to an additional year of data lag. The three methods were 1) age-composition data for the terminal year of the survey, but no age-composition for the fishery catch, 2) full survey age-composition with reduced quality age-composition data for the catch in the terminal year of the stock assessment (to represent using prior years' age and length data to age the catch), and 3) reduced data for the age-compositions of the survey and catch in the terminal year of the assessment (to represent using prior years' data to age the catch and indices). In using prior year's data to infer ages for the catch and indices, we would expect better estimates of biomass by reducing the DML and better performance of the stock assessment. Methods were tested against controls with one year of data and two years of data lag. We evaluated effectiveness using a range of performance measures similar to Chapter

2, as well as an additional metric which tested the proportion of fishery closures. We also tested the approaches under a range of data quality, life history, and stock-recruitment scenarios.

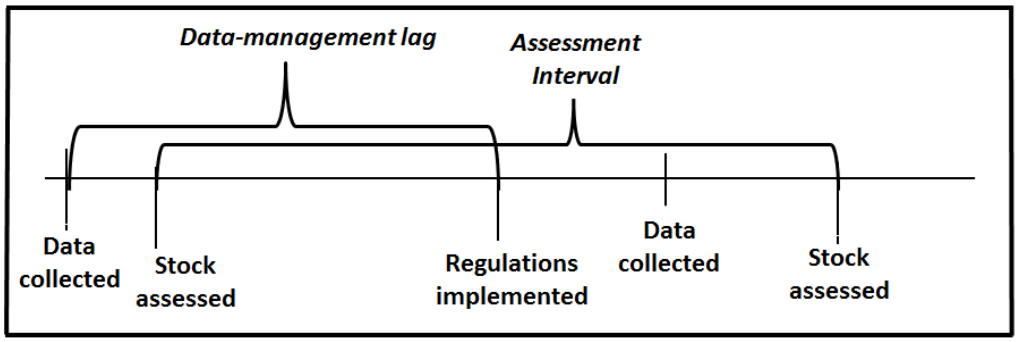

Figure 1.1. Timeline of data-management lag and assessment interval processes.

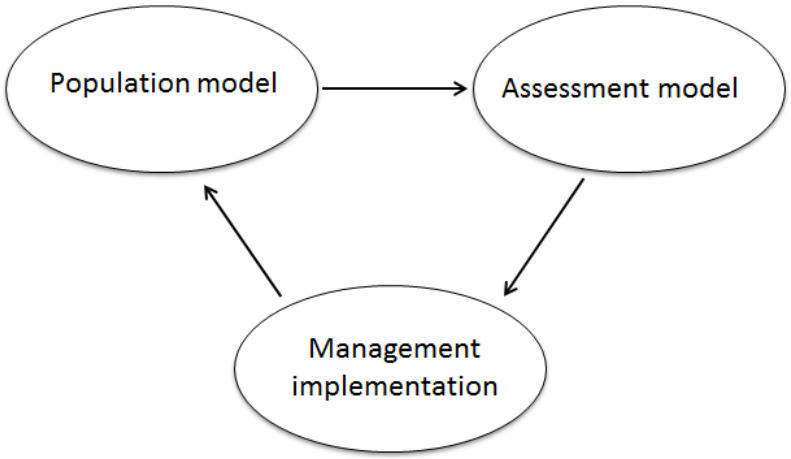

Figure 1.2. Diagram of the management strategy evaluation (MSE) process.

Chapter 2: Effects of assessment interval and data-management lag on Mid-Atlantic fishery management performance

Abstract

The use of stock assessments to inform management and decision making has increased worldwide. Two of the challenges of using stock assessments in management decisions are data availability and the allocation of resources to conduct stock assessments in a frequent and timely manner. While data-management lag and assessment timelines can strongly affect model predictions, few studies have quantified their effects. We conducted a management strategy evaluation that included the population dynamics, stock assessment, and management to determine effects of assessment interval and data-management lag on the probability of overfishing, average catch, average biomass, and the variation in catch. We evaluated assessment intervals ranged from annual assessments to assessments every ten years, and data-management lags (time between the last year of data in the assessment and when new regulations are implemented) ranged from one to three years. Further, we tested the control rules under two life histories (fast and slow) and two data scenarios (good and poor) to identify interactions between these parameters and control rule performance. Increasing assessment interval and data-management lag decreased average catch and biomass across scenarios, with data-management lag having larger than changes to assessment interval. The probability of overfishing was generally higher when assessment intervals were longer whereas the variability in catch decreased with decreasing assessment intervals. Across all performance metrics the effects of poorer data resulted in magnified effects of assessment interval and data-management lag. Our results provide guidance to management by identifying the tradeoffs for management between frequency of assessment and data-management lags.

Introduction

The use of stock assessments to inform management and decision making has increased worldwide. Ideally stock assessments would be conducted on recent data from the fishery and management actions based on the assessment would be implemented immediately. However, two of the challenges of using stock assessments in management decisions are data availability and the allocation of resources to conduct stock assessments in a frequent and timely manner. The use of the most recent data should be critical to stock assessment model performance because management recommendations derived from the most recent years should be the most accurate. However, extended delays between the availability of the most recent data included in the stock assessments and management decision are common. Data-management lags (DML), the time between data collection and implementation of management measures, can extend up to 4 years in some places (e.g. Pacific Management Council). While DMLs and assessment timelines can strongly affect model predictions, few studies have quantified their effects (Shertzer and Prager 2007; Brown et al. 2012; Li et al. in review).

Assessment intervals vary widely among management bodies, and the intervals between stock assessments can have an important effect on management outcomes (Mace et al. 2001, ICES 2012; Li et al., in review). For example, many ICES stock assessments are conducted on an annual basis. (NRC 1998), the Pacific and North Pacific management assessment intervals range from annual to every three years, and the majority of east coast U.S. fisheries assessments are on an order of

every three to five years. However, some stocks managed by the MAFMC have assessments intervals of ten years or greater, such as Illex squid (*Illex argentinus*).

Frequent assessment is important in determining stock status because identifying early trends in biomass can avoid overfishing in the future (Mace 2001). Less productive stocks can be sensitive to assessment frequency, with potentially large decreases in SSB with long periods between assessments (Li et al., in review). Furthermore, if stock size is declining, moving from annual to multiannual assessments could lead to an increase in the risk of spawning stock biomass (SSB) falling below the threshold value and catch limits being held at too high of a level (ICES 2012). Li et al. (in review) found that results of extended assessment intervals were dependent on the approach used to set target harvest, i.e. more successful candidates for longer assessment intervals being those regulated by more conservative control rules.

DML is caused by a variety of interacting factors, but can have detrimental consequences on management success the longer management action is delayed (Shertzer and Prager 2007). In its entirety, the management process can result in a delay of up to three years between data collection and implementation of regulations in most U.S. regions. Shertzer and Prager (2007), discussed the delays in the management of haddock (*Melanogrammus aeglefinus*), cod (*Gadus morhua*), and yellowtail founder (*Pleuronectes Ferrugineus*) in New England, for which fishery management plans were approved only after populations dropped to their lowest recorded levels. Delays can extend five to seven years in some species such as orange

roughy (*Hoplostethus atlanticus*) in South-East Australia (Bax et al. 2003) and up to 12 years in some whale species (Punt and Donovan 2007).

The DML begins after data are collected for a particular stock or set of stocks. After collection, the data must be processed, which includes entering the data, verifying accuracy, aging samples, and any preliminary analyses needed to get information into the appropriate form for the stock assessment. The stock assessment itself can be completed in weeks to months if it is an update of a previously used model, or may take substantially longer if a new assessment methodology is being developed. If the stock assessment undergoes peer review before it is used in management, two to three months may be added to the process. Lastly, requirements of public comment and the promulgation of regulations may add a year or more to developing and implementing management e depending on the region and species. Management processes can extend even longer if management is then delayed due to issues such as scientific uncertainty or to reduce short-term losses in profit for fishers (Brown et al. 2012).

Although DML is present in any management system, its effect has received relatively little scrutiny. A delay of one year has been shown to cause slight increases in SSB and yield and a small negative bias in estimated SSB when comparing a scenario with no lag to an annual lag (Li et al., in review). Delays in management decisions when a fish stock is declining can require more conservative management, thus larger catch reductions in order to rebuild a stock (Shertzer and Prager 2007). Brown et al. (2012), considered the consequences of delayed management in response to ecosystem and climate change. They similarly found that delays reduced harvest

and caused lower and more variable harvest. However, unlike Shertzer and Prager (2007), Brown et al. (2012) found that even with precautionary approaches in management, failing to act on the most recent management advice resulted in significant increases in the risk of collapse.

In the U.S., recent modifications of legislation to guide fisheries management (Restrepo et al. 1998; NMFS 2006) have caused fisheries managers and scientists to consider how often assessments should be conducted and how to design procedures that can quickly feed data into the management system. In the Mid-Atlantic region of the U.S., assessment intervals can range from annual to decadal, and DML can extend up to three years. Assessment interval timelines depend strongly on the fishery and ecosystem importance, stock status, and stock biology (NOAA 2014), while the DML is a result of the data collection and management processes. Similar to other regions of the U.S., the MAFMC has a lengthy process for setting regulations once data have been collected, analyzed and integrated into assessments The MAFMC procedure begins with review and recommendation of an acceptable biological catch (ABC) limit by the MAFMC's scientific statistical committee (SSC). The ABC recommendation is then given to a committee who drafts recommendations for annual catch limits and regulations to achieve those catch limits that are provided to the MAFMC. The MAFMC develops recommendations for regulations. An environmental impact statement is completed which identifies environmental effects of the proposed action and submits alternative actions. Scoping meetings, which are public hearings organized to gather input on the range of issues to be considered are also conducted. A public hearing period is conducted, which is followed by a final

MAFMC recommendation. The recommendations are then sent to NOAA's National Marine Fishery Service where similar steps are followed prior to implementation of the regulation by the U.S. Secretary of Commerce.

Our objective was to use a management strategy evaluation (MSE) to test the effects of stock assessment interval and DML on Mid-Atlantic management performance. The MSE used a simulation approach, which includes both the population dynamics and management processes. A range of management strategies were simulated that included a range of assessment intervals, ranging from annual to decadal, and DMLs, ranging from a one to three year lag. We examined a range of performance measures to represent possible objectives of fishery management including average catch, average biomass, and probability of overfishing. Additionally, we evaluated how life-history, data quality and stock-recruitment dynamics interacted with assessment interval and DML to affect management performance.

Methods

We conducted an MSE to estimate the effects of DML and assessment interval on Mid-Atlantic management performance over a range of data quality, recruitment, and life history scenarios. The MSE included operating and stock assessment models (Figure 2.1). The operating model represented the true dynamics of the stock using an age-structured population model and implemented the management portion of the simulation by conducting stock assessments, which informed the level of ABC removed from the stock based on the selected harvest control rule. The stock assessments were conducted at prescribed intervals (assessment frequency) to

estimate stock biomass and reference points for management. A statistical catch-at-age assessment model was implemented within the MSE. Target catch was then calculated using the MAFMC's harvest control rule, which includes a projection to the year the catch limit will be implemented and uses a probabilistic approach (Shertzer et al. 2010; MAFMC 2011). Alternative management strategies were described by combinations of stock assessment interval (assessments every one, two, three, five, seven and ten years) and DML (of one, two and three years). Each management combination was tested under a range of scenarios of good and poor data quality, fast and slow life history, and high and low variable recruitment variability in order to represent a broad range of potential fisheries. The model was run for a total of 80 years; in the first 30 years the fishery developed with unregulated fishing. During the remaining 50 years the selected management strategy was applied. At the end of each simulation, the performance of the control rule was summarized over the 50-year management period. All models were developed in ADMB (Fournier et al. 2012). Variable definitions and equations are provided in Tables 2.1 and 2.2.

Operating model

The operating model simulated the population dynamics with an age-structured model. The model included 12 age classes for the fast life history, and 20 age classes for the slow life history, where 12+, and 20+ were aggregate age classes or "plus groups" of fish ages 12 or 20 and older. Abundance at age in the first year of the simulation was derived from the expectation from an unfished equilibrium state. We used a Beverton-Holt stock-recruitment relationship with a lognormally

distributed random error to determine recruitment each year (Eq. T1.1). We implemented autocorrelated random errors in the stock-recruitment relationship with a correlation of 0.44 and a standard deviation of either 0.77, which were the average values from a meta- analysis by Thorson et al. (2014), or 1.25, which represented a higher level of variability. Abundance at age was calculated using an exponential mortality model with additive natural and fishing mortality (Eq. T1.2). The weight at age was calculated using length at age from a von Bertalanffy growth model (Eq. T1.4) and an allometric function of length at age (Eq. T1.3). Maturity at age followed a logistic function (Eq. T1.5). The spawning stock biomass (SSB) was the product of maturity-at-age, weight-at-age and abundance-at-age summed over ages for a given year (Eq. T1. 6).

The operating model included a single fishery, and selectivity followed a logistic function. Selectivity of the fishery and natural mortality in the assessment models did not follow exactly the same dynamics as the operating model. Fishery selectivity varied over time by applying an autocorrelated, lognormally distributed error with a standard deviation of 0.1 and autocorrelation of 0.3 or 0.9 to the sf50% parameter (Eq. T1.7 and T1.8), and natural mortality also followed an autocorrelated lognormal random process over time with a log-scale standard deviation of 0.15 and an autocorrelation of 0.3 or 0.9 (Eq. T1.9). Fishing mortality was set to 0.05 yr^{-1} in the first year, after which it increased linearly until it plateaued in year 18 and remained constant until the management period began in year 30. The value of fishing mortality at the plateau depended on the exploitation history. Exploitation scenarios were light, moderate and heavy and used a fishing mortality multiplier (F =

0.5, 1.0, 2.5 x $F_{MSY,}$ respectively) in the plateau year. Total mortality was the sum of the natural mortality and fishing mortality; fishing mortality at age was the product of the selectivity at age of the fishery and the overall fishing mortality rate for a year. Fishery catch-at-age was calculated using the Baranov catch equation (Eq. T1.10). The observed fishery catch was calculated by multiplying total fishery catch by a multiplicative lognormal error with a log-scale standard deviation set at 0.15 (Eq. T1.11).

The operating model also generated catch-at-age expected in a survey as the product of abundance, survey selectivity and survey catchability (Eq. T1. 12). Survey catchability varied according to a random walk on the log scale with normally distributed errors (with a standard deviation of 0.01 or 0.05 depending on the data quality scenario) to allow gradual variation in the catchability over time (Eq. T1.13). The observed index of abundance included observation error with a log-scale standard deviation of 0.3 or 0.7 depending on the data quality scenario (Eq. T1.14). The observed proportions at age fishery and survey were generated by sampling from a multinomial distribution using the true proportions at age (Eq. T1.15) and effective samples sizes of 50 or 200 depending on the data quality scenario.

The "data sets" were provided to a statistical catch-at-age stock assessment model that estimated, among other things, the overfishing limit (OFL) and relative biomass, which were provided to the operating model in order to apply the management control rule. The OFL is the catch that should be achieved by fishing at the limit fishing mortality rate and was calculated by applying the estimated $F_{35\%}$ from the assessment to the projected estimate of abundance. An $F_{35\%}$ mortality rate

was chosen to serve as a proxy for F_{msy} because it is commonly used as a limit fishing mortality reference point for stocks managed by the MAFMC.

After each assessment the operating model implemented the MAFMC P* control rule to find an ABC from the OFL estimated in the assessment (MAFMC 2011). The P* approach adopted by MAFMC assumes that the OFL is lognormally distributed with a CV of 100% and a median from a stock assessment projection to the time the new catch limit would be implemented. The ABC was estimated as the catch that achieves the 40^{th} percentile of the OFL distribution if estimated $B/ B_{35\%}$ (derived from the SPR model in the assessment) exceeded 1.0. If estimated $B/B_{35\%}$ fell below 1.0 then the P* used to calculate ABC decreased linearly to zero until $B/B_{35\%} = 0.10$; below this value the ABC was set to zero and the fishery was closed (MAFMC 2011; Figure 2.2). Once the ABC was determined, the operating model used a numerical search to find the fishing mortality associated with the ABC to apply to the stock. The ABC implemented was constant for the duration of the period between assessments.

Assessment model

The assessment model estimated the OFL using the data generated by the operating model with the first year of data collection beginning in year ten of the operation model simulations and the last year of data being the stock assessment year minus the DML. The assessment model was a statistical catch-at-age (SCAA; - Quinn and Deriso 1999) model that estimated the abundance, fishing mortality, biomass and fishing mortality reference points, and the OFL. The structure of the SCAA model followed the same equations as the operating model, except that survey catchability,

fishery selectivity and natural mortality did not vary over time in the estimation model. The natural mortality in the assessment model was set to the true mean natural mortality of 0.2 for the fast life history and 0.1 for the slow life history. The total negative log likelihood function that was used for model fitting included lognormal distributions for the fishery and survey catch and multinomial distributions for the age composition of the catch (Eq. T.1.16 and T.1.17).

The SCAA required data on the fishery catch-at-age and the survey index of abundance at age. Parameters estimated by the SCAA included recruitment for each year, fishery and survey selectivity parameters, abundance at age in the first year, fully selected fishing mortality for each year and survey catchability. The SCAA model also used the true biological inputs from the operating model including the natural mortality rate, the maturity at age and size at age as well as the estimated selectivity at age to estimate the $F_{35\%}$ reference point. Abundance in the final year of the assessment model was projected forward past the DML years with recruitment and fishing mortality assumed constant in the projection years in order to estimate the OFL for the appropriate management year by finding the catch that would achieve $F_{35\%}$ given the projected abundance at age. $B_{35\%}$ was calculated by multiplying the spawning stock biomass per recruit from fishing at $F_{35\%}$ by the mean estimated recruitment over the time series. The OFL and biological reference points were then returned to the operating model in order to apply the control rule and find the ABC.

Scenarios

Combinations of assessment interval and DML were tested under a factorial design of scenarios that considered alternative assumptions about data quality, stock-

recruitment variability, exploitation history, and life history. We modeled good and poor data quality scenarios. The good data scenario used a coefficient of variation of 0.3 and 0.15 for the total survey and fishery catch, respectively, and an effective sample size of 200 for the proportions at age in the survey and fishery catch. Alternatively, for the poor data scenario a coefficient of variation of 0.7 and 0.15 for the survey and catch, respectively, and an effective sample size of 50 for both the survey index of abundance and fishery catch. Survey catchability also varied by data quality scenarios by random walk using log scale normally distributed errors with a standard deviation of 0.01 for the good data scenario and 0.05 for the poor data scenario. Fishery selectivity varied over data quality scenario by applying an autocorrelated, lognormally distributed error with a standard deviation of 0.1 and autocorrelation of 0.3 or 0.9 to the sf50% parameter for the good and poor data scenario respectively. Natural mortality varied over time by an autocorrelated lognormal random process with a log-scale standard deviation of 0.15 and an autocorrelation of 0.3 for the good data scenario and 0.9 for the poor data scenario. The log-scale standard deviation of the recruitment error was 0.77, the mean recruitment variability from a meta-analysis study by Thorson et al. (2014); additional runs with higher recruitment variability used a log-scale standard deviation for recruitment variability of 1.25. Parameters of the operating model were chosen to represent species with a fast and a slow life history. Life histories were tailored to approximate summer flounder (*Paralichthys dentatus*) for the fast life history and spiny dogfish (*Squalus acanthias*) for the slow life history (parameters in Table 2.2). The fast life history of the summer flounder included early recruitment into the

fishery and early maturation, while the slow life history of the spiny dogfish represented lower natural mortality and late recruitment and maturation. Exploitation scenarios were implemented by including a fishing mortality multiplier (F = 0.5, 1.0, 2.5 x F_{MSY} for the light, moderate, or heavy exploitation) in the pre-management portion in order to determine the abundance at the beginning of the management period. Preliminary model testing showed little difference between exploitation histories; therefore, the 1000 simulations were summarized across exploitation history with the first 333 runs representing an underfished stock, the second 333 runs representing a fully fished stock, and the final 334 runs representing an overfished fished stock.

Performance metrics

The model tracked a range of performance metrics including the true catch, true biomass, probability of overfishing and average annual variability (AAV) of the catch. The catch and biomass performance metrics took the average catch and biomass over the 50 year management period. The probability of overfishing metric was calculated as the proportion of years in which the true fishing mortality exceeded $F_{35\%}$ during the 50 year management period. The AAV in catch was the average of the absolute value of the difference of catch from year to year across the 50 year management period. An ANCOVA was also run on the data set in order to determine individual effects of each data quality, recruitment variability and life history scenarios.

Results

Across all life history and data quality scenarios, increasing assessment interval and DML decreased the average catch, with DML having a larger effect on catch than assessment interval (Figure 2.3). The effects on catch were relatively small for the fast life history and good data scenario, with average catches decreased by 2% for each additional year of DML and 1% with each additional year between assessments. However, effects on catch were magnified in the poor data scenario to a 4% and 3% decrease in catch with an additional year of DML or assessment interval, respectively. Overall, the median average catch decreased by around 20% between good quality and poor quality data for both life history scenarios. Similarly, for both of the life histories with a good data scenario, an annual assessment with three years of DML achieved a similar level of average catch as an assessment every five years with one year of DML. In the poor data scenarios, an annual assessment with three years of DML had similar median average catch to an assessment every seven years with one year of DML. The slow life history scenario displayed less of an effect of assessment interval and DML on average catch in the good data scenario compared to the fast life history. However, the effects of DML were similar to those from the fast life history in the in the poor data scenario. On average the slow life history saw a 1% decrease in the median average catch with each additional year of DML for the good data scenario, which was magnified to a 5% decrease in the median average catch in the poor data scenario. Furthermore, with each additional year between assessments for the slow life history scenario the median average catch decreased by 2% for the good data scenario and 3% for the poor data scenario. The amount of

recruitment variability had relatively little effect on management performance (3% increase in median for biomass and catch and 7% increase in the probability of overfishing) as well as larger variability across all performance metrics (Table 2.3).

The effect of DML and assessment frequency on average population biomass differed with life history and data quality (Figure 2.4). For the fast life history and good data scenario, each additional year of DML and time between assessments caused about a 1% decrease in average biomass. The median average biomass decreased about 3% with each additional year of DML and 2% with each additional year between assessments in the fast life history and poor data scenario. The poor data quality scenarios resulted in about 5% lower average biomass for the fast life history relative to the good data scenario, while the slow life history saw an 11% difference. The slow life history and good data scenario saw a larger effect of assessment interval than DML with a 1% decrease in biomass with each additional year of DML compared to a 2% decrease in biomass with each additional year between assessments. Effects were reversed in the slow life history with poor data scenario with each year of DML causing a 6% decrease in the median biomass compared to a 5% decrease with each additional year between assessments.

The probability of overfishing was affected by both data quality and assessment interval, but showed little response to DML, except when combined with longer assessment intervals (assessments every seven and ten years) across all scenarios (Figure 2.5). Median probabilities of overfishing ranged from 0.25 to 0.4 across all scenarios with relatively small changes in the probability of overfishing with assessment intervals less than seven years (0.25-0.3). The effect of DML was

negligible for the fast life history and poor data scenario, while the probability of overfishing increased 7% with each additional year between assessments. The effect of assessment interval was reduced for the good data scenario with a 4% increase in the probability of overfishing for each additional year between assessments. The effect of DML, conversely, was larger (2% increase in the probability of overfishing with each additional year of DML) relative to the poor data quality scenario. For fast life history scenarios the mean probability of overfishing was around 10% higher when conducting an assessment with poor data compared to an assessment with good data. In contrast, the difference between the two data quality scenarios was only about 2% for the slow life history. Each additional year between assessments caused a 6% increase in the probability of overfishing, and each additional year of DML resulted in a 4% increase in the probability of overfishing in the scenarios with good data. In the poor data scenario assessment interval and DML caused a 2% increase in the probability of overfishing.

Catch AAV generally decreased as DML and assessment interval increased (Figure 2.6). All life history and data quality scenarios followed a consistent downward trend with each additional year between assessments for assessment intervals up to five years. The catch AAV increased for the seven and ten year assessment intervals. For the fast life history scenarios, assessment interval caused a larger response in AAV than the DML. The catch AAV decreased about 7% with each additional year of DML for both the good and poor data scenarios. The effect of assessment interval was an 11% decrease with each additional year. The catch AAV was substantially higher in the poor data scenario for both life histories, showing the

large inter-annual variation in catch that can occur as a result of low data quality. DML had a small effect on catch AAV for the slow life history in both poor data quality scenarios.

ANCOVA results determined that all of the performance metrics tested (catch, biomass, probability of overfishing and AAV in catch) were significantly affected by data lag and stock assessment interval. Life history, data quality scenarios and recruitment deviations also significantly affected performance metrics. I

Discussion

We found substantial differences in management performance as a result of assessment interval and DML across a range of scenarios. Specifically, increases in DML and assessment interval resulted in decreases in both the median catch and biomass. Increases in DML caused larger changes relative to increases in assessment intervals, on average, for all performance metrics besides probability of overfishing and were especially noticeable in the poor data scenarios. The effects of DML and assessment interval on the performance metrics varied among the life history and data quality scenarios. For example, for average catch the effects of DML and assessment interval were relatively low for the fast life history and good data scenarios, with average catches decreased by 2% and 1%, respectively, but were magnified in the poor data scenario to a 4% and 3% decrease in catch with an additional year of DML or assessment interval, respectively. Larger changes in performance metrics were evident with the fast life history compared to slow life history scenarios, but effects varied across performance metrics.

The effect of DML in our study was similar to that described in other studies (Shertzer and Prager 2007; De Leeuw et. al. 2008; Brown et al. 2012) in that delaying the management process can result in increased probability of collapse and variability in harvest. All studies agreed that lengthy DMLs have negative effects on achieving long-term management goals, and can result in unsatisfactory management outcomes. Our study examined shorter DML periods than previous studies (Shertzer and Prager 2007; Brown et al. 2012). However, both previous work and our results reported here found that increases in DML can cause overfishing resulting in reduced catch and biomass. Our results for DML were similar to those from Li et al. (in review), with small effects (<2%) when moving from a DML of one year to two years for shorter assessment intervals (annual, two year and three year intervals). However, Li et al. (in review), recommended there was no reason to rush the data collection and management process because their changes in performance metrics were small. They did, however, warn against extrapolating results beyond the frequencies tested in their study. Assessment intervals and DMLs of at least two years are common in many fisheries, and, therefore, reducing DMLs and assessment intervals may improve management performance.

Our estimates of the effects of assessment interval on management performance were similar to previous findings (Mace et al. 2001; Li et al. in review), but we evaluated a wider range of assessment intervals (up to 10 years). Previous studies used a maximum interval of assessments every five years. Increases in assessment intervals caused decreases in average catch and biomass, increases in the probability of overfishing and increases in the catch AAV similar to Mace et al.

(2001) and Li et al. (in review). One interesting finding from our study involved an increase in the effects of DML when assessment intervals increased past five years. For example, although the probability of overfishing saw little change with increasing DML in the shorter assessment intervals (around 2% change with each additional year of data lag), changes in the probability of overfishing with each additional year of DML increased 6% for the seven and ten year intervals. The interaction of DML and assessment interval was noticeable across both life history and data quality scenarios and highlights a breakdown in management performance when prolonged assessment intervals are paired with extended DMLs. Overall Li et al., (in review), saw slightly smaller effects than we did across all assessment interval scenarios. Results from their lower productive populations were closer to our results with lower SSB and yield for assessment interval. We may have seen stronger effects of assessment intervals because we used a different control rule, which can effect ABC calculations, as well as increased uncertainty in our stock assessments due to the fishery selectivity, natural mortality and survey catchability varying overtime in the operating model, but assumed constant in the assessment model.

Average variation in catch should decrease as assessment intervals increase because catch was constant at the ABC between assessments. Because large fluctuations in catch are highly unsatisfactory to stakeholders as instability in catch can cause social and economic problems (Holland 2010), the results of this performance metric support the use of longer assessment intervals if minimizing variation in catch is a high priority goal. Findings of decreased AAV in catch with

increasing stock assessment interval and DML highlight common competing objectives of fisheries management goals.

Data quality can affect management performance (McGoodwinn et. al. 2007; Smith et. al. 2011), and we observed substantial degradation of performance in our poor data quality scenarios. The large decreases in the median biomass and catch and increases in the probability of overfishing seen in the poor data scenarios of our study are likely due to poor assessment model performance. The poor data scenario included the larger variances for total fishery and survey catch as well as smaller effective sample sizes for the catch and survey age compositions. Additionally, the poor data scenarios included larger changes in survey catchability, natural mortality, and fishery selectivity, which were not included in the SCAA model. These factors caused estimates of biomass in the last year and reference point estimates to be relatively less accurate than the good data scenario. Because our management model then used a short-term projection to determine the catch limit, uncertainty in the abundance at age in the last year would likely be magnified. Even with improved population projections, we would still expect to see poorer management results in scenarios with lower quality data (De Leeuw et. al. 2008). Findings of this study show that another option besides using the best available data is to include better data in stock assessments to decrease the effects of stock assessment interval and DML.

The effects of life history on management performance were similar to results from previous studies (Mace et. al. 2001; Shertzer and Prager 2007; De Leeuw et. al. 2008; Brown et. al. 2012, ICES 2012, Li et. al., in review). DML and assessment interval had smaller effects in the slow life history scenarios than the fast life

histories. These reduced effects of DML and assessment interval may be explained by smaller fishing mortality limit reference point (F_{lim}) for the slow life history than the fast life history; 0.07 and 0.19 respectively. Lower fishing and total mortality rates may cause reduced population responses due to more stability in the stock and less fluctuations in fishing effort year to year (Patterson and Résimont 2007).

Recruitment variability and autocorrelation can decrease the success of extended assessment intervals. Under our high recruitment variability scenario, average catch and biomass were decreased on average 3% for each additional year of DML and 2% with each additional year between assessments. We also saw, on average, a 7% increase in the probability of overfishing with each additional year between assessments for the fast life history, poor data scenario when moving from a recruitment error standard deviation of 0.77 to 1.25. Higher recruitment variability resulted in larger declines in biomass because initial catch was held at too high of a level between assessments (ICES 2012).

Reduction of extended DMLs are essential to improved fishery management. A system that allows up to three years of lag between collection of data and implementation of regulations, sacrifices considerable amounts of biomass and catch, up to 7% and 5% respectively. Additionally, when extended DML is paired with longer assessment intervals, suboptimal outcomes are even more likely. Some potential approaches to reduce DML include a shorter management process and faster data processing. For example, the North Pacific Fishery Management Council (NPFMC) has been successful in achieving a one year DML in the majority of its fisheries (AFSC 2014). First, the NPFMC utilizes real-time in-season reporting. In

addition to active reporting, the NPFMC uses projections to set catch limits two years in the future, which goes through the public comment process the year prior to regulations being set, effectively removing the time lag for public comment in that year. Finally, the NPFMC has reduced the assessment timeline by using partial ageing data for the most recent year within the stock assessment for some species (AFSC 2014); in these cases, only the survey data is aged. Reducing DML does require a quicker turnover of data and a faster management process which may result in increased resource availability. Decreasing the amount of data within stock assessments may also degrade model performance (Ono et al. 2014). The consequences of using only immediately available data (less data within assessments) in order to decrease data lag in comparison to data lag effects is an important matter to explore further. While the use of many of the techniques to decrease DML may not be immediately achievable in some regions, it identifies potential possibilities that should be considered in the future.

Tables and Figures

Table 2.1. Equations governing the population and data-generating dynamics in the operating model.

	Equation	Description
1	$R_t = \frac{S_{t-a_R}}{1-\left(\frac{5h-1}{4h}\right)+\left(1-\frac{S_{t-a_R}}{S_{eq}}\right)} e^{\theta_R - 0.5^{\sigma_R^2}}$	Stock-recruit relationship
2	$N(a,t) = \begin{cases} R(t) & a = a_R \\ N(a-1,t-1)e^{-Z(a-1,t-1)} & a_R < a < a_{max} \\ N(a-1,t-1)e^{-Z(a-1,t-1)} + & a = a_{max} \\ N(a,t-1)e^{-Z(a,t-1)} & \end{cases}$	Abundance at age
3	$L_a = L_\infty(1-e^{-k(a-a_0)})$	Length at age
4	$w_a = bL_a^{\ c}$	Weight at length
5	$m_a = \frac{1}{1+e^{-\left(\frac{a-m_{50\%}}{m_{slope}}\right)}}$	Maturity at age
6	$S_t = \sum_{a=a_R}^{a_{\max}} m_a w_a N_{t,a}$	Spawning biomass
7	$S_{t,a} = \frac{1}{1+e^{-\left(\frac{a-S_{50\%,t}}{S_{slope}}\right)}}$	Selectivity at age in the fishery
8	$S_{50\%_t} = \bar{S}_{50\%} e^{\varepsilon_{S_t} - 0.5_s^2}$ $\varepsilon_{st} = \rho_s \varepsilon_{s_{t-1}} + \sqrt{1-\rho_s^2}\phi_t$ $\phi_t \sim N(0,\sigma_s^2)$	Time-varying and autocorrelated selectivity

9 $$M_t = Me^{\varepsilon_{M_t} - 0.5\sigma_M^2}$$

$$\varepsilon_{M,t} = \rho_M \varepsilon_{M_{t-1}} + \sqrt{1-\rho_M^2}\phi_{M,t}$$

$$\phi_{M,t} \sim N(0, \sigma_M^2)$$

Time-varying and autocorrelated natural mortality

10 $$C_{t,a} = \frac{F_{t,a}}{Z_{t,a}}(1-e^{-Z_{t,a}})N_{t,a}W_{t,a}$$

Baranov-catch Eq.

11 $$C_{obs,t} = C_t e^{\varepsilon_{C_t} - 0.5\sigma_C^2}$$

$$\varepsilon_{C_t} \sim N(0, \sigma_C^2)$$

Observed catch

12 $$I_{t,a} = q_t s_{s,a} N_{t,a}$$

True index of abundance

13 $$q_t = q_{t-1} e^{\varepsilon_{qt}}$$

$$\varepsilon_{q_t} \sim N(0, \sigma_q^2)$$

Catchability

14 $$I_{obs_t} = e^{\varepsilon_{I_t} - 0.5\sigma_I^2} \sum_a I_{a,t}$$

$$\varepsilon_{I_t} \sim N(0, \sigma_I^2)$$

Observed index of abundance

15 $$P_{obs_t} = \frac{1}{n}\boldsymbol{\Theta_t}$$
$$\boldsymbol{\Theta_t} \sim Multinomial(n, \mathbf{p_t})$$
$$\mathbf{p_t} = \frac{1}{I_t}\left(I_{a_r,t}, \ldots, I_{a_{max},t}\right)$$

Proportion of catch-at-age

16 $$\ell_t = 0.5n\log(\sigma_C^2) + \sum_t \left(\log(C_{obs_t}) - \log(C_{est_t})\right)^2$$

$$\ell_t = 0.5n\log(\sigma_I^2) + \sum_t \left(\log(I_{obs_t}) - \log(I_{est_t})\right)^2$$

Lognormal likelihood components for the catch and index of abundance

17

$$\ell_t = -E_C \sum_t \sum_a p_{obs,C_{t,a}} \log(p_{est,C_{t,a}})$$

$$\ell_t = -E_I \sum_t \sum_a p_{obs,I_{t,a}} \log(p_{est,I_{t,a}})$$

Multinomial likelihood components for the proportion of the catch and index

Table 2.2. Symbols, specified life history and time-varying parameters of model.

Parameter	Description	Life history	
		Slow	**Fast**
a_r	Age at recruitment	5	3
a_{max}	Aggregate age class	20	12
M	Natural mortality rate	0.1	0.2
R_0	Unfished equilibrium recruitment	$1x10^6$	$1x10^6$
h	Steepness	0.6	0.75
a_0	Age at length = 0	0	0
L_∞	Asymptotic maximum length	90	90
K	Growth rate	0.07	0.13
B	Length-weight relationship scalar	3.5×10^{-6}	3.5×10^{-6}
C	Length-weight relationship exponent	3	3
m_{50}	Age at 50% maturity	7	3.5
m_{slope}	Slope of maturity function	1	1
s_{f50}	Mean age at 50% selectivity in the fishery	7	3.5
s_{s50}	Mean age at 50% selectivity in the survey	5.3	2.6
s_{fslope}	Slope of fishery selectivity function	1	1
s_{sslope}	Slope of survey selectivity function	1	1
	Time Varying parameters		
σ_R	Standard deviation of stock-recruit relationship	0.77, 1.25	
Φ_R	Autocorrelation in recruitment	0.44	
σ_M	Standard deviation of time-varying M	0.15	
Φ_M	Autocorrelation in M	0.3, 0.9	
σ_f	Standard deviation of age at 50% selectivity in fishery	0.1	
Φ_f	Autocorrelation in fishery selectivity	0.3, 0.9	
σ_C	Standard deviation of catch estimates	0.15	
σ_I	Standard deviation of survey estimates	0.29, 0.63	
	Additional model variables		
A	Age		
a_R	Age at recruitment		
T	Year		
R	Recruitment		

S	Spawning biomass
Sf	Fishery selectivity
Ss	Survey selectivity
N	Abundance
M	Maturity
C	Catch
I	Index of abundance
Q	Catchability
Z	Total mortality
W	Weight-at-age
M	Maturity-at-age
F	Fishing mortality
E	Effective sample size of the catch/ index
N	Number of observations
$Pobs$	Observed proportion at age
$Pest$	Estimated proportions at age
$\boldsymbol{\Theta_t}$	Multinomial function
ℓ_t	Likelihood function
ε_{s_t}	Error for selectivity
ε_{q_t}	Error for catchability
ε_M	Error for natural mortality
ε_{I_t}	Error for index of abundance
ε_{C_t}	Error for catch
σ_q	Standard deviation for catchability
ρ_M	Correlation coefficient of natural mortality
ρ_s	Correlation coefficient of selectivity

Table 2.3. Median values for recruitment variability scenarios across performance metrics. SA refers to stock assessment intervals and DLM refers to data-management lag combinations.

Fast life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	88859	651337	0.2353	0.1247
SA1, DML2	87139	650904	0.2549	0.1221
SA1, DML3	84271	640642	0.2745	0.1163
SA2, DML1	87890	642709	0.2549	0.1049
SA2, DML2	86188	643914	0.2549	0.1020
SA2, DML3	83207	625885	0.2745	0.0961
SA3, DML1	86520	635407	0.2745	0.0970
SA3, DML2	84847	631522	0.2745	0.0924
SA3, DML3	80621	620267	0.2941	0.0870
SA5, DML1	84505	618817	0.2941	0.0869
SA5, DML2	80690	606080	0.3137	0.0791
SA5, DML3	76276	583778	0.3137	0.0743
SA7, DML1	80947	596829	0.3137	0.0893
SA7, DML2	75743	580578	0.3137	0.0807
SA7, DML3	71639	545995	0.3333	0.0746
SA10, DML1	76685	571242	0.3333	0.0816
SA10, DML2	72555	547913	0.3529	0.0712
SA10, DML3	64033	471077	0.3725	0.0651
Slow life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	85482	1079880	0.2157	0.1051
SA1, DML2	83196	1062605	0.2353	0.1049
SA1, DML3	81141	1038235	0.2745	0.1048
SA2, DML1	82647	1060110	0.2549	0.0858
SA2, DML2	81006	1040820	0.2549	0.0853
SA2, DML3	78731	1011020	0.2745	0.0845
SA3, DML1	80755	1035620	0.2549	0.0773
SA3, DML2	78848	1021600	0.2745	0.0760
SA3, DML3	76764	984408	0.2941	0.0755
SA5, DML1	81999	984698	0.2941	0.0665
SA5, DML2	79941	955132	0.2941	0.0652
SA5, DML3	77818	919578	0.3333	0.0650
SA7, DML1	74253	936258	0.2941	0.0662
SA7, DML2	73115	893060	0.3333	0.0646

SA7, DML3	70882	852989	0.3529	0.0645
SA10, DML1	68483	863284	0.3333	0.0601
SA10, DML2	67151	813269	0.3725	0.0571
SA10, DML3	63916	731213	0.3922	0.0562
Fast life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	73307	628711	0.2941	0.2151
SA1, DML2	69198	606077	0.2745	0.2072
SA1, DML3	63615	560657	0.2941	0.1983
SA2, DML1	70350	621079	0.2941	0.1721
SA2, DML2	65976	587776	0.2941	0.1658
SA2, DML3	61499	544146	0.3137	0.1579
SA3, DML1	68639	597018	0.2941	0.1550
SA3, DML2	64303	568067	0.2941	0.1489
SA3, DML3	59494	495378	0.3137	0.1428
SA5, DML1	66753	581235	0.2941	0.1339
SA5, DML2	60410	533683	0.3137	0.1254
SA5, DML3	56966	471217	0.3137	0.1190
SA7, DML1	62676	538751	0.3137	0.1350
SA7, DML2	56582	507100	0.2941	0.1245
SA7, DML3	51674	439601	0.3137	0.1202
SA10, DML1	56603	483412	0.3333	0.1176
SA10, DML2	53129	468120	0.3529	0.1084
SA10, DML3	50361	395472	0.3725	0.1051
Slow life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	68201	1016910	0.2745	0.1990
SA1, DML2	65055	951658	0.2843	0.1979
SA1, DML3	61910	899883	0.2941	0.2003
SA2, DML1	66414	993154	0.2745	0.1546
SA2, DML2	62546	937609	0.2745	0.1528
SA2, DML3	59798	843206	0.3137	0.1561
SA3, DML1	64538	953185	0.2941	0.1336
SA3, DML2	61237	891199	0.2941	0.1337
SA3, DML3	57085	808425	0.2941	0.1355
SA5, DML1	65172	901107	0.2941	0.1133
SA5, DML2	60961	837497	0.2941	0.1127
SA5, DML3	57844	738857	0.3137	0.1147
SA7, DML1	58127	796522	0.2941	0.1081
SA7, DML2	53187	730797	0.2941	0.1065
SA7, DML3	52080	652936	0.2941	0.1083

SA10, DML1	52925	698217	0.2941	0.0974
SA10, DML2	49740	627219	0.3333	0.0969
SA10, DML3	44723	539343	0.3725	0.0973

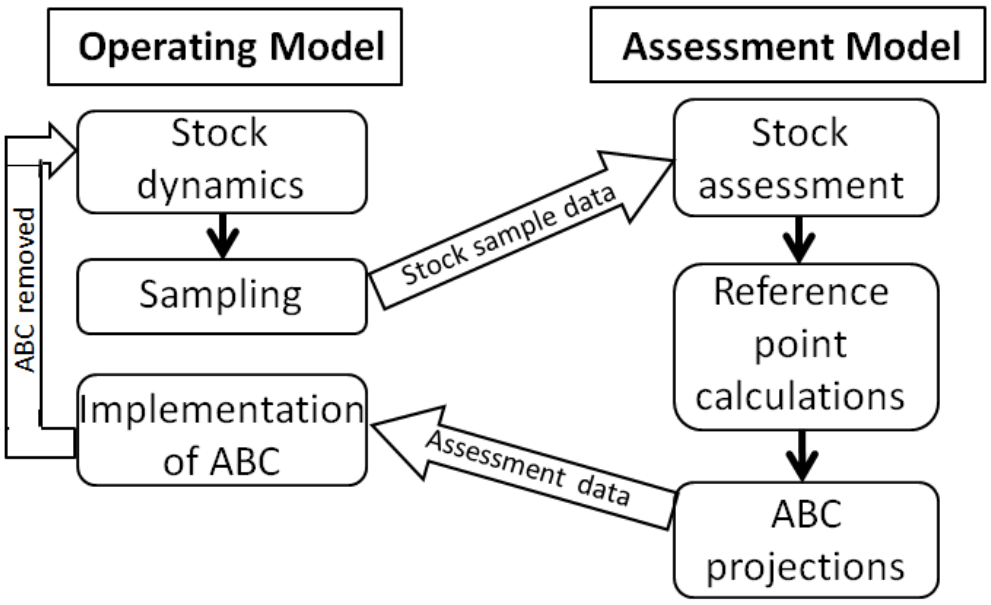

Figure 2.1. Flow diagram of management strategy evaluation model with both operating and estimation models.

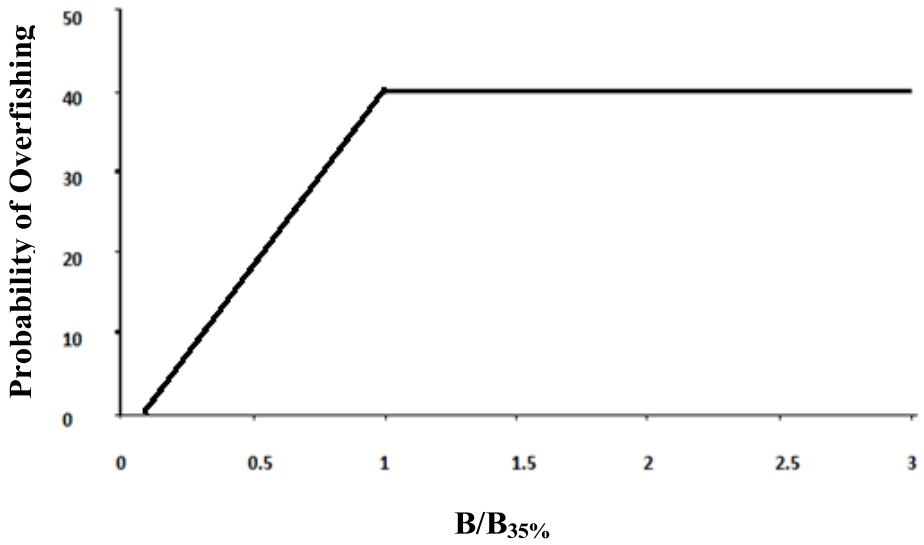

Figure 2.2. Mid-Atlantic P* approach showing decreasing probability of overfishing with a declining $B/B_{35\%}$ ratio.

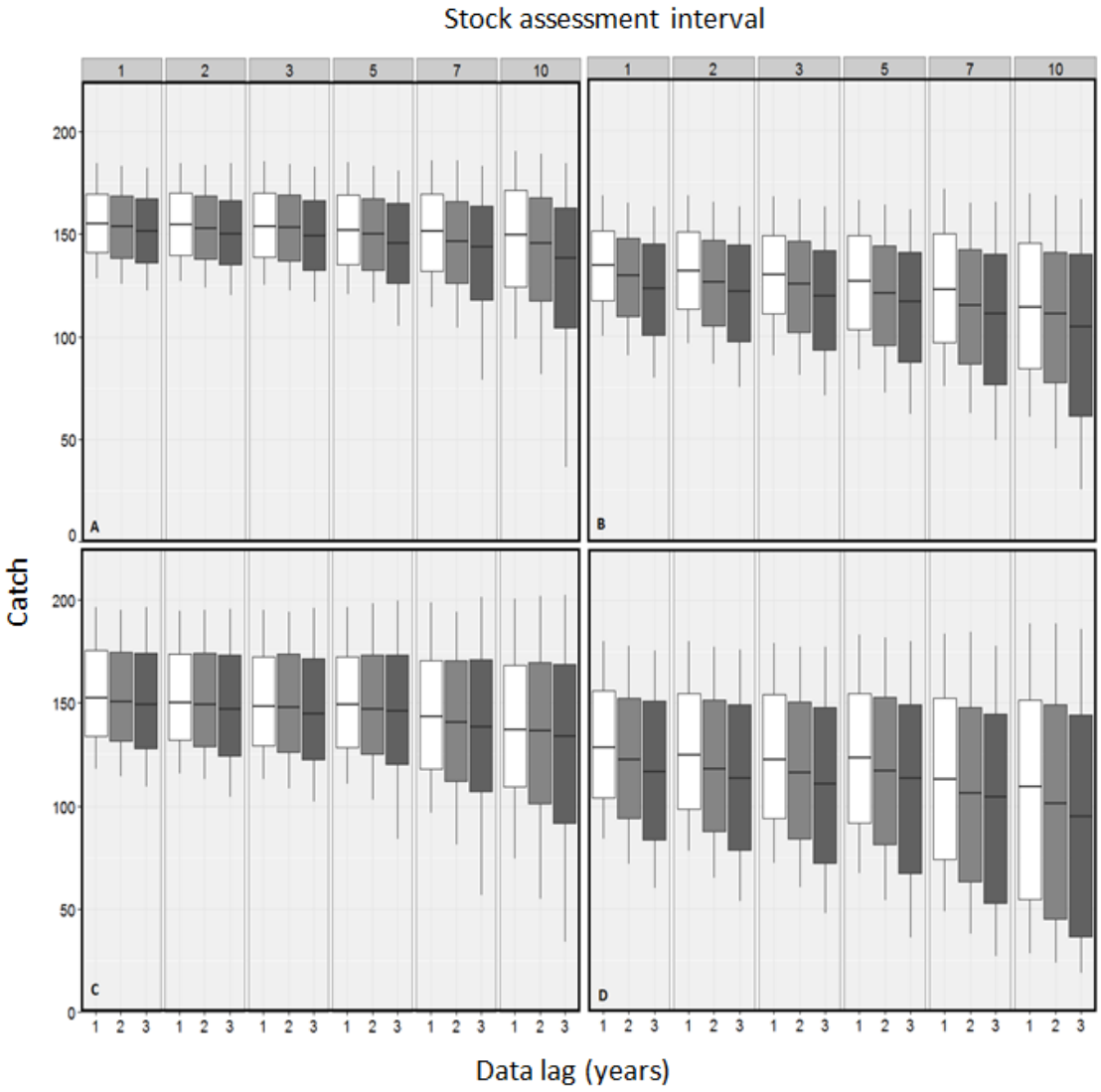

Figure 2.3. Box plots of the catch for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles.

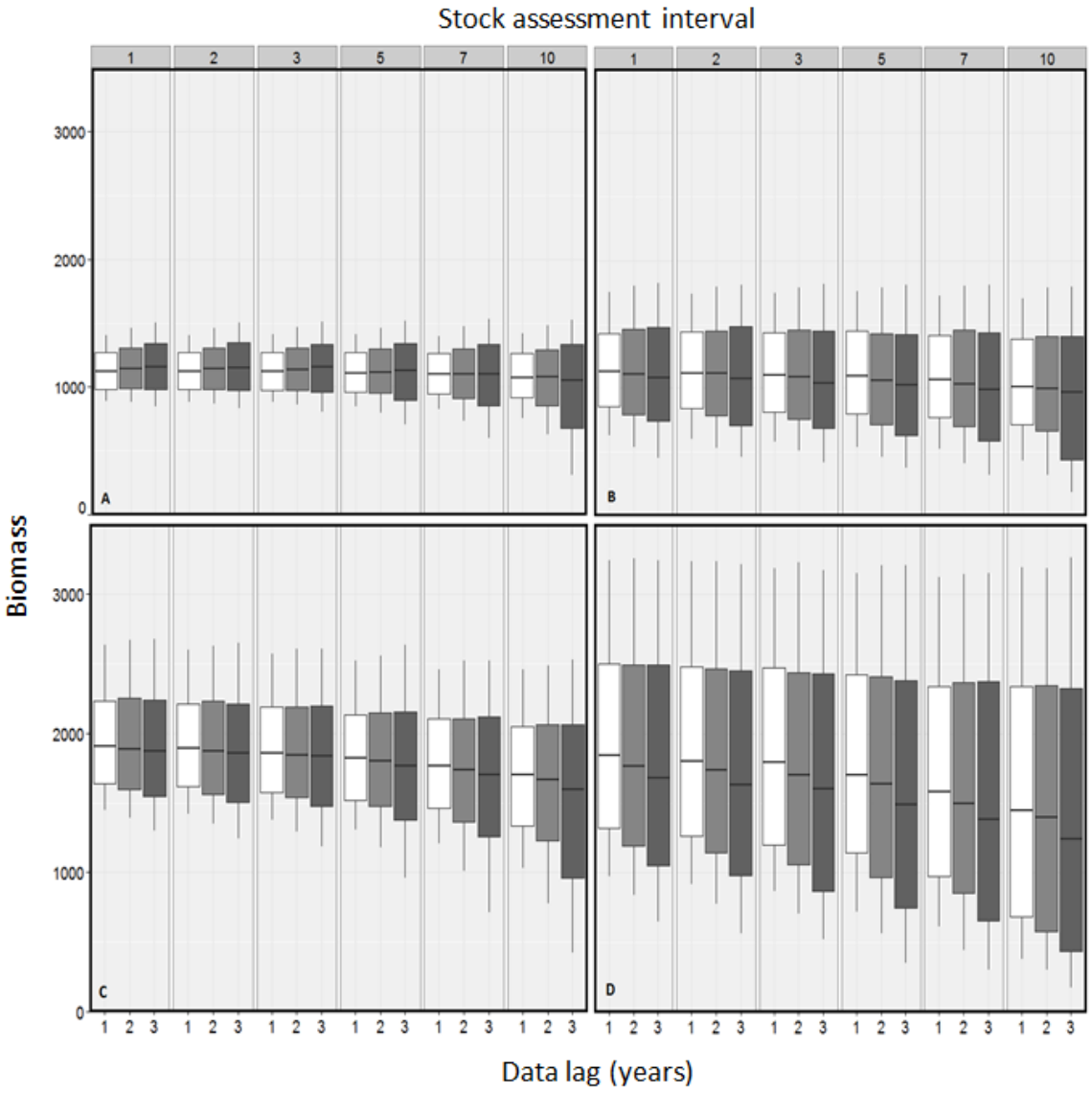

Figure 2.4. Box plots of the biomass for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

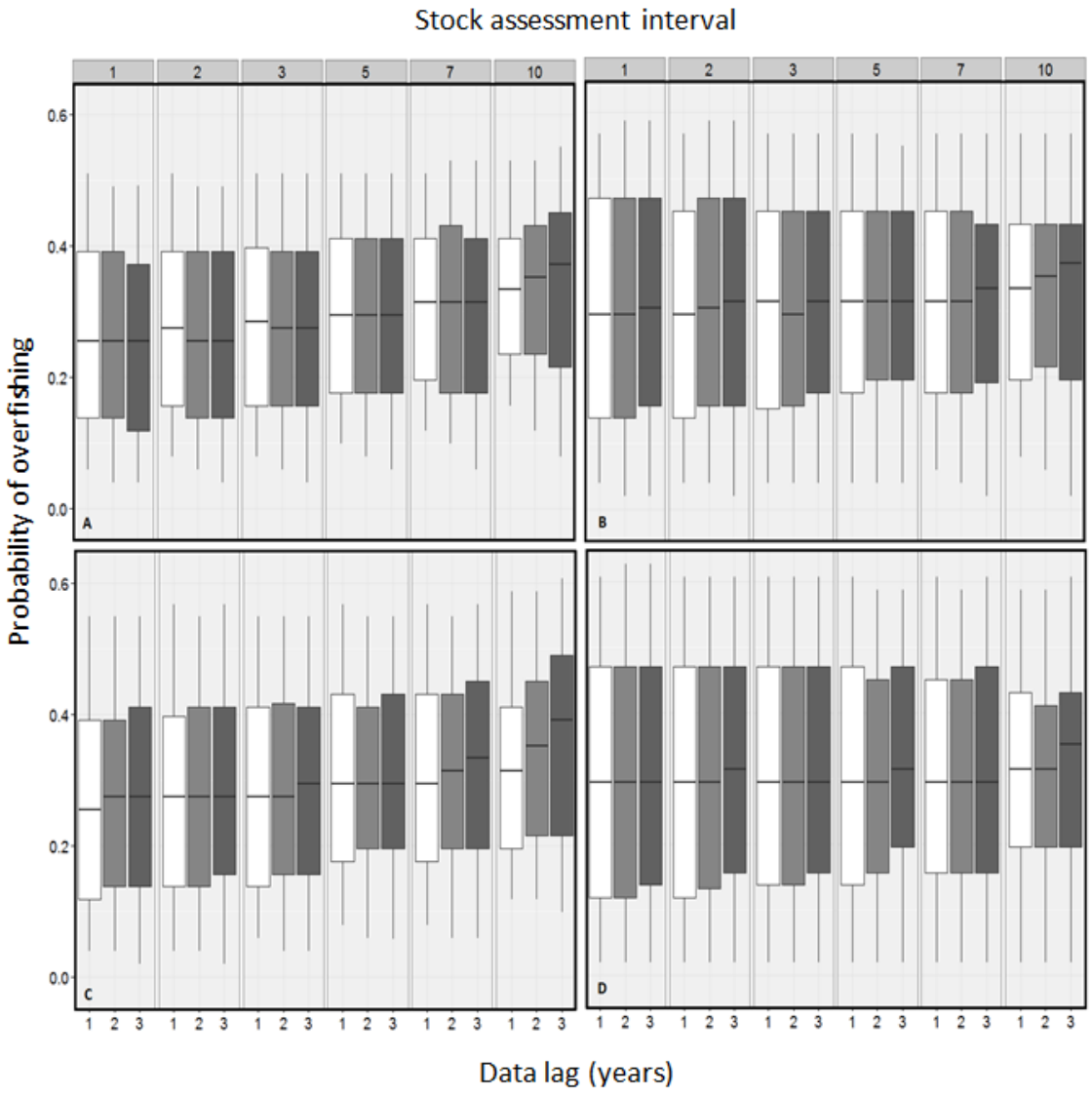

Figure 2.5. Box plots of the probability of overfishing for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

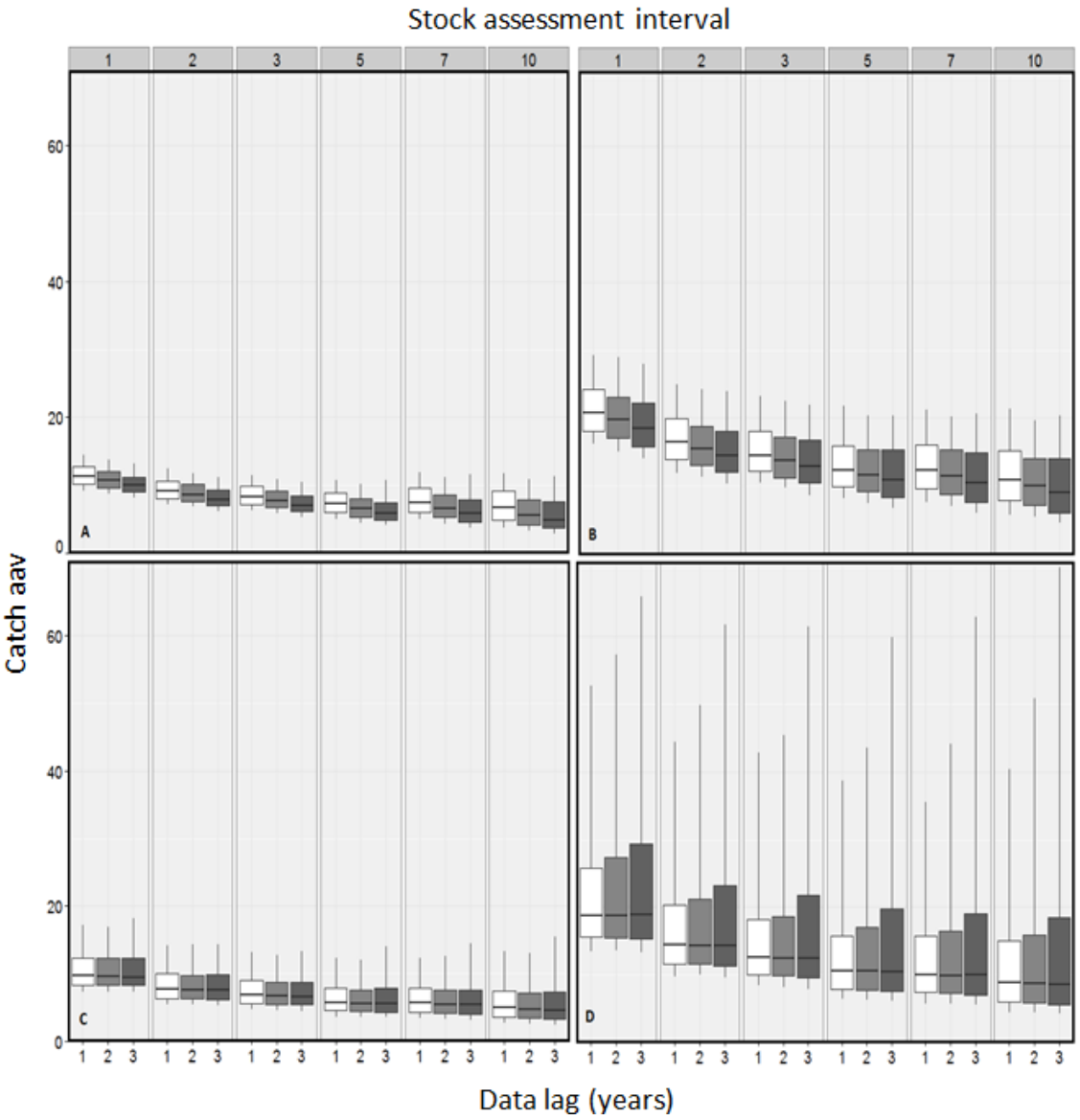

Figure 2.6. Box plots of the catch average annual variation (aav) for each life history and data scenarios: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

Chapter 3: Performance of lag reduction methods for the Mid-Atlantic fishery management

Abstract

Use of the most recent data in stock assessments can be crucial to meeting management goals. Even so, many stock assessments throughout the world experience data-management lags (i.e., the time between data collection and management implementation) that can extend for more than three years. Prior research has suggested that management performance is improved by decreasing the length of the decision making process and using those currently available data instead of waiting for all of the data from a year to be ready. We used a management strategy evaluation to test three methods to reduce data management lags by using partial data in the last year of the assessment: 1) age-composition data for the terminal year of the survey, but no age-composition for the fishery catch, 2) full survey age-composition with reduced quality age-composition data for the catch in the terminal year of the stock assessment (to represent using prior years' age - length data to age the catch), and 3) reduced data for the age-compositions of the survey and catch in the terminal year of the assessment (to represent using prior years' data to age the catch and indices). These methods were tested against controls with one year of data and two years of data lag. Performance metrics were output from the models and included the average catch, average biomass, probability of overfishing and the average annual variation in catch. Lag reduction methods that included some information about the age-composition of the catch and survey performed about as well as not having lag, but they sometimes resulted in higher probabilities of low biomass. Data-

management lag and lag reduction methods should be considered when designing fishery management plans.

Introduction

The application of sophisticated assessments and improved identification of uncertainty have improved fisheries management (Caddy and Cochrane 2001). Yet, even with enhanced techniques, there are still many practical problems in fisheries management throughout the world (Honey et al. 2010, Chen et al. 2002). Examples of such a management issue are the inclusion of timely data within the stock assessment and the lengthy regulation processes that delay implementation of catch limits. Together these represent a data-management lag imposed on the process. Previous studies have noted the importance of using the newest available data in assessments and the decrease in management performance when lengthy data-management lags (the time between data collection and management implementation using that data) are present (Shertzer and Prager 2007; Brown et al. 2012; Li et al., in review; Sylvia Chapter 2). Several approaches to reducing the data-management lag have been used. For age-structured models, many of these approaches have involved using incomplete age-structure information (e.g., using previous year's age and length composition data to estimate age compositions in the current year).

DML is caused by a combination of data preparation, stock assessment, management processes and management delays (Shertzer and Prager 2007; Sylvia Chapter 2). For stocks that use age-structured assessments to provide management advice, the process begins with fishery catch and fishery-independent survey data being collected and processed, usually from multiple sources. Fishery catch can

sometimes be challenging to collect, especially for fisheries with a recreational sector or commercial fisheries that operate in multiple jurisdictions. After the data are collected, they are then processed by verifying and checking the data. One of the lengthiest portions of data collection and processing is the aging. Because aging otoliths can be an intricate and laborious process, it alone can add up to a year to the DML (Begg et al. 2005). Following ageing, preliminary analyses are completed to modify information into the appropriate form for use in the stock assessment. The stock assessment period can then range from a month for updates of an existing model to much longer if new assessment methodology is developed. External peer reviews of stock assessments, done in many parts of the world, can add two to three months to the process. The management portion of DML which includes development and implementation of regulations can take up to a year depending on the region. With all parts of DML adding to the timeline, the management process can see delays up to four years in some U.S. regions. Management delays can extend up to five years in Australian commonwealth fisheries (Smith et al. 2008) and can be even longer (e.g., delays of up to 15-25 years in Lake Ijsselmeer in the Netherlands; de Leeuw et al. 2008).

Few studies have been conducted on the effects of DML (Shertzer and Prager 2007; Brown et al. 2012; Li et al., in review). Each additional year of DML, when compared to a control scenario with one year of DML, caused around a 5% decrease in biomass and catch for poor data quality scenarios while causing up to a 6% increase in the probability of overfishing when paired with longer assessment intervals (Sylvia Chapter 2). We found the largest effects of DML were apparent with

the longer assessment intervals and lower quality data. A study that tested the effect of DML in the management of yellow perch (*Perca flavescens*) in the U.S. Great lakes, found relatively small (<1%) decreases in spawning stock biomass (SSB) and yield when DMLs were increased from one to two years (Li et al., in review). However, increasing DML by delaying management actions can allow continued overexploitation, leading to rigid rebuilding plans and increased probability of stock collapse (Brown et al. 2012; Shertzer and Prager 2007). Waiting for more data may actually be harmful if it delays management even longer (Shertzer and Prager 2007). Slow, stepwise effort reductions in the management of Lake Ijsselmeer in the Netherlands resulted in meeting short term-interests, but failed in recovery of fish stocks and led to stock collapse (de Leeuw et al. 2008).

Strategies to reduce the effects of DML have been developed. Reducing harvest is a potential solution to mitigate the effects of delays, but this approach does little to actually solve the DML problem within the management system (Brown et al. 2012). Making best use of the available data may help to avoid further stock decline or poor management outcomes. Regions such as the North Pacific Management Council (NPMC) have adopted methods that allow them to use partial or missing age-composition data for the most recent year of stock assessments. Converting length composition to age-composition frequently introduces delays due to the aging process, but length composition data is typically available relatively quickly. Predicting ages of fish in the most recent year can often be accomplished by using length composition data in that year and length and age-composition data in previous years to make an inverse age-length key for the most recent year (Quinn and Deriso,

1999). Utilizing past data to estimate age-composition in the most recent year can significantly decrease the data lag that occurs as a result of the aging process. The North Pacific Fishery Management Council (NPFMC) has been successful in achieving a one year DML in majority of its fisheries by using some of the previously mentioned techniques (AFSC 2014). In addition to decreasing the management process by setting catch limits two years in advance, the NPFMC has cut down the data lag by using real time, in-season reporting for catch data and by using assessment models that do not require catch-at-age data in the last year of the stock assessment for some species (AFSC 2014). To accomplish this reduction in DML, NPFMC ages only the survey data, and an inverse age-length key is used to estimate the terminal age-composition of the catch. However, we were unable to find any studies that evaluated the costs and benefits of using such methods to decrease data-management lag (DML).

Our study builds upon previous study recommendations, specifically Shertzer and Prager (2007), by testing methods that use available data to fill in gaps that would otherwise be present in the most recent year. We used a management strategy evaluation (MSE) of the U.S. Mid-Atlantic fisheries management system to test the efficacy of several methods for reducing data lag in the management process, all of which use partial data in the last year of the stock assessment. The MSE simulated the population dynamics, the stock assessment and management process, based on data availability for stocks managed by the Mid-Atlantic Fishery Management Council (MAFMC). We evaluated effectiveness of the approaches using a range of performance measures, which represent how well fishery management objectives

were achieved. We tested the approaches under a range of data quality, life history, and stock-recruitment scenarios.

Methods

We conducted an MSE to test the performance of several methods of reducing DML for MAFMC stocks. We tested two controls (C) and three lag reduction (LR) methods in a two-by-three factorial experiment. The two control rules were C1) a control rule with 1-year data lag and C2) a control with two years of data lag. The lag reduction methods were: LR1) using age-composition data for the terminal year of the survey, but no age-composition for the catch, LR2) full survey age-composition data, but reduced quality age-composition data for the catch in the terminal year, LR3) reduced quality data for both the survey and catch age-compositions in the terminal year of the assessment. The reduced quality age-composition approaches represented cases in which size-at age of the stock is variable such that using prior years' data would inject additional error into the age composition data. For these approaches, length data are available for the most recent year and size-at-age data from previous years would be used to convert length composition to age-composition. If size-at-age of the stock does not vary over time, then use of previous years' data should not result in lower quality age composition information. The two controls used full age-composition data for both the survey and the catch with one and two year DMLs (see Table 3.1). Each lag reduction method was tested under scenarios with good and poor quality data, fast and slow life histories, and high and low recruitment variability in order to represent a broader range of fisheries.

The MSE included operating and assessment models (Figure 3.1). The operating model represented the true dynamics of the stock with an age-structured population model. Data representing catch and surveys were "sampled" from the operating model to represent the information typically used in the assessment process and varied based on the lag reduction method chosen. Subsequently, a statistical catch-at-age (SCAA) model was called at regular stock assessment intervals, varying from annual assessments to assessments every three years, to estimate biomass and age structure in the last year and fishing mortality and biomass reference points. The management portion of the model used the results of the stock assessment to determine target catch using the MAFMC harvest control rule, which includes a short-term projection to the year the catch limit will be implemented and uses a probabilistic approach (Shertzer et al. 2008; MAFMC 2011). We used three DML scenarios with differing data in the last year (Table 3.1). Each simulation was run for a total of 80 years. In the first 30 years the fishery developed with unregulated fishing. During the remaining 50 years the management strategy was in effect. At the end of each simulation, performance of the control rule was summarized over the 50-year management period. All models were developed in ADMB (Fournier et al. 2012). Variable definitions and equations are provided in Tables 3.2 and 3.3.

Operating model

The operating model simulated the population dynamics with an age-structured model. The model included 12 age classes for the fast life history, and 20 age classes for the slow life history , where 12+ and 20+ were aggregate age classes or "plus groups" of fish ages 12 or 20 and older. Abundance-at-age in the first year

was assumed to be in its unfished equilibrium state. We used a Beverton-Holt stock-recruitment relationship with an autocorrelated, lognormally distributed random error to determine recruitment each year (Eq. T2.1). The errors in the stock-recruitment relationship had a correlation of 0.44 and a log-scale standard deviation of 0.77 (the averages from a meta- analysis by Thorson et al. (2014. Abundance-at-age was calculated using an exponential mortality model with additive natural and fishing mortality (Eq. T2.2). The weight-at-age was calculated using length-at-age from a von Bertalanffy growth model (Eq. T2.3) and an allometric function of length-at-age (Eq. T2.4). Maturity-at-age followed a logistic function of age (Eq. T2.5). The spawning stock biomass (SSB) was the product of maturity-at-age, weight-at-age and abundance-at-age summed over ages for a given year (Eq. T2. 6).

The operating model included a single fishery, and selectivity followed a logistic function of age. Selectivity of the fishery and natural mortality were allowed to vary over time so that the assessment models would not exactly match the dynamics of the operating model. Fishery selectivity varied over time by applying an autocorrelated, lognormally distributed error with a standard deviation of 0.1 and autocorrelation of 0.3 or 0.9 to the sf50% parameter (Eq. T2.7 and T2.8), and natural mortality also followed an autocorrelated lognormal random process over time with a log-scale standard deviation of 0.15 and an autocorrelation of 0.3 (Eq. T2.9). Fishing mortality was set to 0.05 in the first year, after which it increased linearly until it plateaued in year 18 and remained constant until the management period began in year 30. The value of fishing mortality at the plateau depended on the exploitation history. Exploitation scenarios were light, moderate and heavy and used a fishing

mortality multiplier (F = 0.5, 1.0, 2.5 x $F_{MSY,}$ respectively) in the plateau year. Total mortality was the sum of the natural mortality and fishing mortality; fishing mortality at age was the product of the selectivity at age of the fishery and the overall fishing mortality rate for a year. Fishery catch-at-age was calculated using the Baranov catch equation (Eq. T2.10).

The operating model also generated catch-at-age in a survey as the product of abundance, survey selectivity, and survey catchability (Eq. T2.11). Survey catchability varied according to a random walk on the log scale with normally distributed errors (with a standard deviation of 0.01 or 0.05 depending on the data quality scenario) to allow gradual variation in the catchability over time (Eq. T2.12). The observed catch was calculated by multiplying total fishery catch by a lognormal error with a log-scale standard deviation set at 0.15 (Eq. T2.13). The observed index of abundance was calculated similarly, where the log-scale standard deviation of the random error was set to 0.3 or 0.7 depending on the data quality scenario (Eq. T2.14). The observed proportions at age for the fishery and survey were generated by sampling from a multinomial distribution using the true proportions at age (Eq. T2.15) and effective samples sizes (ESSs) of 50 or 200 depending on the data quality scenario for all years prior to the terminal year of the stock assessment. The ESSs in the terminal year depended on the lag reduction methods.

After each assessment the operating model implemented the MAFMC's P* control rule to estimate a target catch from the overfishing limit (OFL; MAFMC 2011). The OFL was estimated by applying the estimated $F_{35\%}$ from the assessment to the terminal estimate of abundance. $F_{35\%}$ is commonly used as a limit fishing

mortality reference point (Clark 2002). The MAFMC P* approach assumes that the OFL is lognormally distributed with a median value from the stock assessment projections and a CV of 100%. The target catch was calculated as the catch that achieves the 40^{th} percentile of the OFL distribution if estimated $B/B_{35\%}$ (derived from the SPR model in the assessment) exceeded 1.0. If estimated $B/B_{35\%}$ fell below 1.0 then the P* used to calculate ABC decreased linearly to zero until $B/B_{35\%} = 0.10$; below this value the target catch was set to 0 (MAFMC 2011; Figure 3.2). Once the target catch was determined, the operating model used a golden section search to find the fishing mortality resulting from achieving that catch. The target catch remained constant for the duration of the period between assessments.

Lag reduction methods

Age-structured stock assessment models, such as the SCAA implemented here, require two sources of data, the fishery catch at age and an additional source of auxiliary data, commonly a survey index of abundance (Fournier and Archibald 1982; Quinn and Deriso, 1999). The observations of age-composition from fishery dependent and survey data are often modeled using a multinomial distribution (Quinn and Deriso, 1999). With increased sample sizes the observation error in the proportions would decrease (i.e. less data available to represent strong age classes). Likewise, the accuracy of the method used to age the sample affects the effective sample size. If resources are not available to complete the aging of samples in the most recent year, other methods are needed to obtain the age composition. One such method is to use size-age data from previous years to generate an inverse age-length key, which is then used to estimate age composition in the most recent year using the

length data. If growth patterns are changing over time, using size-age data from prior years to estimate the most recent year's age composition will increase the amount of observation error. We approximated this process by using a lower effective sample size to generate age composition data in the most recent year using three approaches. Our study utilizes this idea by using a smaller effective sample size in order to simulate the error associated with using prior years' age-length data to age the samples in the most recent year.

Differences in the quality of the age-composition data in the last year of the assessment were included by modifying the ESS of the survey and catch age-composition for that year. Reduced ESSs were chosen to represent the error associated with the use of an inverse-age-length key in order to estimate age-composition from the length composition and data from previous years. For the two controls, the terminal year had the same ESS as the prior years, which were dependent on the data quality scenario. LR1 represented a situation in which no age composition data were used for the catch in the terminal year of the stock assessment, and all estimated of age-composition for that year were made from the prior years age composition. LR1 included the same ESS for the terminal year as for previous years for the survey age-composition, but did not include any age-composition data for the fishery catch in the terminal year. LR2 used the same ESS as the prior years for the survey proportions at age and an ESS of $1/4^{th}$ that of the prior years for fishery proportions at age. LR2 was chosen to represent a scenario where length data for the catch was collected in the terminal year of the stock assessment and used to estimate the catch age composition from prior years age and length data. The decrease in the

effective sample size for LR2 catch age composition can simulate a possible degradation in data when age compositions are inferred from length data. LR3 was chosen to represent a case where no age composition data (for both the survey and catch) in the terminal year of the stock assessment was used and only length data was collected to infer age from prior years age and length data. LR3 used an ESS of $1/4^{th}$ the prior years ESS' for both the survey and fishery proportions at age in the terminal year of the stock assessment.

C1 and C2 served as controls for our study. C1 represented a DML reduction of 1 year, or the goal of lag reduction methods, while C2 represented an additional year of DML or the "status quo" in most fisheries management systems. The goal of including the controls was to compare the lag reduction methods to them in order to determine the best available technique to reduce data lag.

Sample sizes of the original ESS, one half the original ESS and one fourth the original ESS showed slight differences (<2%) in performance metrics for both data quality scenarios. An ESS of one fourth the original size should represent a case with substantial sampling variability because the estimation of age-composition using inverse age-length keys should not degrade in performance if growth does not change over time.

Assessment model

We included two assessment models, which differed in whether age-composition data was available for the fishery in the terminal year. The assessment models used the data generated by the operating model with the first year of data collection beginning in year ten and the last year of data being the stock assessment

year minus the DML. The assessment models were SCAA models that estimated the abundance, fishing mortality, fishing mortality and biomass reference points, and the OFL. The structure of the SCAA models followed the same equations as the operating model, except that survey catchability, fishery selectivity, and natural mortality did not vary over time. The natural mortality in the assessment model was set to the mean true natural mortality of 0.2 for the fast life history and 0.1 for the slow life history. The negative log likelihood functions included lognormal distributions for the fishery and survey catch and multinomial distributions for the age-composition of the catch (Eq. T.2.16 and T.2.17). Each SCAA required the two data sets that were created in the operating model: the fishery catch-at-age and the survey index of abundance-at-age. The individual assessment models differed based on how they handled the survey and fishery age-composition and are described below. While LR2 and LR3 as well as C1 and C2 used a likelihood function that included all years for all data sources, LR1 did not include the age-composition of the catch in the last year. All of the models were provided the effective sample sizes that were used to generate the age-composition data for each year.

Parameters estimated by the SCAAs included recruitment parameters, fishery and survey selectivity parameters, abundance-at-age in the first year, fully selected fishing mortality for each year, and survey catchability. The SCAA models also used the true biological inputs from the operating model such as the maturity at age and size at age as well as the estimated selectivity at age to estimate the $F_{35\%}$ and $B_{35\%}$ reference points. Abundance in the final year of the assessment model was projected forward past the DML years in order to estimate the OFL for the year in which the

catch limit would be implemented by finding the catch that would achieve $F_{35\%}$ given the projected abundance-at-age. $B_{35\%}$ was calculated by multiplying the SPR from fishing at $F_{35\%}$ by the mean recruitment over the time series (Haltuch et al. 2008). The OFL and biological reference points were then returned to the operating model in order to apply the control rule and find the target catch.

Scenarios

Each lag reduction method was tested under annual, two year, and three year stock assessment intervals with a DML of one year and compared to the controls with a DML of one and two years. These combinations were tested under a factorial design of scenarios that considered alternative assumptions about data quality, stock-recruitment variability, exploitation history and life history. The simulations included two data quality types that differed in the amount of observation error in the survey index, the ESSs of the age-composition of the catch and survey, and the amount of variability in fishery selectivity, natural mortality, and survey catchability. The good data scenario used a coefficient of variation of 0.3 and 0.15 for the total survey and fishery catch, respectively, and an ESS of 200 for the proportions at age in the survey and fishery catch. For the poor data scenario a coefficient of variation of 0.7 and 0.15 for the survey and catch respectively and an effective sample size of 50 for both the survey index of abundance and fishery catch. Parameters of the operating model were chosen to represent species with a fast and a slow life history. Life histories were tailored to approximate summer flounder (*Paralichthys dentatus*) for the fast life history and spiny dogfish (*Squalus acanthias*) for the slow life history (parameters in Table 3.2). The fast life history of the summer flounder included early recruitment

into the fishery and early maturation, while the slow life history of the spiny dogfish represented low natural mortality and late recruitment and maturation. Due to preliminary model testing showing little difference between exploitation histories, the 2000 simulations were summarized across exploitation history with the first 667 runs representing an underfished stock, the second 667 runs representing a fully fished stock, and the final 666 runs representing an overfished fished stock. Exploitation scenarios were implemented by including a fishing mortality multiplier (F = 0.5, 1.0, 2.5 x F_{MSY} for the light, moderate, or heavy exploitation) in the pre-management years.

Performance metrics

The model tracked a range of performance metrics including the true catch, true biomass, probability of overfishing, average annual variability (AAV) of the catch, fishery closures and overfishing. The catch and biomass performance metrics took the average catch and biomass from the 50 year management period. The probability of overfishing metric was calculated as the proportion of years in which the true fishing mortality exceeded the fishing mortality limit during the 50 year management period. The AAV in catch was the average of the absolute value of the difference of catch from year to year across the 50 year management period. Fishery closures were calculated by taking the number of times fishing mortality in the model was set to zero for the fishing year and divided by the number of runs. Means, maximums, minimums, standard deviations and percent change from one year to the next and between lag reduction methods were taken to compare scenario outcomes. We estimated 95% confidence intervals for the medians of each performance metric

and included a Bonferonni correction to account for multiple comparisons between controls and lag reduction methods. A traditional method to calculate confidence intervals around the median was used because the median follows a normal distribution independent of the sample distribution (Samuels et al. 2012).

Results

Management that used approaches to reduce data lag by using partial data in the last year generally performed better than the approach that increased DML by using the most recent full year of data. Increases in assessment intervals increased the effects of DML with larger changes occurring between assessment intervals of one and two years and smaller changes between assessment intervals of two and three years across all performance metrics. LR1 and LR2 differed on average <2% from the results of C1 across all performance metrics and were effective in reducing the impacts of the two year DML. Life history also altered the effectiveness of lag reduction methods as the differences between the methods were smaller for the slow life history than the fast life history.

Catch for each lag reduction method was higher compared to C2 for all life history, data quality and assessment interval scenarios (Figure 3.3). The effects of the lag reduction methods were relatively small for the good data scenarios. Differences in median average catch were larger in the poor data scenario. Median average catch was 7% higher in C1 than C2. When compared to C2, the lag reduction methods achieved about a 7% increase in the median catch respectively for the scenario with fast life history and poor data. Increasing assessment intervals also increased the differences in average catch among controls and lag reduction method comparisons

slightly. For the data poor scenario with the slow life history, C1, LR1, LR2, and LR3 produced a median average catch that was about 4% higher than C2.

The average population biomass achieved by each of the lag reduction methods was similar to the results for catch (Figure 3.4). Overall, the lag reduction methods seemed to perform similar to C1. The effect of reducing DML on average biomass was greater for the fast life history and poor data scenario with a 9% decrease in the median biomass between C1 and C2. Lag reduction methods resulted in 6-8.3% higher median biomass than the control with a two-year DML. However, in the slow life history scenarios with poor data quality median average biomass was 6% lower in C1 than C2. Differences between C1 and LR1, LR2, and LR3 were less than 2% for the good data scenarios. Similarly, the lag reduction methods had performance that was similar to C1, 5-6% higher median average biomass compared to C2. Assessment interval effects for biomass were largest for the poor data scenarios with an average 3% increase with each additional year between assessments and saw the largest differences between C2 and the lag reduction methods for 3 year assessment intervals across all scenarios. The probability of fishery closure, when stock size (biomass) is ½ of the B/B_{proxy}, was much higher for both the data poor and slow life history scenarios, ranging from overfishing 2% of the time for the good data fast life history to 46% of the time for the poor data slow life history. All of the lag reduction methods had similar performance, <1% difference between their probabilities of fishery closure. However, two year data lag scenarios had a higher probability of fishery closure compared to C1 and the lag reduction methods.

Median probabilities of overfishing ranged from 29% to 43% across all methods and scenarios (Figure 3.5). The lag reduction methods generally resulted in a lower probability of overfishing than C2, except in the scenario with a fast life history and good data, which had very little difference among the methods. Results of the slow life history good data scenario were very similar to those for the fast life history. The mean probability of overfishing was about 18% higher for C1 than C2 for the poor data quality scenarios and 2% higher for the good data quality scenarios. Changes between C1 and LR1 and LR2 for the both the good and poor data quality scenarios were virtually zero, while LR3 decreased the probability of overfishing by around 4% when compared C1. The effectiveness of lag reduction methods were largest with the 3 year assessment intervals across all scenarios with the largest effects seen in the poor data quality scenarios.

Confidence intervals around the median catch AAVs indicated no significant differences in performance among the approaches for all of the scenarios (Figure 3.6). The main differences in catch AAV were that it was lower in the good data quality scenarios than the poor data quality scenarios and that catch AAV generally decreased with increasing assessment interval.

Discussion

Our results show that lag reduction methods can be successful in reducing the effects of DML to better achieve management goals. Overall, lag reduction methods had the largest effects when the data quality was relatively poor, and effects were small when data quality was high. Lag reduction methods that used age-composition information from the survey, but no or reduced information from the catch in the

terminal year of the stock assessment achieved performance that was similar to C1 with full data in the terminal year. Life history, data quality, and assessment interval all played important roles in the effects of DML and effectiveness of lag reduction methods, but lag reduction methods provided benefits over waiting in almost all cases.

Approaches that used age composition data for the terminal year of the survey, but no age composition for the catch (LR1) performed relatively well at meeting management goals and required the least amount of data. Even with missing age-composition of the catch in the terminal year of the assessment, performance metrics showed less than a 2% difference between a C1 and LR1. Ono et al. (2014) similarly did not see strong differences with the addition of fishery age composition data to stock assessment. While our study used SCAA models (and Ono et al. (2014) used length and age structured models), this technique may also be successfully applied to other stock assessment models.

Lag reduction methods that used full survey age composition data, but reduced quality age composition data for the catch in the terminal year (LR2) seemed to be the most successful at meeting management goals of the three lag reduction methods. LR2 differed, on average, <1% from the results of the annual DML (C1) across all performance metrics and was effective in offsetting the impacts of the two year DMLs. This method is currently used successfully with some stocks managed by the North Pacific Fishery Management Council in order to reach a one year data lag goal (AFSC 2014). LR2 was most successful because it had the best age composition data. Having a reduced effective sample size of the age composition in

the last year should not have a large effect on estimation model performance if information from the history of the fishery is available (Ono et al. 2014). Method LR2 was, however, the most data heavy of the three methods. While length-at-age data should be relatively fast to collect, additional data in the model may result in longer data collection preparation time and may still result in longer lags with comparatively small gains compared to LR1.

The final method evaluated considered reduced quality data for both the survey and catch age composition in the terminal year of the assessment (LR3). LR3 produced average catch, biomass, and probability of overfishing similar to the other lag reduction methods. While this method saw slight differences compared to methods LR1 and LR2 it still performed relatively well when compared to a two year data lag scenario (C2). However, another lag reduction method we tested that used no catch or survey age-composition data in the terminal year of the assessment was rarely able to estimate recruitment in the last year and was excluded from the study, thus highlighting the limits of partial data within stock assessments.

Practical implementation of our results relies on the ability to collect informative age and length composition data, especially in years prior to the terminal year. The control which used one year of data lag (C1) was used to represent an idealized case in which DML can be reduced to one year by either aging all the samples for that year or having a situation where growth does not change over time and age-length data from earlier years can be used to generate the age compositions in the terminal year. Changes in fish growth overtime have been observed for many fish

stocks (Thorson and Minte-Vera, 2014), but growth has remained relatively constant in many others (Hilborn and Minte-Vera, 2008; Thorson and Minte-Vera, 2014).

Our control with 2 years of data lag or the "status quo" (C2) approach represents how age composition data are used in many U.S. fisheries management systems, where years with missing data are excluded from the stock assessment, thus adding to DML. LR1 and LR2 represents fisheries with differing amounts of error between the data used in prior years and those used in the terminal year of the catch composition. Results of LR1 and LR2 emphasize that even if temporal trends in growth are not substantial, assessments that use previous years' age-length data may still perform well as long as some good age composition data is available. Our reductions in effective sample size in the last year are probably larger than would be expected if inverse age-length keys were used to generate age compositions for the most recent year. In fact, if growth has not changed over time, using all the age-length data to estimate expected size at age would likely outperform using individual year's data because the sampling error would be reduced.

Our LR3 approach performed slightly poorer than LR1 and LR2, but was still more successful than C2. In many fisheries management systems assessment scientists do not include terminal years with missing data in the stock assessment, under the assumption that using partial data in terminal years of the stock assessment will decrease model performance. Cases such as these are comparable to our two year DML control. Our findings did not support this assumption; when only survey age composition data were included in the terminal year of the stock assessment (LR3), it still performed better than a two year data lag (CR2). The choice ultimately comes

down to using some data in the terminal year of the stock assessment (lag reduction methods), or using no data in the terminal year of the stock assessment (CR2).

While LR1 and LR2 had similar performance to C1, there are still additional potential cases where lag reduction methods may not succeed. Complications with missing age-composition data in the assessment such as failing to notice changing trends in recruitment and failure to forecast future conditions can be a likely result of these techniques (Mace et al. 2001). One strong assumption of the lag reduction methods is that there were no trends in growth over time. We would expect a reduction in success of these methods if growth changed over time as growth rates must be explicitly estimated in age-structured population models (Quinn and Deriso, 1999). Ono et al. (2014) tested the quality and quantity of length and age-composition data in three species and found that the usefulness of age-composition data decreased as the variation of the relationship between length and age increased. Species or data sets with high variation in length-at-age may be less successful in the use of lag reduction methods.

Our research supports the conclusions of previous work that DML and assessment intervals can have a relatively large effect on fisheries management outcomes (Chapter 2). The most effective way to decrease DML effects may be to decrease the fisheries management timeline (Shertzer and Prager 2007; Brown et al. 2010). We agree that shortening the management process would be an effective means to mitigate problems associate with DML without decreasing the amount of data used in stock assessments (Sylvia, chapter 2). Because changing the management system can be a lengthy process, we recommend adopting assessment procedures that

attempt to fully use the most recent available data instead of waiting on more data (Shertzer and Prager 2007). Having no catch age-composition, or using length data to infer age-composition in the terminal year of the assessment are both successful techniques in reducing DML and should be considered as such in future fisheries management plans.

Tables and Figures

Table 3.1. Design and details of lag reduction methods. Lag reduction method 1 is a control with annual data lag. Lag reduction method 2 is a control with two year data lag.

Lag reduction method	Full age-composition in terminal year	ESS_{survey}(terminal year)	ESS_{catch}(terminal year)	Data lag
C1	Yes	200, 50*	200, 50*	1
LR1	No	200, 50*	0	1
LR2	No	200, 50*	50, 12*	1
LR3	No	50, 12*	50, 12*	1
C2	Yes	200, 50*	200, 50*	2

Table 3.2. Equations governing the population and data-generating dynamics in the operating model.

	Equation	Description
1	$$R_t = \frac{S_{t-a_R}}{1-\left(\frac{5h-1}{4h}\right)+\left(1-\frac{S_{t-a_R}}{S_{eq}}\right)} e^{\theta_R - 0.5\sigma_R^2}$$	Stock-recruit relationship
2	$$N(a,t) = \begin{cases} R(t) & a = a_R \\ N(a-1,t-1)e^{-Z(a-1,t-1)} & a_R < a < a_{max} \\ N(a-1,t-1)e^{-Z(a-1,t-1)} + & a = a_{max} \\ N(a,t-1)e^{-Z(a,t-1)} & \end{cases}$$	Abundance-at-age
3	$$L_a = L_\infty(1-e^{-k(a-a_0)})$$	Length-at-age
4	$$w_a = bL_a^{\ c}$$	Weight-at-length
5	$$m_a = \frac{1}{1+e^{-\left(\frac{a-m_{50\%}}{m_{slope}}\right)}}$$	Maturity-at-age
6	$$S_t = \sum_{a=a_R}^{a_{\max}} m_a w_a N_{t,a}$$	Spawning biomass
7	$$S_{t,a} = \frac{1}{1+e^{-\left(\frac{a-S_{50\%,t}}{S_{slope}}\right)}}$$	Selectivity-at-age in the fishery
8	$$s_{50\%_t} = \bar{s}_{50\%} e^{\varepsilon_{s_t} - 0.5\sigma_s^2}$$ $$\varepsilon_{st} = \rho_s \varepsilon_{s_{t-1}} + \sqrt{1-\rho_s^2}\phi_t$$ $$\phi_t \sim N(0,\sigma_s^2)$$	Time-varying and autocorrelated selectivity
9	$$M_t = Me^{\varepsilon_{M_t} - 0.5\sigma_M^2}$$	Time-varying and autocorrelated natural mortality

$$\varepsilon_{M,t} = \rho_M \varepsilon_{M_{t-1}} + \sqrt{1-\rho_M^2}\phi_{M,t}$$

$$\phi_{M,t} \sim N(0, \sigma_M^2)$$

10 $$C_{t,a} = \frac{F_{t,a}}{Z_{t,a}}(1-e^{-Z_{t,a}})N_{t,a}W_{t,a}$$ Baranov-catch Eq.

11 $$I_{t,a} = q_t s_{s,a} N_{t,a}$$ True index of abundance

12 $$q_t = q_{t-1}e^{\varepsilon_{qt}}$$ Catchability

$$\varepsilon_{q_t} \sim N(0, \sigma_q^2)$$

13 $$C_{obs,t} = C_t e^{\varepsilon_{C_t} - 0.5\sigma_C^2}$$ Observed catch

$$\varepsilon_{C_t} \sim N(0, \sigma_C^2)$$

14 $$I_{obs_t} = e^{\varepsilon_{I_t} - 0.5\sigma_I^2} \sum_a I_{a,t}$$ Observed index of abundance

$$\varepsilon_{I_t} \sim N(0, \sigma_I^2)$$

15 $$P_{obs_t} = \frac{1}{n}\boldsymbol{\Theta_t}$$
$$\boldsymbol{\Theta_t} \sim Multinomial(n, \mathbf{p_t})$$
$$\mathbf{p_t} = \frac{1}{I_t}\left(I_{a_r,t}, \ldots, I_{a_{max},t}\right)$$ Proportion of catch-at-age

16 $$\ell_1 = 0.5n\log(\sigma_C^2) + \sum_t \left(\log(C_{obs_t}) - \log(C_{est_t})\right)^2$$ Lognormal likelihood components for the catch and index of abundance

$$\ell_t = 0.5n\log(\sigma_I^2) + \sum_t \left(\log(I_{obs_t}) - \log(I_{est_t})\right)^2$$

17 $$\ell_3 = -E_C \sum_t \sum_a p_{obs,C_{t,a}} \log(p_{est,C_{t,a}})$$

Multinomial likelihood components for the proportion of the catch and index

$$\ell_4 = -E_I \sum_t \sum_a p_{obs,I_{t,a}} \log(p_{est,I_{t,a}})$$

Table 3.3. Symbols, specified life history and time-varying parameters of model.

Parameter	Description	Life history: Slow	Life history: Fast
a_r	Age at recruitment	5	3
a_{max}	Aggregate age class	20	12
M	Natural mortality rate	0.1	0.2
R_0	Unfished equilibrium recruitment	$1x10^6$	$1x10^6$
h	Steepness	0.6	0.75
a_0	Age at length = 0	0	0
L_∞	Asymptotic maximum length	90	90
K	Growth rate	0.07	0.13
B	Length-weight relationship scalar	3.5×10^{-6}	3.5×10^{-6}
C	Length-weight relationship exponent	3	3
m_{50}	Age at 50% maturity	7	3.5
m_{slope}	Slope of maturity function	1	1
s_{f50}	Mean age at 50% selectivity in the fishery	7	3.5
s_{s50}	Mean age at 50% selectivity in the survey	5.3	2.6
s_{fslope}	Slope of fishery selectivity function	1	1
s_{sslope}	Slope of survey selectivity function	1	1
	Time Varying parameters		
σ_R	Standard deviation of stock-recruit relationship	0.77, 1.25	
Φ_R	Autocorrelation in recruitment	0.44	
σ_M	Standard deviation of time-varying M	0.15	
Φ_M	Autocorrelation in M	0.3, 0.9	
σ_f	Standard deviation of age at 50% selectivity in fishery	0.1	
Φ_f	Autocorrelation in fishery selectivity	0.3, 0.9	
σ_C	Standard deviation of catch estimates	0.15	
σ_I	Standard deviation of survey estimates	0.29, 0.63	
	Additional model variables		
A	Age		
a_R	Age at recruitment		
T	Year		

R	Recruitment
S	Spawning biomass
Sf	Fishery selectivity
Ss	Survey selectivity
N	Abundance
M	Maturity
C	Catch
I	Index of abundance
Q	Catchability
Z	Total mortality
W	Weight-at-age
M	Maturity-at-age
F	Fishing mortality
E	Effective sample size of the catch/ index
N	Number of observations
$Pobs$	Observed proportion at age
$Pest$	Estimated proportions at age
$\boldsymbol{\Theta_t}$	Multinomial function
ℓ_t	Likelihood function
ε_{s_t}	Error for selectivity
ε_{q_t}	Error for catchability
ε_M	Error for natural mortality
ε_{I_t}	Error for index of abundance
ε_{C_t}	Error for catch
σ_q	Standard deviation for catchability
ρ_M	Correlation coefficient of natural mortality
ρ_s	Correlation coefficient of selectivity

Table 3.4. 95 % confidence intervals for each life history and data scenarios, combinations are paired by stock assessment interval and lag reduction methods. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. Confidence intervals were calculated around the median with a Bonferonni correction to account for multiple comparisons.

Fast life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	149872, 152671	1057152, 1086097	0.301, 0.326	0.102, 0.105
1, LRM1	149382, 152191	1056443, 1085771	0.301, 0.326	0.104, 0.106
1, LRM2	150113, 152915	1059006, 1088053	0.282, 0.307	0.103, 0.105
1, LRM3	150001, 152805	1059458, 1088606	0.292, 0.316	0.103, 0.105
1, C2	147139, 150415	1054232, 1089738	0.300, 0.327	0.098, 0.101
2, C1	148788, 151683	1051867, 1081402	0.302, 0.326	0.082, 0.085
2, LRM1	148474, 151409	1050691, 1080748	0.301, 0.326	0.083, 0.086
2, LRM2	148846, 151744	1066017, 1095678	0.302, 0.326	0.084, 0.087
2, LRM3	148618, 151524	1067035, 1097099	0.282, 0.306	0.087, 0.090
2, C2	145745, 149426	1046385, 1084399	0.300, 0.328	0.079, 0.082
3, C1	148292, 151297	1045451, 1075808	0.321, 0.345	0.075, 0.077
3, LRM1	148503, 151563	1042991, 1074173	0.321, 0.345	0.075, 0.078
3, LRM2	148748, 151809	1050940, 1081709	0.302, 0.326	0.076, 0.079
3, LRM3	148236, 151318	1057048, 1088516	0.302, 0.326	0.078, 0.081
3, C2	145218, 149685	1030411, 1073458	0.319, 0.348	0.070, 0.074
Slow life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	1787503, 1849926	151451, 155410	0.299, 0.328	0.082, 0.092
1, LRM1	1785591, 1848433	151242, 155228	0.299, 0.328	0.083, 0.093
1, LRM2	1792442, 1854772	151183, 155131	0.299, 0.328	0.083, 0.093
1, LRM3	1790125, 1852504	151262, 155214	0.299, 0.328	0.083, 0.093
1, C2	1756096, 1824853	148449, 152670	0.318, 0.348	0.082, 0.094
2, C1	1753202, 1816382	148569, 152674	0.319, 0.348	0.064, 0.074
2, LRM1	1752144, 1815985	148388, 152526	0.319, 0.348	0.064, 0.074
2, LRM2	1768102, 1832042	148610, 152723	0.299, 0.328	0.065, 0.075
2, LRM3	1757832, 1822072	148205, 152300	0.300, 0.328	0.067, 0.077
2, C2	1727088, 1797711	146169, 150612	0.318, 0.348	0.064, 0.075
3, C1	1723293, 1787656	145426, 149629	0.319, 0.348	0.057, 0.066
3, LRM1	1720562, 1785702	145299, 149547	0.319, 0.348	0.056, 0.066
3, LRM2	1725256, 1790033	145614, 149820	0.319, 0.348	0.058, 0.068

3, LRM3	1739827, 1806607	147303, 151593	0.319, 0.348	0.060, 0.070
3, C2	1679967, 1753858	143856, 148615	0.338, 0.368	0.055, 0.067
Fast life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	127568, 131423	1029996, 1090433	0.3184, 0.3483	0.1888, 0.1973
1, LRM1	126889, 130973	1017022, 1080643	0.3181, 0.3486	0.1923, 0.2010
1, LRM2	127760, 131642	1036910, 1099054	0.3183, 0.3483	0.1887, 0.1971
1, LRM3	127514, 131390	1036611, 1098783	0.3183, 0.3483	0.1892, 0.1977
1, C2	118399, 124359	957009, 1034658	0.3532, 0.3919	0.1790, 0.1928
2, C1	125743, 129983	1007044, 1070811	0.3183, 0.3484	0.1458, 0.1547
2, LRM1	123538, 127984	1003869, 1069285	0.3179, 0.3487	0.1483, 0.1575
2, LRM2	124930, 129039	1035461, 1098968	0.3188, 0.3478	0.1534, 0.1621
2, LRM3	124453, 128671	1044723, 1109476	0.2997, 0.3277	0.1616, 0.1704
2, C2	114623, 120969	930327, 1010505	0.3821, 0.4218	0.1399, 0.1545
3, C1	996838, 1062011	121976, 126405	0.3380, 0.3679	0.1303, 0.1392
3, LRM1	979703, 1046956	121563, 126278	0.3375, 0.3683	0.1320, 0.1414
3, LRM2	1009667, 1074987	122951, 127428	0.3383, 0.3675	0.1343, 0.1435
3, LRM3	1019362, 1085092	122410, 126863	0.3191, 0.3476	0.1385, 0.1474
3, C2	907267, 990009	110857, 117578	0.4108, 0.4520	0.1210, 0.1368
Slow life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	124306, 129342	1678362, 1798767	0.317, 0.350	0.161, 0.185
1, LRM1	123328, 128477	1658306, 1780558	0.317, 0.350	0.160, 0.185
1, LRM2	124430, 129475	1674319, 1795330	0.317, 0.350	0.160, 0.184
1, LRM3	124762, 129820	1662994, 1784370	0.317, 0.350	0.161, 0.185
1, C2	118757, 124507	1581803, 1714426	0.335, 0.371	0.160, 0.187
2, C1	121704, 127077	1610497, 1732942	0.336, 0.369	0.120, 0.144
2, LRM1	120930, 126427	1591832, 1716037	0.336, 0.370	0.119, 0.144
2, LRM2	121339, 126771	1618910, 1742684	0.317, 0.350	0.123, 0.148
2, LRM3	121159, 126586	1641798, 1766686	0.317, 0.349	0.128, 0.153
2, C2	115690, 121751	1533127, 1668607	0.354, 0.391	0.120, 0.148
3, C1	119846, 125470	1596476, 1721388	0.337, 0.369	0.103, 0.128
3, LRM1	119868, 125653	1579166, 1706223	0.336, 0.370	0.103, 0.129
3, LRM2	119310, 124991	1581327, 1707877	0.337, 0.369	0.105, 0.131
3, LRM3	119296, 125003	1587388, 1714706	0.337, 0.369	0.109, 0.134
3, C2	113999, 120494	1452285, 1591834	0.374, 0.411	0.100, 0.129

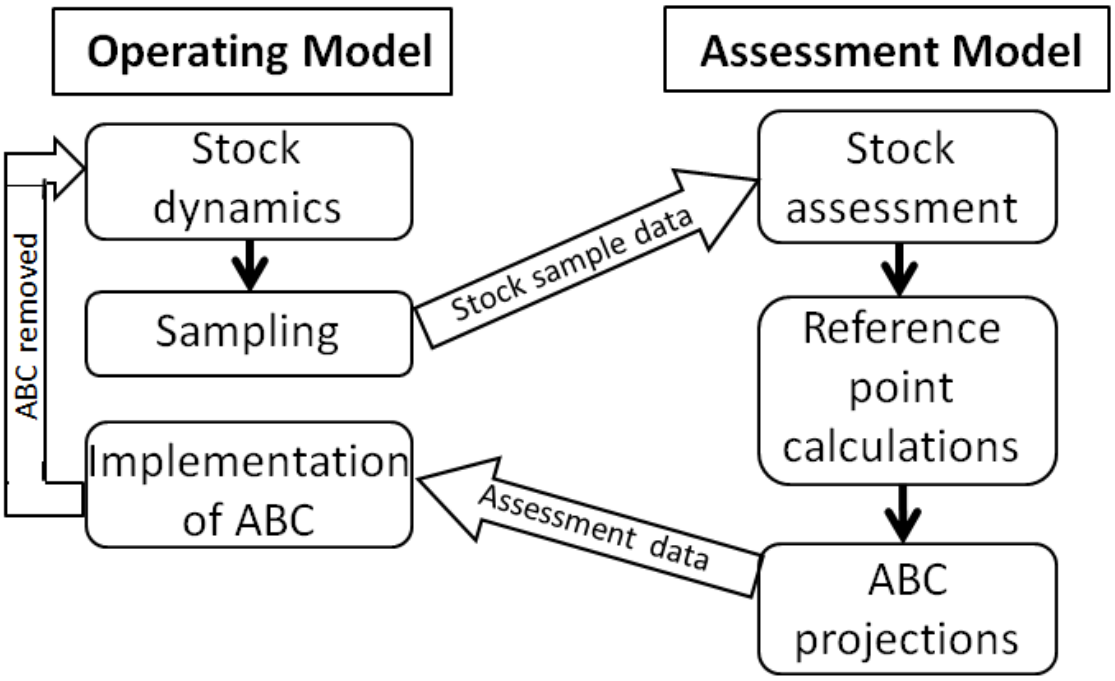

Figure 3.1. Flow diagram of management strategy evaluation model with both operating and estimation models.

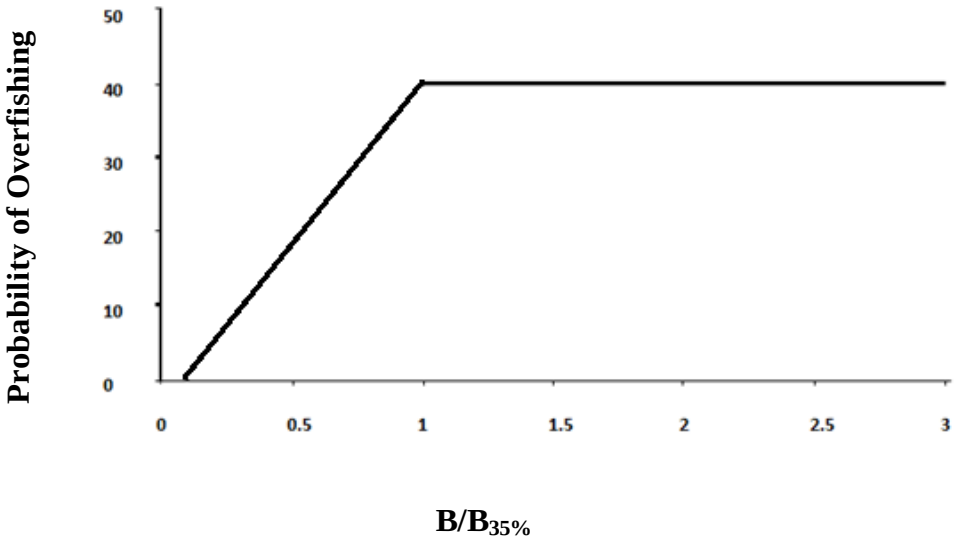

Figure 3.2. Mid-Atlantic P* approach showing decreasing probability of overfishing with a declining $B/B_{35\%}$ ratio

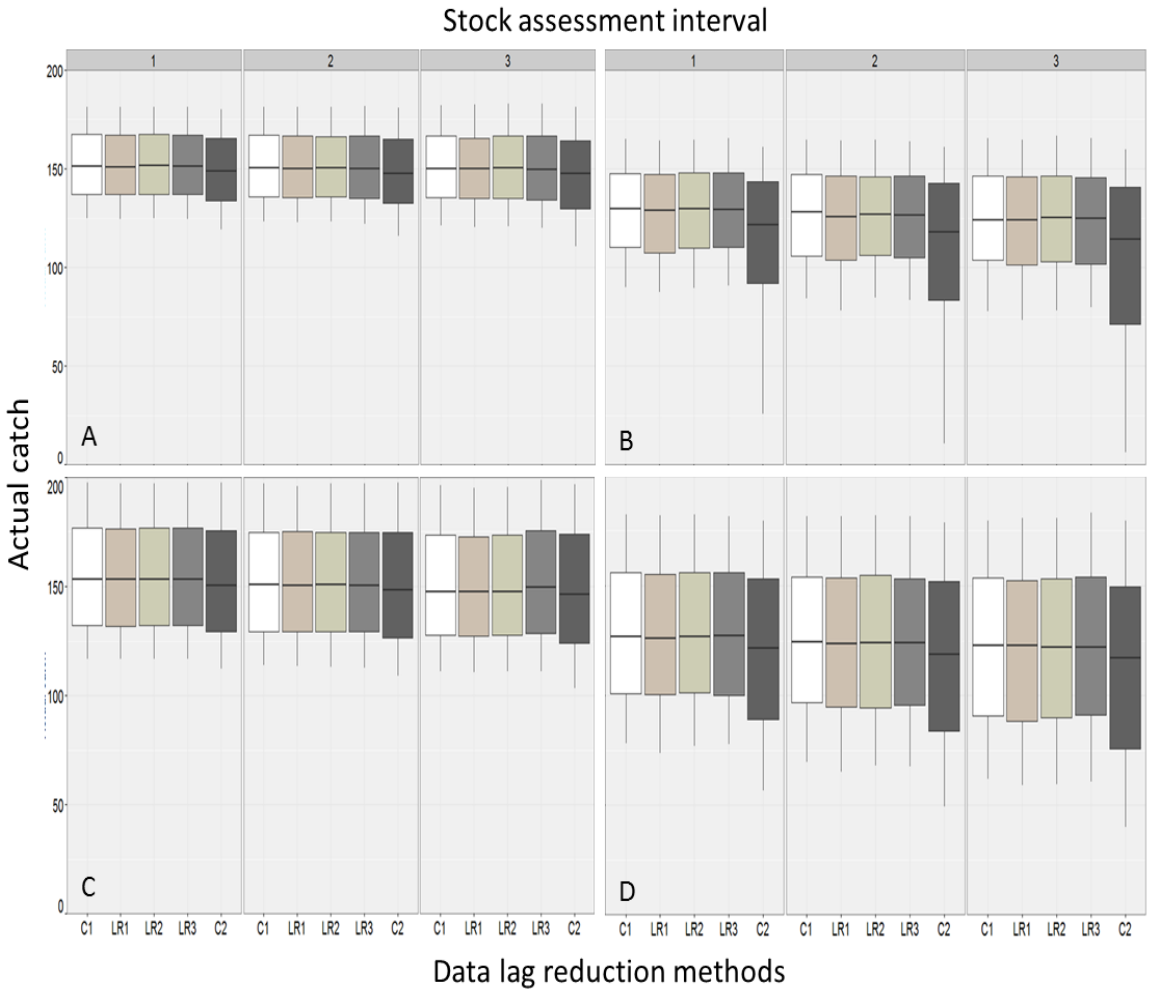

Figure 3.3. Box plots of the catch for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles.

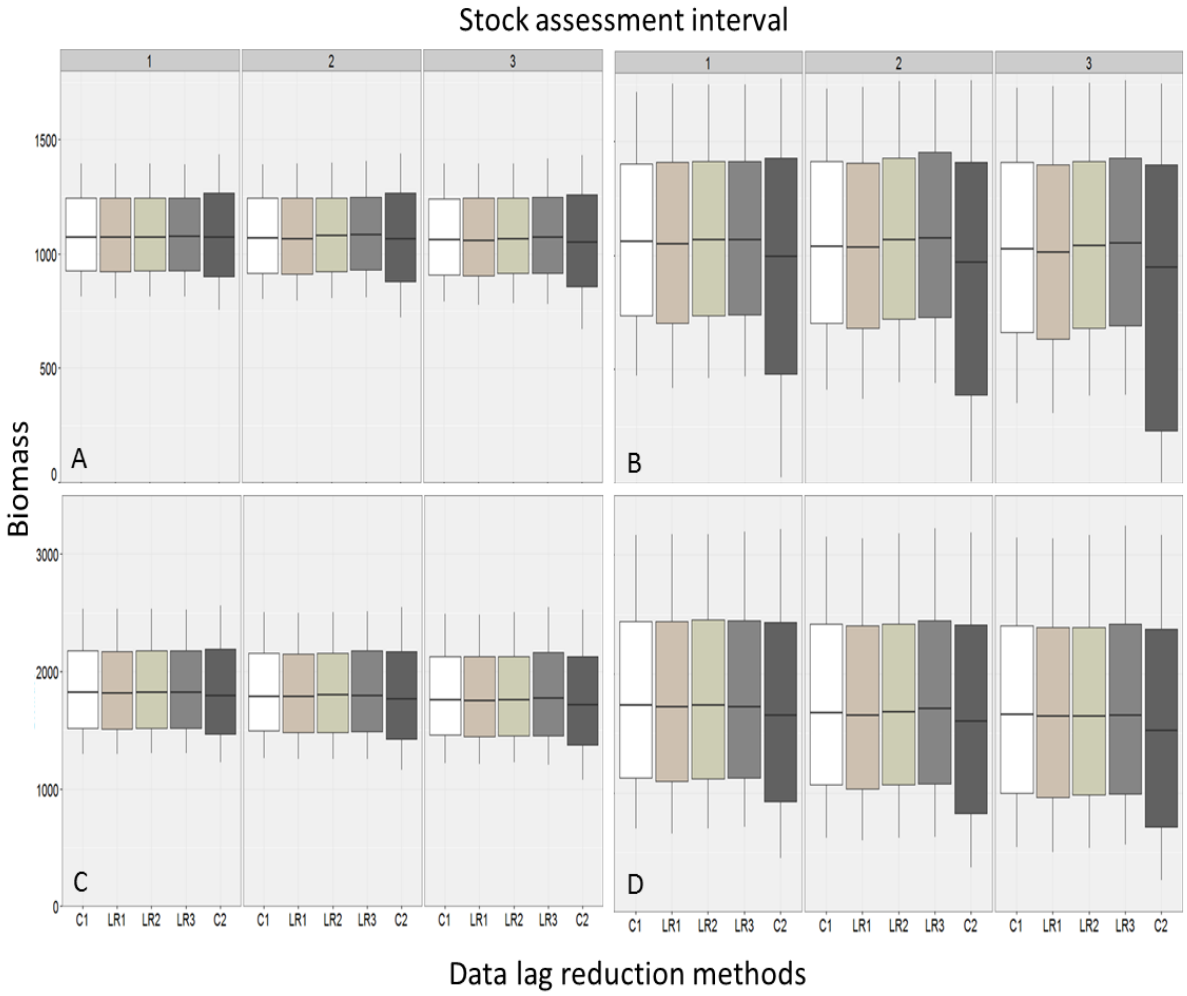

Figure 3.4. Box plots of the biomass for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles.

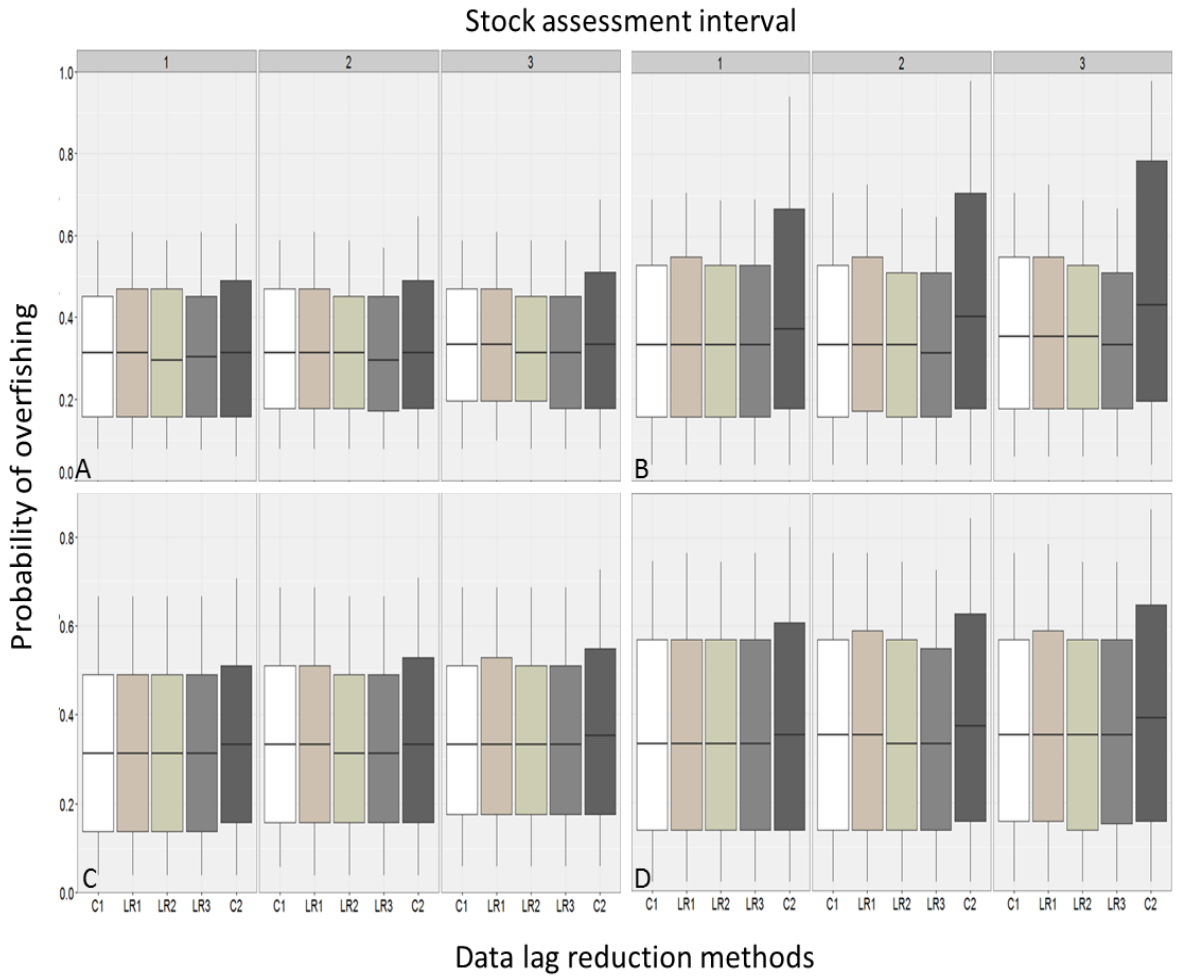

Figure 3.5. Box plots of the probability of overfishing for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LR1 is lag reduction method 1, LR2 is lag reduction method 2 and LR3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles.

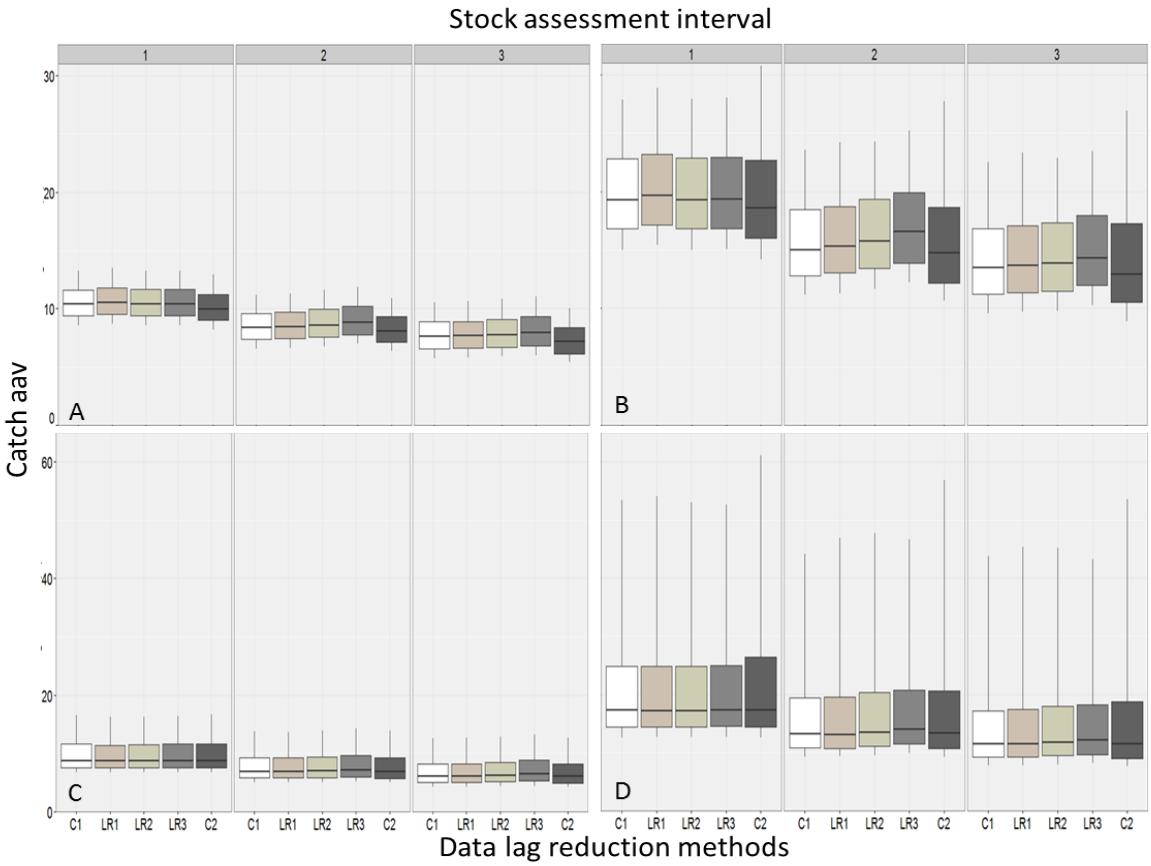

Figure 3.6. Box plots of the AAV of catch for each life history and data scenarios. Scenario A is the fast life history with good data, scenario B is the fast life history with poor data. Scenario C is the slow life history with good data, and scenario D is the slow life history with poor data. C1 represents control 1 of the lag reduction methods with an annual data lag, LR1 is lag reduction method 1, LR2 is lag reduction method 2 and LR3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. The horizontal lines of the box plot show the median biomass and the whiskers represent the 10^{th} and 90^{th} percentiles.

Appendices

Appendix I: Extended results of Chapter 2: Effects of assessment interval and data-management lag on Mid-Atlantic harvest control rule performance. Included medians, confidence intervals and additional figures.

Table I.1 Chapter 1 median values for low recruitment variability scenarios across performance metrics. SA refers to stock assessment intervals and DLM refers to data-management lag combinations

Fast life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	155126	1124275	0.255	0.113
SA1, DML2	153642	1142520	0.255	0.107
SA1, DML3	151585	1160225	0.255	0.100
SA2, DML1	154624	1121440	0.275	0.091
SA2, DML2	152968	1142380	0.255	0.087
SA2, DML3	150250	1152175	0.255	0.079
SA3, DML1	153935	1122225	0.284	0.083
SA3, DML2	153447	1134070	0.275	0.078
SA3, DML3	149297	1156165	0.275	0.071
SA5, DML1	151911	1110390	0.294	0.073
SA5, DML2	150308	1114175	0.294	0.066
SA5, DML3	145826	1132315	0.294	0.058
SA7, DML1	151703	1100225	0.314	0.074
SA7, DML2	146678	1104580	0.314	0.065
SA7, DML3	143940	1102770	0.314	0.059
SA10, DML1	149489	1076710	0.333	0.068
SA10, DML2	145718	1082010	0.353	0.057
SA10, DML3	138473	1053975	0.373	0.049
Slow life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	152348	1904750	0.255	0.097
SA1, DML2	150784	1884120	0.275	0.096
SA1, DML3	149084	1873625	0.275	0.095
SA2, DML1	150199	1889940	0.275	0.077
SA2, DML2	149371	1866815	0.275	0.076
SA2, DML3	146904	1856525	0.275	0.075
SA3, DML1	148417	1858690	0.275	0.069

SA3, DML2	147785	1845335	0.275	0.067
SA3, DML3	144886	1832375	0.294	0.066
SA5, DML1	149205	1820120	0.294	0.058
SA5, DML2	147046	1803120	0.294	0.056
SA5, DML3	146130	1767120	0.294	0.055
SA7, DML1	143133	1766610	0.294	0.058
SA7, DML2	140721	1733845	0.314	0.055
SA7, DML3	138494	1703670	0.333	0.054
SA10, DML1	137008	1697950	0.314	0.050
SA10, DML2	136382	1662820	0.353	0.048
SA10, DML3	133663	1597655	0.392	0.046
Fast life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	134442	1127795	0.294	0.208
SA1, DML2	129124	1110195	0.294	0.197
SA1, DML3	123173	1082590	0.304	0.185
SA2, DML1	131778	1113425	0.294	0.165
SA2, DML2	126151	1112400	0.304	0.155
SA2, DML3	121694	1072750	0.314	0.145
SA3, DML1	129738	1102525	0.314	0.145
SA3, DML2	125227	1088190	0.294	0.137
SA3, DML3	119344	1034780	0.314	0.129
SA5, DML1	126474	1091260	0.314	0.123
SA5, DML2	120737	1058420	0.314	0.116
SA5, DML3	116396	1024570	0.314	0.108
SA7, DML1	122340	1063725	0.314	0.124
SA7, DML2	114935	1031820	0.314	0.115
SA7, DML3	110555	987028	0.333	0.105
SA10, DML1	113785	1008195	0.333	0.109
SA10, DML2	110568	998942	0.353	0.100
SA10, DML3	104552	966664	0.373	0.090
Slow life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	128444	1838565	0.294	0.188
SA1, DML2	122546	1766290	0.294	0.188
SA1, DML3	116719	1678380	0.294	0.189
SA2, DML1	124826	1801545	0.294	0.145
SA2, DML2	117921	1733355	0.294	0.143
SA2, DML3	113613	1631610	0.314	0.144
SA3, DML1	122622	1789210	0.294	0.126
SA3, DML2	116180	1701720	0.294	0.125

SA3, DML3	110796	1599740	0.294	0.125
SA5, DML1	123228	1702095	0.294	0.107
SA5, DML2	116850	1637950	0.294	0.106
SA5, DML3	113229	1490400	0.314	0.105
SA7, DML1	112855	1578035	0.294	0.100
SA7, DML2	106183	1497825	0.294	0.099
SA7, DML3	104152	1385935	0.294	0.101
SA10, DML1	109158	1450320	0.314	0.089
SA10, DML2	101322	1400445	0.314	0.088
SA10, DML3	94954	1241100	0.353	0.086

Table I.2: Chapter 1 95 % confidence intervals for each life history and DML scenarios for the low recruitment variability scenario. Combinations are paired by stock assessment interval and DML. SA is stock assessment interval; DML is data-management lag. Confidence intervals were calculated around the median with a Bonferonni correction to account for multiple comparisons.

Fast life history, Good data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	153779.0, 156473.0	1111105.5, 1137444.5	0.244, 0.266	0.111, 0.114
SA1, DML2	152256.2, 155026.8	1128041.5, 1156998.5	0.244, 0.266	0.106, 0.109
SA1, DML3	150140.2, 153029.8	1143954.1, 1176495.9	0.244, 0.266	0.099, 0.102
SA2, DML1	153232.7, 156014.3	1108220.5, 1134659.5	0.264, 0.285	0.090, 0.093
SA2, DML2	151532.4, 154403.6	1127631.6, 1157128.4	0.244, 0.265	0.085, 0.088
SA2, DML3	148736.8, 151762.2	1135556.0, 1168794.0	0.244, 0.266	0.078, 0.081
SA3, DML1	152489.9, 155380.1	1108728.3, 1135721.7	0.274, 0.295	0.082, 0.085
SA3, DML2	151949.8, 154944.2	1119011.1, 1149128.9	0.264, 0.285	0.076, 0.079
SA3, DML3	147695.1, 150897.9	1138790.6, 1173539.4	0.264, 0.285	0.069, 0.073
SA5, DML1	150353.3, 153468.7	1096401.7, 1124378.3	0.284, 0.304	0.071, 0.075
SA5, DML2	148661.9, 151954.1	1097979.0, 1130371.0	0.284, 0.304	0.065, 0.068
SA5, DML3	143887.2, 147763.8	1112692.8, 1151937.2	0.284, 0.304	0.057, 0.060
SA7, DML1	149915.5, 153490.5	1085950.9, 1114499.1	0.304, 0.323	0.073, 0.076
SA7, DML2	144664.2, 148690.8	1086825.0, 1122335.0	0.304, 0.324	0.064, 0.067
SA7, DML3	141517.9, 146361.1	1080521.4, 1125018.6	0.303, 0.324	0.057, 0.061
SA10, DML1	147253.8, 151723.2	1060451.6, 1092968.4	0.324, 0.342	0.066, 0.070
SA10, DML2	143079.3, 148356.7	1061377.8, 1102642.2	0.343, 0.363	0.055, 0.059
SA10, DML3	135271.9, 141674.1	1026259.0, 1081691.0	0.362, 0.384	0.046, 0.051
Slow life history, Good data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	153779.0, 156473.0	1111105.5, 1137444.5	0.244, 0.266	0.111, 0.114
SA1, DML2	152256.2, 155026.8	1128041.5, 1156998.5	0.244, 0.266	0.106, 0.109
SA1, DML3	150140.2, 153029.8	1143954.1, 1176495.9	0.244, 0.266	0.099, 0.102
SA2, DML1	153232.7, 156014.3	1108220.5, 1134659.5	0.264, 0.285	0.090, 0.093
SA2, DML2	151532.4, 154403.6	1127631.6, 1157128.4	0.244, 0.265	0.085, 0.088
SA2, DML3	148736.8, 151762.2	1135556.0, 1168794.0	0.244, 0.266	0.078, 0.081
SA3, DML1	152489.9, 155380.1	1108728.3, 1135721.7	0.274, 0.295	0.082, 0.085
SA3, DML2	151949.8, 154944.2	1119011.1, 1149128.9	0.264, 0.285	0.076, 0.079
SA3, DML3	147695.1, 150897.9	1138790.6, 1173539.4	0.264, 0.285	0.069, 0.073
SA5, DML1	150353.3, 153468.7	1096401.7, 1124378.3	0.284, 0.304	0.071, 0.075
SA5, DML2	148661.9, 151954.1	1097979.0, 1130371.0	0.284, 0.304	0.065, 0.068
SA5, DML3	143887.2, 147763.8	1112692.8, 1151937.2	0.284, 0.304	0.057, 0.060

SA7, DML1	149915.5, 153490.5	1085950.9, 1114499.1	0.304, 0.323	0.073, 0.076
SA7, DML2	144664.2, 148690.8	1086825.0, 1122335.0	0.304, 0.324	0.064, 0.067
SA7, DML3	141517.9, 146361.1	1080521.4, 1125018.6	0.303, 0.324	0.057, 0.061
SA10, DML1	147253.8, 151723.2	1060451.6, 1092968.4	0.324, 0.342	0.066, 0.070
SA10, DML2	143079.3, 148356.7	1061377.8, 1102642.2	0.343, 0.363	0.055, 0.059
SA10, DML3	135271.9, 141674.1	1026259.0, 1081691.0	0.362, 0.384	0.046, 0.051
Fast life history, Poor data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	132753.3, 136130.7	1099949.4, 1155640.6	0.282, 0.307	0.204, 0.212
SA1, DML2	127265.3, 130982.7	1079549.3, 1140840.7	0.282, 0.307	0.193, 0.202
SA1, DML3	121155.4, 125190.6	1049911.9, 1115268.1	0.291, 0.317	0.180, 0.189
SA2, DML1	129972.4, 133582.6	1085146.9, 1141703.1	0.282, 0.306	0.160, 0.169
SA2, DML2	124204.8, 128096.2	1081505.2, 1143294.8	0.292, 0.316	0.151, 0.159
SA2, DML3	119572.1, 123815.9	1039970.5, 1105529.5	0.301, 0.326	0.140, 0.150
SA3, DML1	127879.0, 131597.0	1073693.7, 1131356.3	0.302, 0.326	0.140, 0.149
SA3, DML2	123181.1, 127271.9	1057084.0, 1119296.0	0.282, 0.306	0.133, 0.142
SA3, DML3	117131.5, 121556.5	1001495.9, 1068064.1	0.302, 0.326	0.124, 0.135
SA5, DML1	124400.1, 128547.9	1062058.1, 1120461.9	0.302, 0.326	0.119, 0.128
SA5, DML2	118512.2, 122961.8	1026847.4, 1089992.6	0.302, 0.326	0.111, 0.120
SA5, DML3	113994.7, 118796.3	990333.9, 1058806.1	0.302, 0.325	0.103, 0.114
SA7, DML1	120002.2, 124677.8	1034370.3, 1093079.7	0.302, 0.325	0.119, 0.129
SA7, DML2	112459.2, 117410.8	998967.3, 1064672.7	0.302, 0.325	0.110, 0.120
SA7, DML3	107871.8, 113237.2	951870.7, 1022185.3	0.322, 0.345	0.100, 0.111
SA10, DML1	111118.6, 116450.4	977391.4, 1038998.6	0.322, 0.344	0.104, 0.115
SA10, DML2	107656.0, 113479.0	964889.6, 1032993.4	0.342, 0.364	0.095, 0.106
SA10, DML3	101316.5, 107786.5	928692.7, 1004635.3	0.361, 0.384	0.083, 0.097
Slow life history, Poor data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	132753.3, 136130.7	1099949.4, 1155640.6	0.282, 0.307	0.204, 0.212
SA1, DML2	127265.3, 130982.7	1079549.3, 1140840.7	0.282, 0.307	0.193, 0.202
SA1, DML3	121155.4, 125190.6	1049911.9, 1115268.1	0.291, 0.317	0.180, 0.189
SA2, DML1	129972.4, 133582.6	1085146.9, 1141703.1	0.282, 0.306	0.160, 0.169
SA2, DML2	124204.8, 128096.2	1081505.2, 1143294.8	0.292, 0.316	0.151, 0.159
SA2, DML3	119572.1, 123815.9	1039970.5, 1105529.5	0.301, 0.326	0.140, 0.150
SA3, DML1	127879.0, 131597.0	1073693.7, 1131356.3	0.302, 0.326	0.140, 0.149
SA3, DML2	123181.1, 127271.9	1057084.0, 1119296.0	0.282, 0.306	0.133, 0.142
SA3, DML3	117131.5, 121556.5	1001495.9, 1068064.1	0.302, 0.326	0.124, 0.135
SA5, DML1	124400.1, 128547.9	1062058.1, 1120461.9	0.302, 0.326	0.119, 0.128
SA5, DML2	118512.2, 122961.8	1026847.4, 1089992.6	0.302, 0.326	0.111, 0.120

SA5, DML3	113994.7, 118796.3	990333.9, 1058806.1	0.302, 0.325	0.103, 0.114
SA7, DML1	120002.2, 124677.8	1034370.3, 1093079.7	0.302, 0.325	0.119, 0.129
SA7, DML2	112459.2, 117410.8	998967.3, 1064672.7	0.302, 0.325	0.110, 0.120
SA7, DML3	107871.8, 113237.2	951870.7, 1022185.3	0.322, 0.345	0.100, 0.111
SA10, DML1	111118.6, 116450.4	977391.4, 1038998.6	0.322, 0.344	0.104, 0.115
SA10, DML2	107656.0, 113479.0	964889.6, 1032993.4	0.342, 0.364	0.095, 0.106
SA10, DML3	101316.5, 107786.5	928692.7, 1004635.3	0.361, 0.384	0.083, 0.097

Table I.3: Chapter 1 95 % confidence intervals for each life history and DML scenarios for the high recruitment variability scenario. Combinations are paired by stock assessment interval and DML. SA is stock assessment interval; DML is data-management lag. Confidence intervals were calculated around the median with a Bonferonni correction to account for multiple comparisons.

Fast life history, Good data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	87475.8, 90242.2	639954.7, 662718.3	0.225, 0.245	0.123, 0.126
SA1, DML2	85721.0, 88556.2	638482.7, 663325.3	0.245, 0.265	0.121, 0.124
SA1, DML3	82753.3, 85789.1	626303.3, 654980.7	0.264, 0.285	0.115, 0.118
SA2, DML1	86473.2, 89306.0	631265.7, 654151.3	0.245, 0.265	0.103, 0.106
SA2, DML2	84733.2, 87642.7	631266.4, 656561.6	0.245, 0.265	0.100, 0.104
SA2, DML3	81620.4, 84793.8	611030.6, 640738.4	0.264, 0.285	0.094, 0.098
SA3, DML1	85061.6, 87978.2	623766.1, 647047.9	0.265, 0.284	0.095, 0.099
SA3, DML2	83330.9, 86364.0	618450.4, 644592.6	0.265, 0.284	0.091, 0.094
SA3, DML3	78943.0, 82299.8	604536.6, 635997.4	0.284, 0.304	0.085, 0.089
SA5, DML1	83000.1, 86009.2	606595.6, 631037.4	0.285, 0.303	0.085, 0.089
SA5, DML2	79058.1, 82321.5	591899.2, 620260.8	0.304, 0.323	0.077, 0.081
SA5, DML3	74383.4, 78169.1	566265.0, 601290.0	0.304, 0.323	0.072, 0.077
SA7, DML1	79251.5, 82641.9	584253.5, 609403.5	0.305, 0.322	0.087, 0.091
SA7, DML2	73833.1, 77653.2	564912.4, 596243.6	0.305, 0.323	0.078, 0.083
SA7, DML3	69489.8, 73787.9	539905.1, 712701.4	0.324, 0.343	0.072, 0.077
SA10, DML1	74772.2, 78596.8	541157.2, 721374.1	0.325, 0.342	0.079, 0.084
SA10, DML2	70396.3, 74714.3	543182.3, 719350.5	0.344, 0.362	0.069, 0.074
SA10, DML3	61621.3, 66444.2	526407.9, 695653.3	0.363, 0.382	0.062, 0.068
Slow life history, Good data quality				

COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	150476.1, 154219.9	1875916.0, 1933584.0	0.243, 0.267	0.092, 0.102
SA1, DML2	148824.3, 152742.7	1853386.2, 1914853.8	0.263, 0.286	0.090, 0.102
SA1, DML3	146938.2, 151229.8	1840031.4, 1907218.6	0.263, 0.286	0.088, 0.102
SA2, DML1	148276.5, 152121.5	1861145.0, 1918735.0	0.263, 0.286	0.072, 0.083
SA2, DML2	147312.6, 151429.4	1835705.1, 1897924.9	0.263, 0.286	0.071, 0.082
SA2, DML3	144606.0, 149201.0	1821800.1, 1891249.9	0.263, 0.286	0.068, 0.083
SA3, DML1	146445.0, 150389.0	1829653.2, 1887726.8	0.263, 0.286	0.063, 0.074
SA3, DML2	145628.9, 149940.1	1813508.0, 1877162.0	0.263, 0.286	0.061, 0.073
SA3, DML3	142502.4, 147268.6	1796882.0, 1867868.0	0.282, 0.306	0.059, 0.073
SA5, DML1	147136.9, 151273.1	1790257.5, 1849982.5	0.283, 0.306	0.052, 0.063
SA5, DML2	144667.5, 149424.5	1769197.4, 1837042.6	0.283, 0.305	0.050, 0.062
SA5, DML3	143380.3, 148878.7	1727454.2, 1806785.8	0.283, 0.306	0.048, 0.063
SA7, DML1	140641.3, 145623.7	1735141.7, 1798078.3	0.283, 0.305	0.051, 0.064
SA7, DML2	137907.9, 143534.1	1696627.4, 1771062.6	0.303, 0.325	0.049, 0.061
SA7, DML3	135211.4, 141776.6	1659905.1, 1747434.9	0.321, 0.345	0.047, 0.062
SA10, DML1	134069.0, 139947.0	1662429.2, 1733470.8	0.303, 0.325	0.043, 0.058
SA10, DML2	133028.2, 139735.8	1621504.1, 1704135.9	0.342, 0.364	0.041, 0.056
SA10, DML3	129926.3, 137399.7	1548317.4, 1646992.6	0.380, 0.404	0.037, 0.055
Fast life history, Poor data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	71843.5, 74770.9	534673.0, 712859.2	0.282, 0.306	0.210, 0.220
SA1, DML2	67652.1, 70744.7	531204.3, 705696.4	0.262, 0.287	0.202, 0.212
SA1, DML3	61931.1, 65298.8	522485.5, 686587.8	0.282, 0.306	0.192, 0.204

SA2, DML1	68838.5, 71862.2	520105.9, 693085.3	0.283, 0.306	0.167, 0.177
SA2, DML2	64358.1, 67594.5	509204.5, 674593.8	0.282, 0.306	0.161, 0.171
SA2, DML3	59780.6, 63217.6	489675.4, 642854.5	0.302, 0.325	0.152, 0.164
SA3, DML1	67086.7, 70191.9	576628.7, 617407.3	0.283, 0.306	0.150, 0.160
SA3, DML2	62667.9, 65939.0	545991.8, 590142.2	0.283, 0.306	0.144, 0.154
SA3, DML3	57691.7, 61296.7	471514.6, 519240.4	0.303, 0.325	0.136, 0.149
SA5, DML1	65073.4, 68433.5	560191.9, 602278.1	0.283, 0.305	0.129, 0.139
SA5, DML2	58602.4, 62218.2	510898.6, 556467.4	0.303, 0.325	0.120, 0.131
SA5, DML3	55108.9, 58822.2	446979.3, 495454.7	0.303, 0.325	0.112, 0.126
SA7, DML1	60871.0, 64481.4	517767.2, 559733.8	0.303, 0.324	0.129, 0.141
SA7, DML2	54659.1, 58504.5	483502.8, 530696.2	0.284, 0.305	0.118, 0.131
SA7, DML3	49621.3, 53727.0	414581.6, 464619.4	0.303, 0.324	0.113, 0.128
SA10, DML1	54647.6, 58557.4	461585.7, 505237.3	0.323, 0.343	0.111, 0.124
SA10, DML2	51016.8, 55241.1	444229.4, 492010.6	0.343, 0.363	0.102, 0.115
SA10, DML3	48176.1, 52546.0	369477.2, 421465.8	0.362, 0.383	0.097, 0.113
Slow life history, Poor data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, DML1	126137.0, 130750.0	1782806.7, 1894323.3	0.280, 0.308	0.177, 0.200
SA1, DML2	120021.4, 125070.6	1706822.4, 1825757.6	0.281, 0.308	0.175, 0.200
SA1, DML3	113942.0, 119496.0	1614846.9, 1741913.1	0.281, 0.308	0.175, 0.203
SA2, DML1	122375.8, 127276.2	1745292.5, 1857797.5	0.281, 0.308	0.133, 0.157
SA2, DML2	115239.6, 120601.4	1673043.0, 1793667.0	0.281, 0.307	0.131, 0.156
SA2, DML3	110700.6, 116524.4	1566875.1, 1696344.9	0.300, 0.327	0.129, 0.158
SA3, DML1	120020.0, 125223.0	1731691.8, 1846728.2	0.281, 0.307	0.114, 0.138

SA3, DML2	113391.7, 118968.3	1640192.8, 1763247.2	0.281, 0.307	0.112, 0.138
SA3, DML3	107718.2, 113873.8	1533750.3, 1665729.7	0.281, 0.307	0.110, 0.140
SA5, DML1	120417.8, 126038.2	1642783.7, 1761406.3	0.281, 0.307	0.094, 0.119
SA5, DML2	113742.1, 119956.9	1574607.6, 1701292.4	0.281, 0.307	0.092, 0.119
SA5, DML3	109899.4, 116557.6	1421926.8, 1558873.2	0.301, 0.327	0.090, 0.121
SA7, DML1	109654.0, 116055.0	1516804.5, 1639265.5	0.281, 0.307	0.086, 0.115
SA7, DML2	102764.9, 109601.1	1431748.4, 1563901.6	0.282, 0.307	0.084, 0.114
SA7, DML3	100577.4, 107725.6	1315740.7, 1456129.3	0.281, 0.307	0.085, 0.117
SA10, DML1	105383.6, 112932.4	1382681.9, 1517958.1	0.301, 0.326	0.072, 0.106
SA10, DML2	97370.3, 105272.7	1329330.9, 1471559.1	0.302, 0.326	0.070, 0.106
SA10, DML3	90926.1, 98981.5	1166312.8, 1315887.2	0.340, 0.366	0.067, 0.105

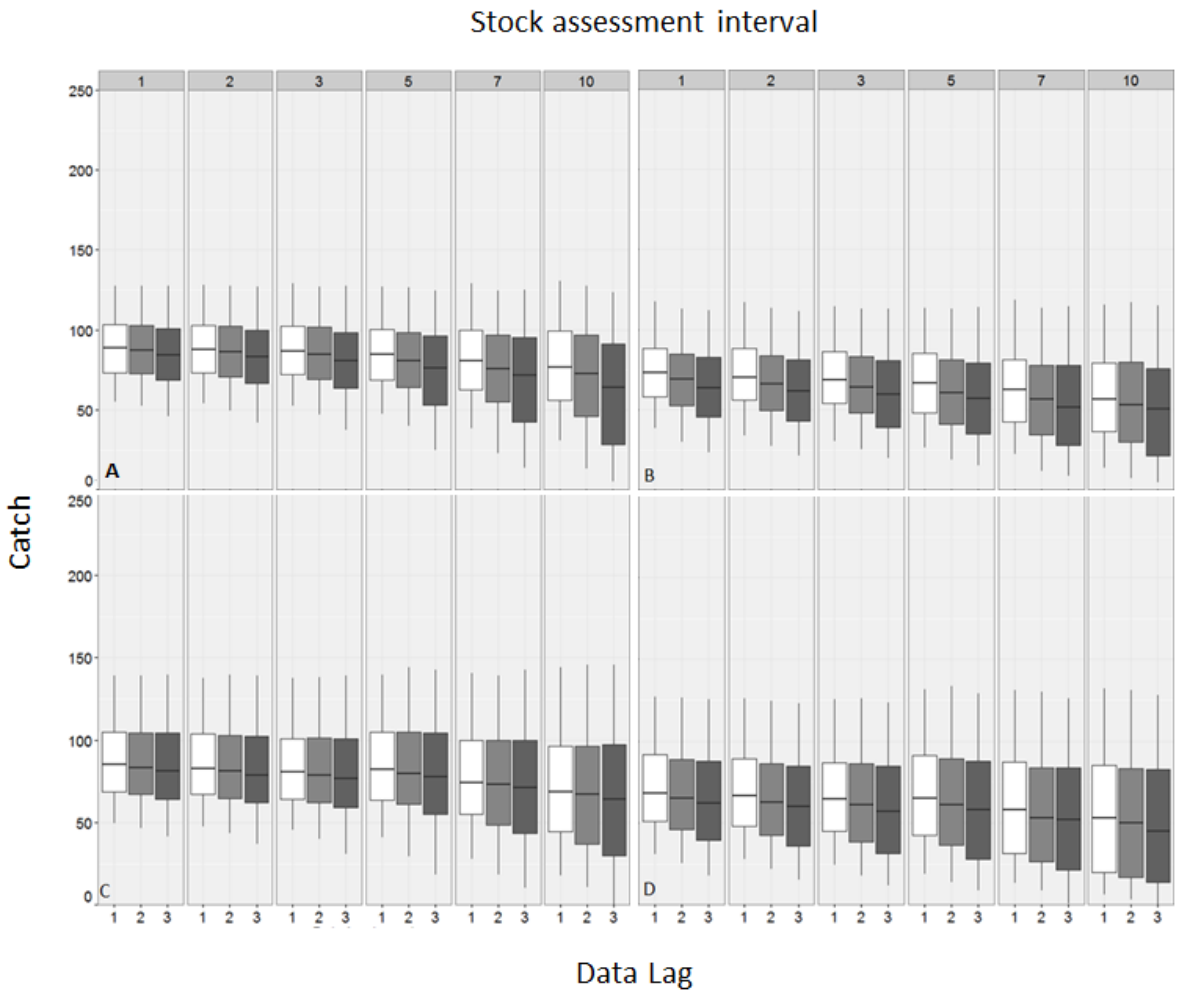

Figure I.1: Box plots of the catch for each life history and data scenarios for higher recruitment variability scenarios for chapter 1: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

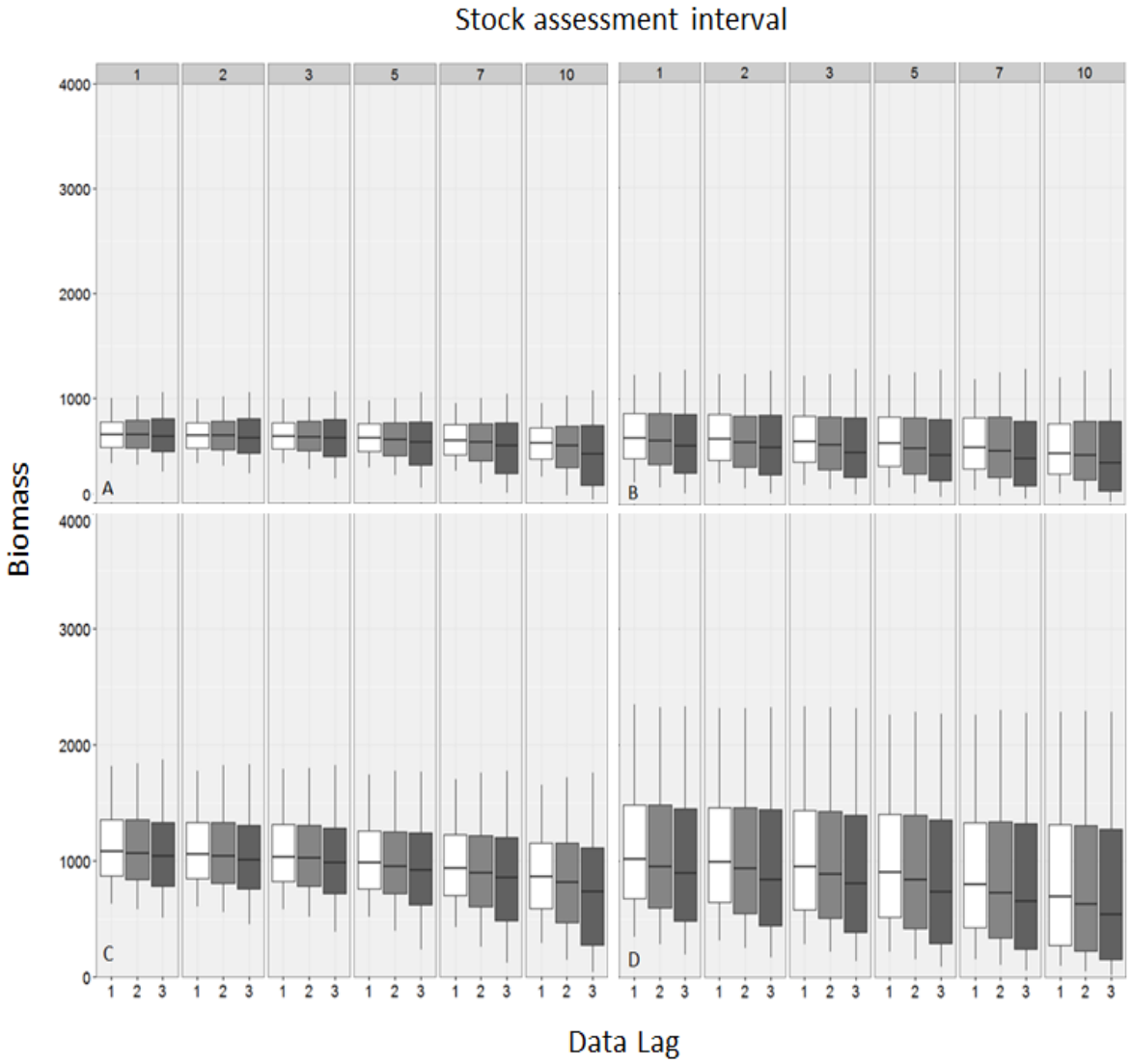

Figure I.2: Box plots of the Biomass for each life history and data scenarios for higher recruitment variability scenarios for chapter 1: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

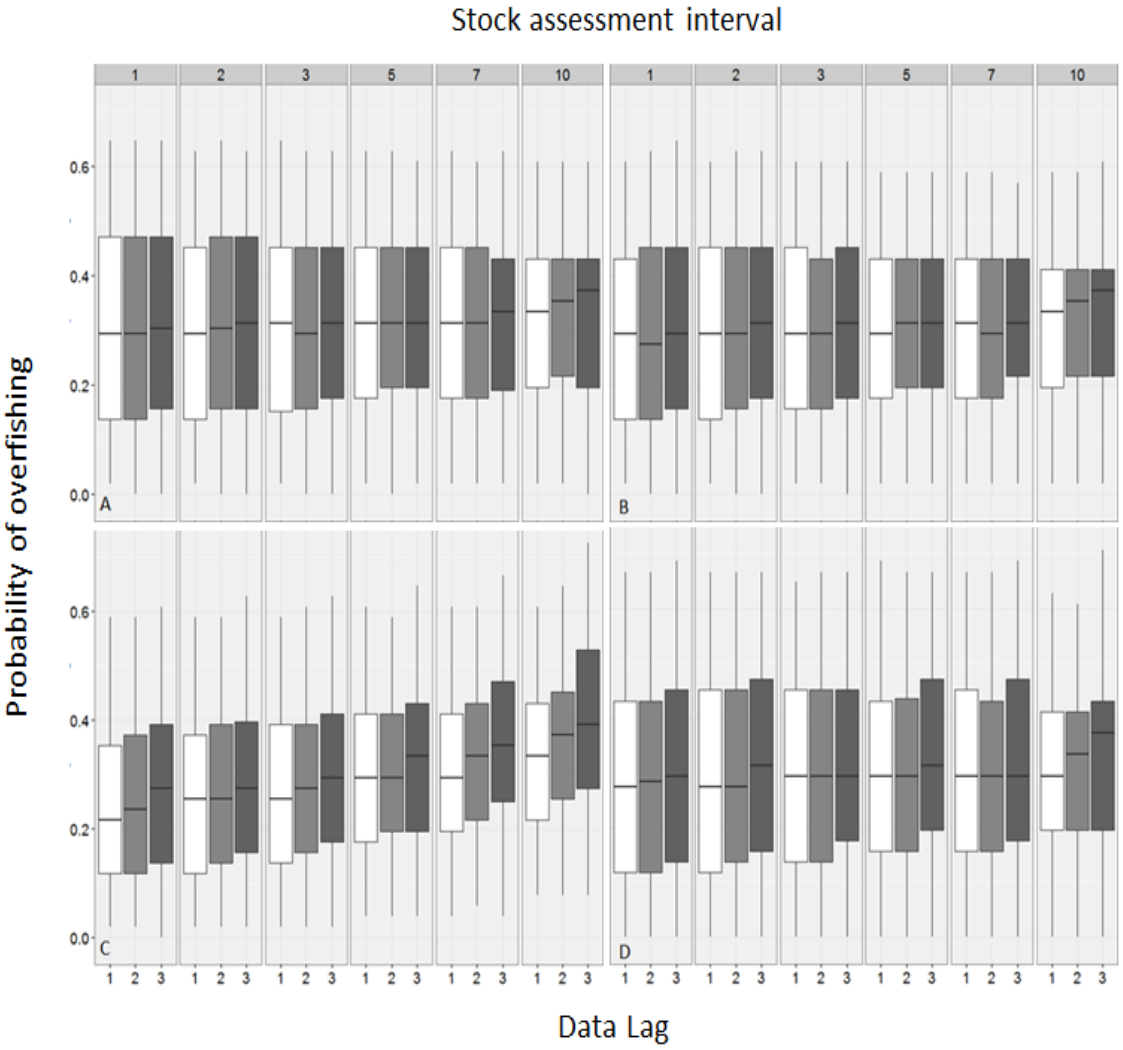

Figure I.3: Box plots of the probability of overfhisng for each life history and data scenarios for higher recruitment variability scenarios for chapter 1: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles.

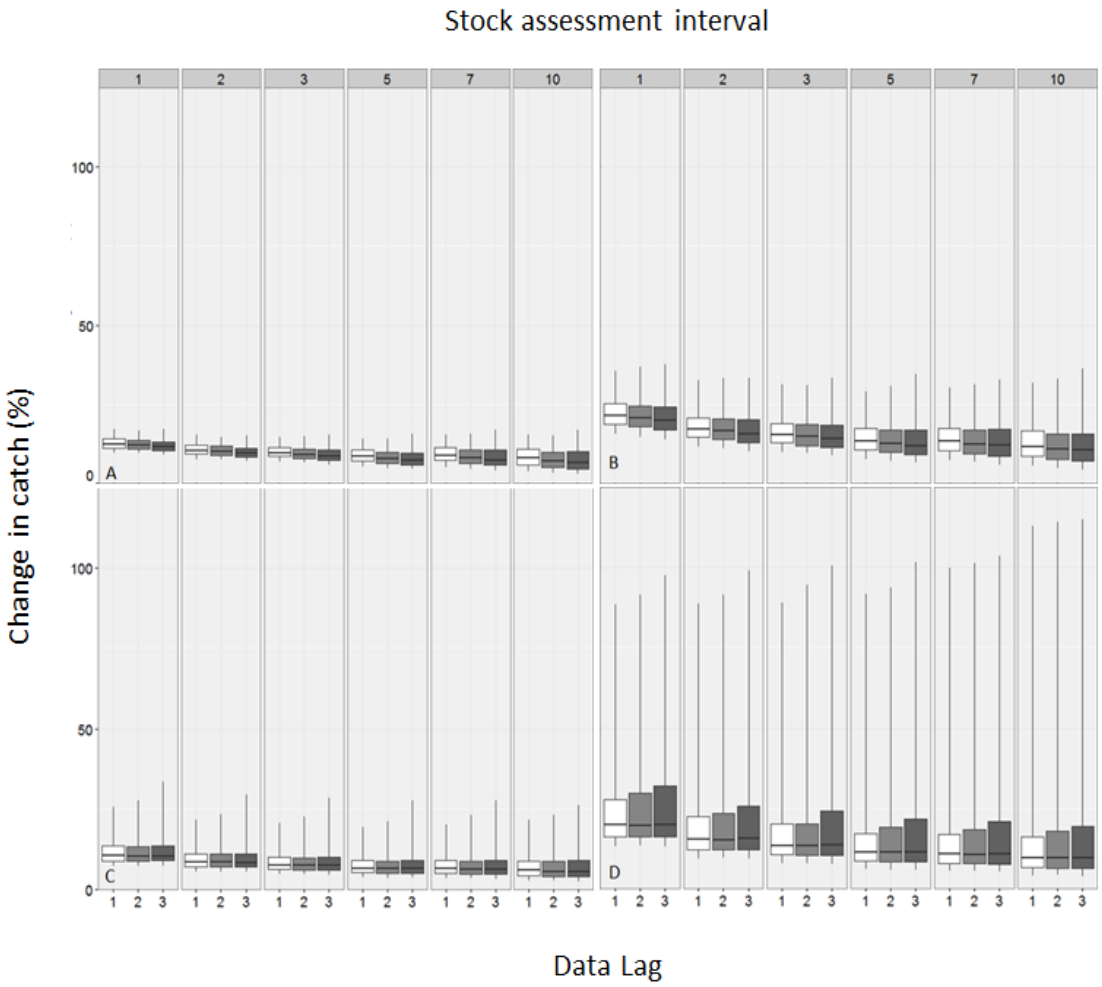

Figure I.4: Box plots of the aavc of catch for each life history and data scenarios for higher recruitment variability scenarios for chapter 1: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

Appendix II: Extended results of Chapter 3: Peformance of lag reduction method for Mid-Atlantic fishery management. Included medians, confidence intervals and additional figures.

Table II.1 Chapter 2 median values for low recruitment variability scenarios across performance metric. SA refers to stock assessment intervals and LR refers to data-management lag combinations. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag.

Fast life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, C1	151272	1071625	0.314	0.104
SA1, LR1	150787	1071108	0.314	0.105
SA1, LR2	151513	1073530	0.294	0.104
SA1, LR3	151404	1074033	0.304	0.104
SA1, C2	148778	1071985	0.314	0.100
SA2, C1	150236	1066635	0.314	0.084
SA2, LR1	149942	1065720	0.314	0.084
SA2, LR2	150296	1080848	0.314	0.086
SA2, LR3	150072	1082068	0.294	0.088
SA2, C2	147586	1065393	0.314	0.081
SA3, C1	149795	1060630	0.333	0.076
SA3, LR1	150034	1058583	0.333	0.077
SA3, LR2	150279	1066325	0.314	0.078
SA3, LR3	149778	1072783	0.314	0.079
SA3, C2	147452	1051935	0.333	0.072
Slow life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, C1	153431	1818715	0.314	0.087
SA1, LR1	153236	1817013	0.314	0.088
SA1, LR2	153158	1823608	0.314	0.088
SA1, LR3	153239	1821315	0.314	0.088
SA1, C2	150560	1790475	0.333	0.088
SA2, C1	150622	1784793	0.333	0.069
SA2, LR1	150458	1784065	0.333	0.069
SA2, LR2	150667	1800073	0.314	0.070

SA2, LR3	150253	1789953	0.314	0.072
SA2, C2	148391	1762400	0.333	0.069
SA3, C1	147528	1755475	0.333	0.062
SA3, LR1	147424	1753133	0.333	0.061
SA3, LR2	147718	1757645	0.333	0.063
SA3, LR3	149449	1773218	0.333	0.065
SA3, C2	146236	1716913	0.353	0.061
Fast life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, C1	129496	1060215	0.333	0.193
SA1, LR1	128932	1048833	0.333	0.197
SA1, LR2	129702	1067983	0.333	0.193
SA1, LR3	129452	1067698	0.333	0.193
SA1, C2	121379	995834	0.373	0.186
SA2, C1	127864	1038928	0.333	0.150
SA2, LR1	125762	1036578	0.333	0.153
SA2, LR2	126985	1067215	0.333	0.158
SA2, LR3	126563	1077100	0.314	0.166
SA2, C2	117797	970417	0.402	0.147
SA3, C1	124191	1029425	0.353	0.135
SA3, LR1	123921	1013330	0.353	0.137
SA3, LR2	125190	1042328	0.353	0.139
SA3, LR3	124637	1052228	0.333	0.143
SA3, C2	114218	948639	0.431	0.129
Slow life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
SA1, C1	126825	1738565	0.333	0.173
SA1, LR1	125903	1719433	0.333	0.172
SA1, LR2	126953	1734825	0.333	0.172
SA1, LR3	127292	1723683	0.333	0.173
SA1, C2	121632	1648115	0.353	0.174
SA2, C1	124391	1671720	0.353	0.132
SA2, LR1	123679	1653935	0.353	0.132
SA2, LR2	124056	1680798	0.333	0.135
SA2, LR3	123873	1704243	0.333	0.141
SA2, C2	118721	1600868	0.373	0.134
SA3, C1	122659	1658933	0.353	0.115
SA3, LR1	122761	1642695	0.353	0.116
SA3, LR2	122151	1644603	0.353	0.118
SA3, LR3	122150	1651048	0.353	0.121
SA3, C2	117247	1522060	0.392	0.115

Table II.2: Chapter 2 median values for high recruitment variability scenarios across performance metrics for chapter two. SA refers to stock assessment intervals and LR refers to data lag reduction methods. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag.

Fast life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	84753	615495	0.314	0.116
1, LRM1	84488	613478	0.314	0.117
1, LRM2	84921	615595	0.314	0.116
1, LRM3	85040	615933	0.314	0.116
1, C2	81990	585875	0.353	0.116
2, C1	83632	606821	0.333	0.096
2, LRM1	83304	604189	0.333	0.097
2, LRM2	83635	611396	0.314	0.099
2, LRM3	83541	615002	0.314	0.101
2, C2	80212	566302	0.373	0.097
3, C1	82578	594350	0.353	0.089
3, LRM1	82457	590652	0.353	0.089
3, LRM2	82579	595780	0.333	0.090
3, LRM3	82512	598265	0.333	0.092
3, C2	78914	536031	0.412	0.086
Slow life history, Good data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	89572	1056896	0.345	0.121
1, LRM1	89408	1053321	0.348	0.121
1, LRM2	89574	1057143	0.346	0.121
1, LRM3	89531	1057128	0.346	0.121
1, C2	87897	1031627	0.375	0.127
2, C1	87593	1028944	0.364	0.101
2, LRM1	87426	1029365	0.366	0.101
2, LRM2	87585	1032889	0.360	0.103
2, LRM3	87345	1036706	0.356	0.104
2, C2	85995	1003133	0.396	0.106
3, C1	85612	1003059	0.384	0.093
3, LRM1	85319	998842	0.386	0.093
3, LRM2	85427	1004223	0.380	0.094
3, LRM3	85587	1008792	0.378	0.095
3, C2	83380	963788	0.426	0.097
Fast life history, Poor data quality				

Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	127909	1083008	0.358	0.209
1, LRM1	126266	1072118	0.363	0.213
1, LRM2	127889	1088163	0.357	0.209
1, LRM3	128029	1090168	0.358	0.209
1, C2	111115	971169	0.435	0.215
2, C1	125073	1062709	0.364	0.168
2, LRM1	123188	1048546	0.370	0.171
2, LRM2	125220	1085682	0.352	0.174
2, LRM3	124590	1098877	0.340	0.181
2, C2	106630	935905	0.453	0.180
3, C1	122432	1043462	0.370	0.153
3, LRM1	120776	1028758	0.375	0.156
3, LRM2	123078	1059602	0.361	0.157
3, LRM3	122677	1073655	0.352	0.161
3, C2	101539	892323	0.477	0.166
Slow life history, Poor data quality				
Combination	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	72604	1036881	0.373	0.274
1, LRM1	71759	1026289	0.377	0.275
1, LRM2	72659	1039139	0.373	0.273
1, LRM3	72621	1040613	0.373	0.272
1, C2	67689	972655	0.420	0.292
2, C1	70004	1002897	0.383	0.230
2, LRM1	69206	990899	0.389	0.232
2, LRM2	69689	1009141	0.378	0.234
2, LRM3	69883	1021401	0.369	0.239
2, C2	65307	939489	0.435	0.249
3, C1	67930	972333	0.395	0.213
3, LRM1	67035	961835	0.400	0.216
3, LRM2	67886	976565	0.391	0.216
3, LRM3	67913	987825	0.386	0.217
3, C2	62817	898458	0.459	0.233

Table II.3: Chapter 2 95 % confidence intervals for each life history and DML scenarios for the high recruitment variability scenario. Combinations are paired by stock assessment interval and data lag reduction methods. SA refers to stock assessment intervals and LR refers to data lag reduction methods. C1 represents control 1 of the lag reduction methods with an annual data lag, LRM1 is lag reduction method 1, LR2 is lag reduction method 2 and LRM3 is lag reduction method 3. C2 is control 2 of the lag reduction method with a 2 year data lag. Confidence intervals were calculated around the median with a Bonferonni correction to account for multiple comparisons.

Fast life history, Good data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	83297.9, 86208.6	579606.3, 604808.7	0.301, 0.326	0.114, 0.117
1, LRM1	83026.2, 85949.1	578086.3, 603479.2	0.301, 0.326	0.115, 0.118
1, LRM2	83465.2, 86376.6	578831.4, 604031.1	0.301, 0.326	0.114, 0.117
1, LRM3	83577.7, 86502.9	580688.0, 605997.5	0.301, 0.326	0.114, 0.117
1, C2	80297.5, 83682.1	557771.6, 588951.9	0.337, 0.368	0.114, 0.118
2, C1	82133.9, 85130.1	571155.4, 596849.6	0.321, 0.346	0.095, 0.098
2, LRM2	81795.0, 84813.4	569558.5, 595615.5	0.321, 0.346	0.095, 0.098
2, LRM3	82141.0, 85129.8	575072.2, 600849.8	0.302, 0.326	0.097, 0.100
2, LRM4	82043.2, 85039.3	579164.9, 605347.1	0.302, 0.326	0.099, 0.102
2, C2	78358.3, 82064.8	542959.7, 576126.3	0.357, 0.388	0.095, 0.099
3, C1	81013.5, 84143.0	556892.9, 583786.6	0.341, 0.365	0.087, 0.091
3, LRM1	80868.4, 84046.4	557559.8, 584895.2	0.341, 0.365	0.088, 0.091
3, LRM2	81002.3, 84155.4	562705.7, 589845.8	0.321, 0.346	0.089, 0.092
3, LRM3	80926.7, 84097.7	562663.8, 590346.7	0.321, 0.346	0.090, 0.094
3, C2	76858.7, 80969.2	521567.5, 557599.5	0.395, 0.429	0.084, 0.089
Slow life history, Good data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	84471.0, 88014.5	980085.0, 1029755.0	0.299, 0.328	0.090, 0.101
1, LRM1	84794.3, 88352.4	976380.1, 1026261.9	0.309, 0.338	0.090, 0.101
1, LRM2	84487.1, 88024.4	979025.2, 1028685.3	0.299, 0.328	0.090, 0.101
1, LRM3	84679.8, 88224.7	979531.5, 1029326.5	0.299, 0.328	0.090, 0.102
1, C2	82840.2, 86540.4	942973.4, 996325.1	0.337, 0.369	0.090, 0.104
2, C1	82647.2, 86270.9	949555.1, 999510.9	0.338, 0.368	0.071, 0.082
2, LRM2	82358.5, 86000.8	945179.6, 996250.4	0.338, 0.368	0.072, 0.083
2, LRM3	83131.4, 86770.7	953722.2, 1004146.3	0.319, 0.348	0.073, 0.084
2, LRM4	82732.1, 86359.0	956841.4, 1007552.6	0.319, 0.348	0.075, 0.086
2, C2	80376.1, 84207.2	914292.0, 968617.0	0.357, 0.388	0.072, 0.085
3, C1	80536.0, 84203.0	913131.4, 963703.1	0.358, 0.387	0.064, 0.075
3, LRM1	80365.3, 84058.1	912485.7, 963347.3	0.358, 0.387	0.064, 0.075

3, LRM2	80235.6, 83908.9	913851.1, 964685.4	0.358, 0.387	0.065, 0.076
3, LRM3	80561.4, 84261.5	915892.8, 967255.2	0.358, 0.387	0.066, 0.077
3, C2	78396.4, 82380.6	874486.8, 930373.7	0.395, 0.428	0.063, 0.077
Fast life history, Poor data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	127568.3, 131423.7	1029996.8, 1090433.2	0.318, 0.348	0.189, 0.197
1, LRM1	126889.6, 130973.4	1017022.0, 1080643.0	0.318, 0.349	0.192, 0.201
1, LRM2	127760.9, 131642.1	1036910.3, 1099054.7	0.318, 0.348	0.189, 0.197
1, LRM3	127514.0, 131390.0	1036611.5, 1098783.5	0.318, 0.348	0.189, 0.198
1, C2	118399.0, 124359.0	957009.2, 1034658.3	0.353, 0.392	0.179, 0.193
2, C1	125743.1, 129983.9	1007044.0, 1070811.0	0.318, 0.348	0.146, 0.155
2, LRM2	123538.9, 127984.1	1003869.6, 1069285.4	0.318, 0.349	0.148, 0.158
2, LRM3	124930.8, 129039.2	1035461.3, 1098968.7	0.319, 0.348	0.153, 0.162
2, LRM4	124453.9, 128671.1	1044723.6, 1109476.4	0.300, 0.328	0.162, 0.170
2, C2	114623.7, 120969.3	930327.9, 1010505.6	0.382, 0.422	0.140, 0.154
3, C1	121976.0, 126405.0	996838.9, 1062011.1	0.338, 0.368	0.130, 0.139
3, LRM1	121563.0, 126278.0	979703.8, 1046956.2	0.338, 0.368	0.132, 0.141
3, LRM2	122951.7, 127428.3	1009667.6, 1074987.4	0.338, 0.368	0.134, 0.144
3, LRM3	122410.3, 126863.7	1019362.5, 1085092.5	0.319, 0.348	0.139, 0.147
3, C2	110857.7, 117578.3	907267.6, 990009.4	0.411, 0.452	0.121, 0.137
Slow life history, Poor data quality				
COMB.	Catch	Biomass	Probability of overfishing	AAV in catch
1, C1	67389.7, 71311.3	884927.7, 967382.3	0.336, 0.370	0.167, 0.195
1, LRM1	66903.5, 70869.0	873092.0, 956400.5	0.336, 0.370	0.168, 0.196
1, LRM2	67334.2, 71259.5	878723.6, 961529.9	0.336, 0.370	0.167, 0.196
1, LRM3	67765.4, 71700.6	886830.3, 969919.7	0.326, 0.360	0.169, 0.196
1, C2	63543.0, 67798.0	809960.7, 899339.3	0.354, 0.391	0.167, 0.199
2, C1	65474.9, 69530.0	841635.3, 925069.2	0.336, 0.370	0.125, 0.154
2, LRM2	64624.7, 68751.0	824875.2, 909202.8	0.336, 0.370	0.125, 0.155
2, LRM3	64937.8, 69040.5	837876.0, 922305.0	0.336, 0.370	0.129, 0.158
2, LRM4	65412.4, 69514.6	856477.8, 941640.2	0.337, 0.369	0.134, 0.163
2, C2	60858.8, 65265.2	758923.4, 849637.6	0.373, 0.411	0.127, 0.159
3, C1	63600.6, 67805.4	808774.7, 893515.8	0.356, 0.389	0.108, 0.137
3, LRM1	63254.4, 67526.3	799423.4, 885263.6	0.355, 0.390	0.108, 0.138
3, LRM2	63204.3, 67442.5	796761.3, 882495.7	0.356, 0.389	0.112, 0.141
3, LRM3	63604.7, 67817.0	817244.0, 903822.5	0.356, 0.389	0.114, 0.143
3, C2	59133.7, 63772.3	719662.7, 812468.8	0.411, 0.451	0.107, 0.140

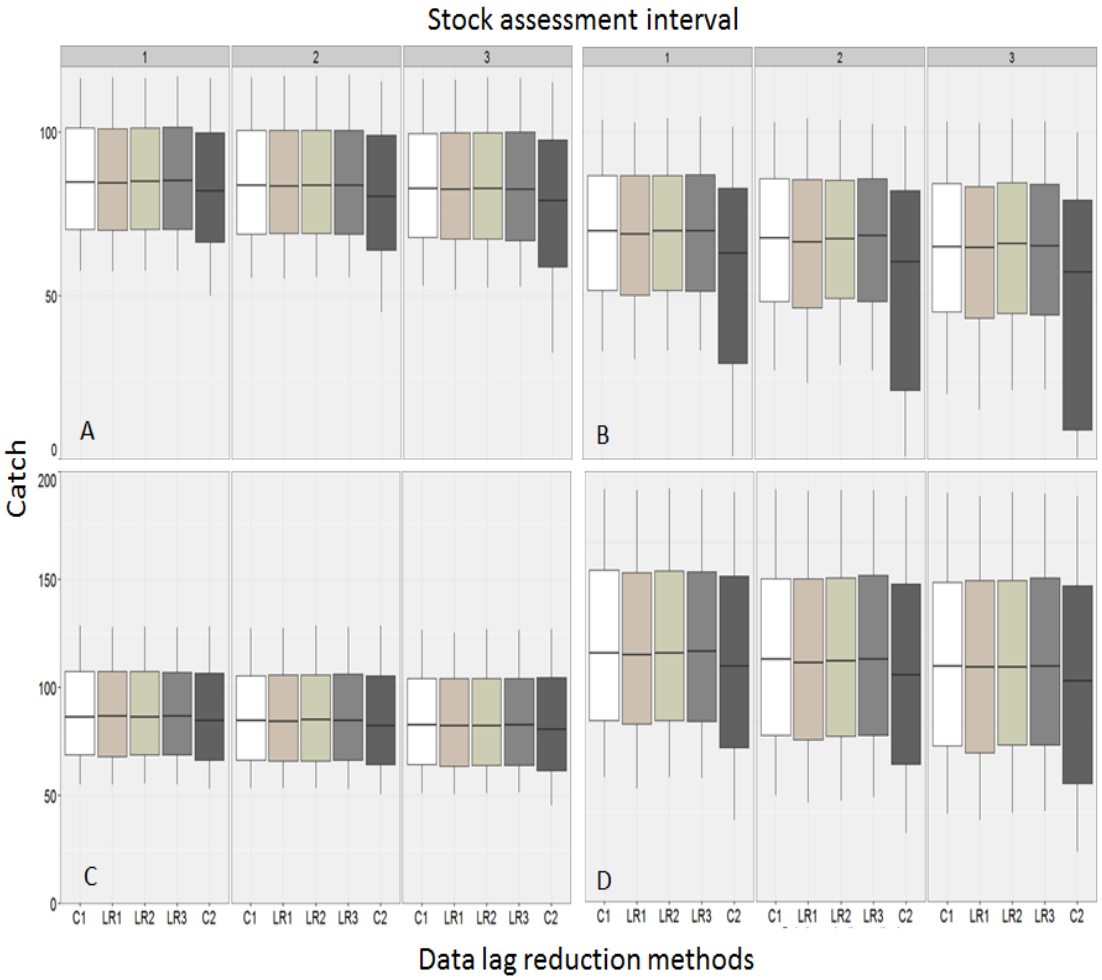

Figure II.1: Box plots of the catch for each life history and data scenarios for higher recruitment variability scenarios for chapter 2: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles.

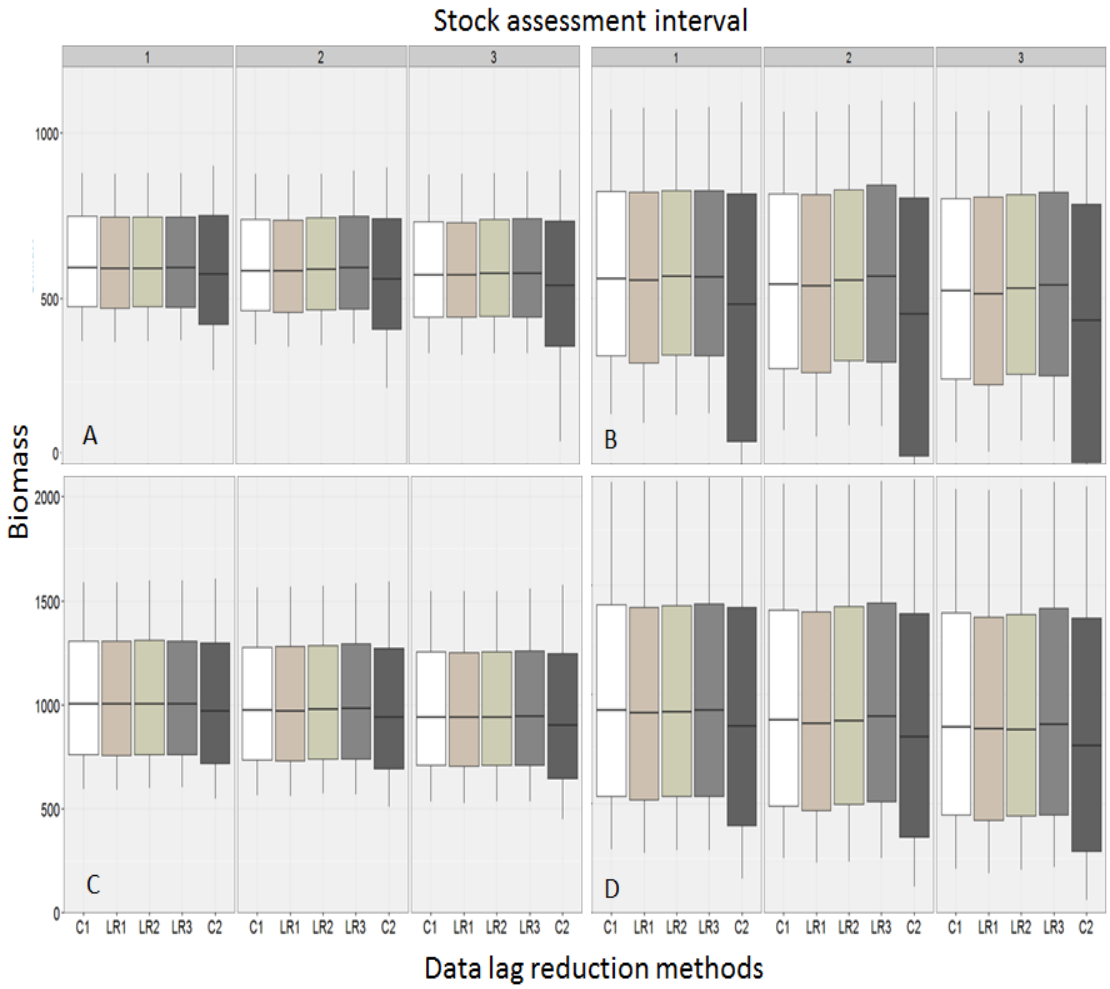

Figure II.2: Box plots of the Biomass for each life history and data scenarios for higher recruitment variability scenarios for chapter 2: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10th and 90th percentiles.

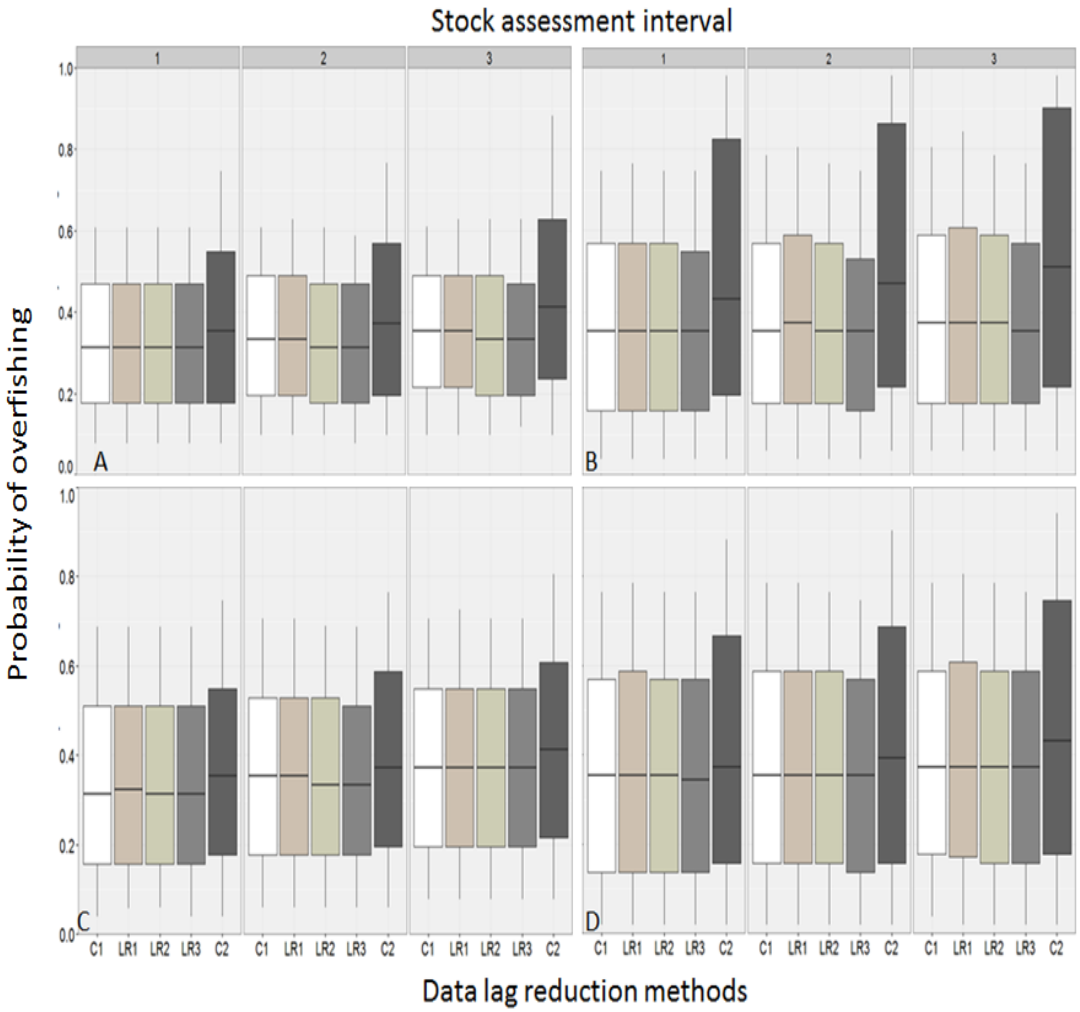

Figure II.3: Box plots of the probability of overfishing for each life history and data scenarios for higher recruitment variability scenarios for chapter 2: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles.

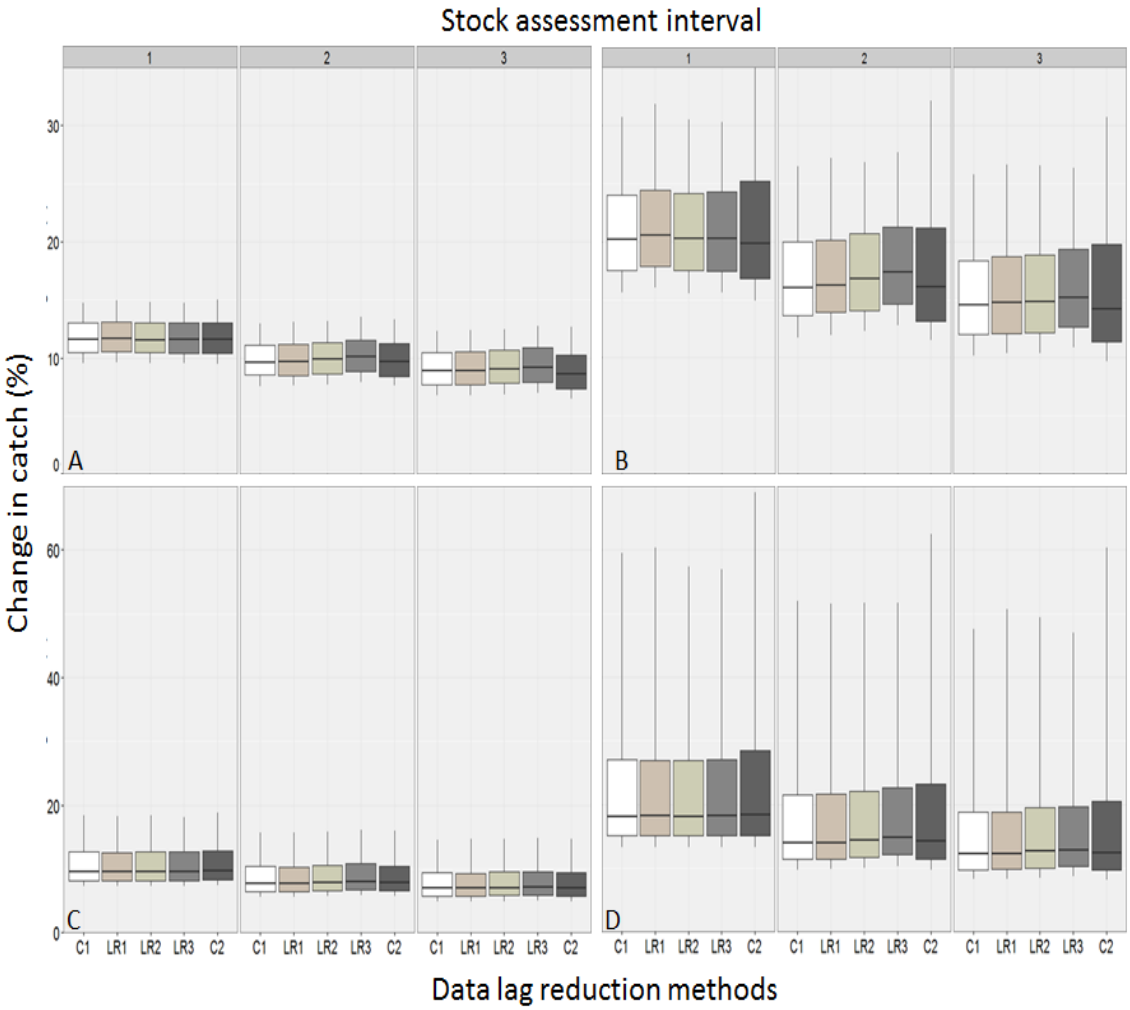

Figure II.4: Box plots of the AAV for catch for each life history and data scenarios for higher recruitment variability scenarios for chapter 2: A) fast life history with good data, B) fast life history with poor data, C) slow life history with good data, and D) slow life history with poor data. The horizontal lines of the box plot show the median catch, the box shows the interquartile range, and the whiskers represent the 10^{th} and 90^{th} percentiles

Appendix III. ADMB code for chapter one and chapter two.

Code III.1 Chapter 1 ADMB code- Operating model: the operating model represented the true dynamics of the stock using an age-structured population model and implemented the management portion of the simulation by applying the ABC to the stock.

```
// Code created by:
// John Wiedenmann and Andrea Sylvia
// Operating model for the ABC MSE
// Last modified Feb 24th, 2015
// ############   NOTES  ####################
//
// This file is the operating model. It is the core of the MSE simulation
// and calls additional executables for various purposes.
//
// ########################################

// =========== TO DO ======================
// Add in control rule specifics
//
// Add projections?

  //TOP_OF_MAIN_SECTION
  //time(&start);
  //arrmblsize = 2880117328;
  //gradient_structure::set_GRADSTACK_BUFFER_SIZE(1.e7);
  //gradient_structure::set_CMPDIF_BUFFER_SIZE(1.e7);
  //gradient_structure::set_MAX_NVAR_OFFSET(5000);
  //gradient_structure::set_NUM_DEPENDENT_VARIABLES(5000);

DATA_SECTION


  //
**********************************************************************
*****
  //  DATA READ IN FROM:
  //       • Main parameters are in rec_opmodel.dat
  //       • Life history specific parameters are in
  //                 - slow life history -> slow_lh.dat
  //                 - medium life history -> med_lh.dat
  //       - fast life history -> fast_lh.dat
  //
**********************************************************************
*****
```

```
  init_int i_max              // total iterations
  init_int lag_runs           // # of different data lag and assessment intervals (ex: 3 la
  init_vector lag_vec(1,4)
  init_int int_runs            // # of different
  init_vector int_vec(1,6)  //
  init_int m_runs                 // # of sa error and recruitment runs
  init_int years                  // years in the model
  init_int first_data_yr          // year when data first collected
  init_int first_SA_yr            // year when assessment first done
  init_int cr_runs                // control rule runs
  init_int lh                     // life history index
  init_vector amax_vec(1,3)    // age classes in model
  init_vector h_vec(1,3)
  init_vector M_vec(1,3)
  init_number t0                  // age at length = 0
  init_number L_inf               // L infinity
  init_number a_LW                // L-W scale
  init_number b_LW                // L-W exponent
  init_number SPR_lim             // SPR limit (i.e. F35%)
  init_number SPR_targ            // SPR target
  init_number q_surv              // catchability of the survey
  init_vector q_sigma(1,2)
  init_vector ESS(1,2)             // effective sample size
  init_int F_shape                // controls the shape of the initial F pattern
  init_vector Fmult(1,3)
  init_number Finit_sd            // s.d. of initial F
  init_number M_sigma             // variability in M
  init_vector M_cor(1,2)          // autocorrelation
  init_vector rel_M_bounds(1,2)  // relative bounds on sf50
  init_number s_sigma             // variability in sf50
  init_vector s_cor(1,2)          // autocorrelation in sf50
  init_vector rel_s_bounds(1,2)          // relative bounds on sf50
  init_number C_pse               // PSE in catch estimates
  init_vector I_pse(1,2)          // PSE in index
  init_int acor_switch
  init_number Fcor
  init_number R0                  // unfished recruitment
  init_number R_cor
  init_vector R_sigmas(1,2)    // values for S-R variability
  init_number Rboom_yrs_mult // avg number of years for a boom recruitment to
occur (as a multiple of amax)
  init_number Rboom_mult     // dtermines the size of the boom recruitment event
  init_number M_k_ratio      // ratio of M / k
  init_number M_mat_mult     // product of M and age at maturity
```

```
 init_number surv_sel_mult  // multiplier relating survey a50 to fishery a50
 init_vector abc_mult(1,2)     // multiplier ofr calculating the ABC
 init_matrix pstar_table(1,3,1,100)  //table of P star values

 init_int EOF_flag_1                    // end of file flag

 //!!ad_comm::change_datafile_name("med_lh.dat");
 //init_int a_r                         // age at recruitment to pop
 //init_number h                              // steepness
 //init_number M                              // Natural mortality
 //init_int EOF_flag_2                  // end of file flag


 int a_r
 int amax
 int i                                  // integer index
 int t                                  // year index
 int tt                             // year index used to set future catches
 int tt_max                         // final year catches are projected (following and
assessment)
 int ttt                            // year index used for printing out results
 int temp                               // int used for printing out data to be read into
assessment model
 int a                                  // age index
 int sa
 int seed                               // random number seed
 int last_SA_yr                     // year when last assessment is done
 int SA_yr                              // year when an assessment is to be conducted
 int SA_num                         // counter of the number of assessments
 int total_SAs        // total nuber of assessment
 int file_iters             // number of data points read in for a particular variable
 int fi                             // index used for reading in output from assessment
 int f              // index for calculating lognormal densitites
 int xmax
 int F_iters          // iterations for the
 int fm
 int eh                                 // index for different exploitation histories
 int rv                                 // index for recrutiment variability
 int iv                                 // index for survey index variability
 int RR
 int r
 int rr
 int cnt
 int lag_int
 int sa_int
 int mr
```

```
 int mr_max
 int cr
 int ps_int
 int data_yrs
 int proj_years
 int max_ESS
 int min_ESS
 int R_type
 int tvp_int
 int rp_switch
 int proj_switch
 int abc_switch
 int aa
 int ps
 int rp
 int max_SA_interval
 int SA_interval
 int data_lag
 int mlv //ASylvia  maximum lag

 !! max_ESS = ESS(1);
 !! min_ESS = ESS(2);
 !! max_SA_interval = int_vec(6)+1;  //Asylvia added +1
 !! mlv=max(lag_vec);     //added by Asylvia
 //!! cout << max_SA_interval << endl;
 //!! data_yrs = years - first_data_yr - data_lag + 1;

 // Vectors and matrices used to generate random variables
 vector R_error(1,years)                    // recruitment deviations
 vector Rboom_error(1,years)                // recruitment deviations
 vector C_error(1,years)                    // Error in observed catch
 vector I_error(1,years)                    // Abundance index error
 vector M_error(1,years)                    // M variability
 vector q_error(1,years)                    // M variability
 vector sf50_error(1,years)                 // variability in age @ 50% selex in
fishery
 matrix pC_error_lg(1,years,1,max_ESS)      // rv's used to calculate multinomial r.v.
for catch
 matrix pI_error_lg(1,years,1,max_ESS)      // rv's used to calculate multinomial r.v.
for index
 matrix pC_error_sm(1,years,1,min_ESS)      // rv's used to calculate multinomial r.v.
for catch
 matrix pI_error_sm(1,years,1,min_ESS)      // rv's used to calculate multinomial r.v.
for index

 vector F_error(1,years)
```

```
number Fmax;
number pstar_imp  //implemented pstar value based on B/Bmsy

vector sa_err_vec(1,16)
vector R_sigma_vec(1,16)
vector R_type_vec(1,16)
vector proj_vec(1,16)
vector abc_vec(1,16)
vector rp_vec(1,16)

vector first_est_yr(1,lag_runs)
vector final_est_yr(1,lag_runs)

vector est_Flim(1,years)
vector est_Starg(1,years)
vector est_termR(1,years)
vector est_termS(1,years)
vector est_termF(1,years)
vector est_OFL(1,years)
vector OFL_in(1,max_SA_interval)

//
number h
number M
number ss50
number sf50
number m50
number k
number ss_slope
number sf_slope
number m_slope
number boom_prob

!! amax = amax_vec(lh);
!! h = h_vec(lh);
!! M = M_vec(lh);
!! m50 = M_mat_mult / M;
!! sf50 = m50;
!! ss50 = sf50 * surv_sel_mult;
!! a_r = floor(ss50);
!! if(a_r < 1) a_r = 1;
!! boom_prob = 1.0 / (amax * Rboom_yrs_mult);

//Previously from the Parameter Section
number S0                    // Unfished spawning biomass
number alpha                 // stock-recruit alpha (beverton-holt model)
```

```
number beta                    // stock-recruit beta (beverton-holt model)

number C_var_obs               // Variance in commercial catch
number C_sigma                 // s.d. of commercial catch
vector I_var_obs(1,2)          // Variance in index
vector I_sigma(1,2)            // variability in recruitment deviations
number R_sigma

                                // These parameters control the initial F
                                // If F_shape = 0 it plateaus, if = 1 it's dome-shaped
number F_0
number F_end
number F_slope_up
number F_slope_down
number F_plat                          //
number F_int_up                //
number F_int_down     //
number Yr_plat                 //

// Biological Reference Points
number Flim                    // Limit fishing mortality rate (Fmsy of Fx% proxy)
number Starg                   // target spawning biomass (Smsy)
number Flim_temp               // temp estimate of Flim read in from assessment
number Starg_temp              // temp estimate of Starg read in from assessment
number S_temp
number R_temp
number F_temp
number eq_Fmult
number Fmult_msy
number Fmult_spr
vector Zmsy(a_r,amax)
vector Zproxy(a_r,amax)

number S_ratio
number term_S
number OFL_buffer
number C_temp;

vector L(a_r,amax)             // Length at age
vector W(a_r,amax)             // Weight at age
vector m(a_r,amax)             // maturity at age
vector ss(a_r,amax)            // selex at age in survey
vector sf(a_r,amax)       // selex at age in com fishery
vector M_t(1,years)            // M as a function of t
vector sf50_t(1,years)         // slectivity as a function of t
vector mu_sf(a_r,amax)
```

```
  matrix N(1,years,a_r,amax)  // Numerical abundance
  matrix Z(1,years,a_r,amax)  // Total mortality
  matrix sf_t(1,years,a_r,amax)          // fishery selectivity

  matrix C_act(1,years,a_r,amax)       // Actual catch at age in fishery
  matrix C_obs(1,years,a_r,amax)       // Obs. catch at age in fishery

  matrix I_obs(1,years,a_r,amax)                // Obs. abundance index at age
  matrix pI_obs(1,years,a_r,amax)               // Proportion of Index abundance
  matrix pC_obs(1,years,a_r,amax)       // Proportion of catch

  vector N_init(a_r,amax)             // Initial abundance
  vector N_0(a_r+1,amax)              // Initial abundance in the assessment
  vector R(1,years)                  // Recruits
  vector Rboom_vec(1,years)           // Recruits
  vector S(1,years)                  // Spawning Biomass by year
  vector relF(1,first_SA_yr)          // relative Fishing mortality
  vector F_init(1,first_SA_yr)        // Fishing mortality
  vector F(1,years)                  // Fishing mortality
  vector Fcom(1,years)                // Fishing mortality
  vector Flim_OFL(1,years)
  vector Fmsy_OFL(1,years)
  vector CW_tot(1,years)              // catch in weight
  vector I_obs_tot(1,years)                              // Total index
  vector qt(1,years)                                     // Effective sample size for index
at age

  vector est_F(1,years)     // estimated F in fishery
  vector est_S(1,years)           // estimated Spawning biomass
  vector est_R(1,years)                  // estimated Recruitment
  vector ABC(1,years)                    // ABC
  vector OFL(1,years+mlv)                // OFL ASylvia edit
  vector est_sf50(1,years)
  vector est_sf_slope(1,years)
  vector gmax(1,years)

  number est_sf50_temp
  number est_sf_slope_temp
  number gmax_temp

  // Variable used in the calculation of SPR-based BRPs
  number mu_sf50
  vector N_brp(a_r,amax)
  vector Z_brp(a_r,amax)
  number F_brp
```

```
 number S_brp
 number SPR
 number SPR_proxy
 number Sbrp_0
 number YPR
 number R_brp
 number Y_brp

 number minF
 number maxF
 number Finc
 number MSY
 number Smsy
 number Fmsy
 number F_proxy
 number S_proxy

 number Fmax_in
 number Fmult_in

 //intermediate numbers for calculating Pstar
 number t1
 number t2


 LOCAL_CALCS

                if(EOF_flag_1 != 12345)
                {

                        // If the last value read in from the .dat file
                        // doesn't match up, print the values to check
                        // where the error might be
                        cout << "-------------------------------------------" << endl;

                        cout <<" Data not read in properly to abc_opmodel.tpl!"<<
endl;
                        cout << "-------------------------------------------" << endl;
                        cout << "i_max" <<" " << i_max << endl;
                        cout <<"years" << " "<< years << endl;
                        cout <<"first_data_yr" << " "  << first_data_yr << endl;
                        cout <<"first_SA_yr" << " "  << first_SA_yr << endl;
                cout <<"SA_interval" << " "  << SA_interval << endl;
                        cout <<"data_lag" << " " << data_lag << endl;
                        cout <<"proj_switch" << " " << proj_switch << endl;
                        cout <<"rp_switch" << " " << rp_switch << endl;
```

```
                cout <<"cr_runs" << " " << cr_runs << endl;
                cout <<"q_surv" << " " << q_surv << endl;
                cout <<"ESS" << " " << ESS << endl;
                cout <<"F_shape" << " " << F_shape << endl;
                cout <<"EOF_flag_1" << " " << EOF_flag_1 << endl;
                cout << "lh" << " " << lh << endl;
                cout <<"amax" << " " << amax << endl;
                cout <<"a_r" << " " << a_r << endl;
                cout <<"R0" << " " << R0 << endl;
        cout <<"R_sigmas " << " " << R_sigmas << endl;
                cout <<"h" << " " << h << endl;
                cout <<"M" << " " << M << endl;
                cout <<"t0" << " " << t0 << endl;
                cout <<"k" << " " << k << endl;
                cout <<"L_inf" << " " << L_inf << endl;
                cout <<"a_LW" << " " << a_LW << endl;
                cout <<"b_LW" << " " << b_LW << endl;
                cout <<"m50" << " " << m50 << endl;
                cout <<"m_slope" << " " << m_slope << endl;
                cout <<"sf50" << " " << sf50 << endl;
                cout <<"sf_slope" << " " << sf_slope << endl;
                cout <<"ss50" << " " << ss50 << endl;
                cout <<"ss_slope" << " " << ss_slope << endl;
        cout <<"SPR_lim" << " " << SPR_lim << endl;
                cout <<"SPR_targ" << " " << SPR_targ << endl;

            }


 END_CALCS

PARAMETER_SECTION


  objective_function_value NLL

        LOCAL_CALCS


    //2 x 2 x 2 = 8 x 2 = 16 x 3 = 48 x 2 x 2


      cnt = 1;
```

```
for(sa=1; sa<=2; sa++)
{
   for(aa=1; aa<=2; aa++)
   {
      for(ps=1; ps<=2; ps++)
      {
         for(rp=1; rp<=2; rp++)
         {
            sa_err_vec(cnt) = sa;
            R_sigma_vec(cnt) = R_sigmas(sa);
            abc_vec(cnt) = aa;
            proj_vec(cnt) = ps;
            rp_vec(cnt) = rp;

            cnt++;


         }
      }
   }
}

  // cout << sa_err_vec << endl;
  // cout << R_sigma_vec << endl;
  // cout << R_type_vec << endl;
  // cout << abc_vec << endl;
  // cout << proj_vec << endl;
  // cout << rp_vec << endl;




            // Calculate the total number of stock assessments

   // Uncertainty in commercial landings and survey estimates
            C_var_obs = log(square(C_pse)+1.);
            C_sigma = sqrt(C_var_obs);
            I_var_obs = log(square(I_pse)+1.);
            I_sigma = sqrt(I_var_obs);

   // How much recruitment vasriability do you want to use?

            // Calculate years for projections

//  Calculate selectivity, maturity and growth params
```

```
      m_slope = 1.0;
      ss_slope = 1.0;
      sf_slope = 1.0;
      k = M / M_k_ratio;
      //cout << m50 << " " << sf50 << " " << ss50 << " " << floor(ss50) << " " << k
<< endl;

                  get_init_vals();
                  get_init_F();
                  get_numbers();
      cout << "Model finished!" << endl;
      exit(0);

         END_CALCS



PROCEDURE_SECTION
FUNCTION get_init_vals

 for(a=a_r; a<=amax; a++)
 {

         // Calc initial abundance
         if(a==a_r) N_init(a) = R0;
         else if(a < amax) N_init(a) = N_init(a-1)*exp(-M);
         else N_init(a) = N_init(a-1)*exp(-M)/(1.-exp(-M));

         // calc age-specific vectors (length, weight,selex,mat)
         L(a) = L_inf*(1.-exp(-k*(a - t0)));
         W(a) = a_LW * pow(L(a),b_LW);
         m(a) = 1. / (1.+exp(-(a-m50)/m_slope));
         ss(a) = 1. / (1.+exp(-(a-ss50)/ss_slope));
         sf(a) = 1. / (1.+exp(-(a-sf50)/sf_slope));

 }
   //cout << ss << endl;
   //cout << sf << endl;
   //cout << m << endl;

   // Calculate S0 and B-H params
         S0 = sum(elem_prod(N_init,elem_prod(m,W)));
         alpha = S0 * (1.-h)/(4.*h*R0);
         beta = (5.*h-1.)/(4.*h*R0);

         //cout << alpha << " " << beta << endl;
```

```
FUNCTION get_init_F

        // These values are all relative
        F_0 = 0.05;
        F_end = 0.6;
        Yr_plat = 18.;
        F_plat = 1;
        F_slope_up = (F_plat - F_0)/(Yr_plat-1.);
        F_slope_down = (F_end - F_plat) / (first_SA_yr - 1.);
      F_int_up = F_0 - F_slope_up * 1.;
        F_int_down = F_plat - F_slope_down * Yr_plat;

        for(t=1; t<=first_SA_yr; t++)
        {
                if(F_shape==0)  // intitial F plateaus
                {
                        if(t <= Yr_plat) relF(t) = F_slope_up * t + F_int_up;
                        else relF(t) = F_plat;
                }

                else if(F_shape==1)
                {
                        if(t <= Yr_plat) relF(t) = F_slope_up * t + F_int_up;
                        else  relF(t) = F_slope_down * t + F_int_down;
        }




        }

        F_init = relF;




FUNCTION get_numbers

        // Remove the output files before running

        if(lh==1)
        {
    system("rm final_assessment_estimates_fast.txt");
         system("rm performance_measures_fast.txt");
```

```
        system("rm terminal_estimates_fast.txt");
    system("rm obs_est_fast.txt");


     ofstream ofs_head("performance_measures_head_fast.txt");
     ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << " R_type" <<"
"<< " SA_interval"<< " " << "data_lag"<< " " <<"acor_switch" <<" "<<"abc_switch"
<< " " << "proj_switch" << " " << "rp_switch" << " "  <<"cr" << " "<<  "eh" << " "
<< "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<  "true_S"
<<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<" est_Sproxy" << "
"<< "true_R"<<" "<< "est_R"<< " " << "F " << " " << "est_F"<< " "<<"Fmsy" << " "
<< "Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<
"C_act"<<  " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<  endl;

   }

        else if(lh==2)
        {
      system("del final_assessment_estimates_medium.txt");
      system("del performance_measures_medium.txt");
      system("del terminal_estimates_medium.txt");
      system("del obs_est_medium.txt");

      ofstream ofs_head("performance_measures_head_medium.txt");
      ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << " R_type" <<"
"<< " SA_interval"<< " " << "data_lag"<< " " <<"acor_switch" <<" "<<"abc_switch"
<< " " << "proj_switch" << " " << "rp_switch" << " "  <<"cr" << " "<<  "eh" << " "
<< "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<  "true_S"
<<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<" est_Sproxy" << "
"<< "true_R"<<" "<< "est_R"<< " " << "F " << " " << "est_F"<< " "<<"Fmsy" << " "
<< "Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<
"C_act"<<  " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<  endl;
        }

        else if(lh==3)
        {
    system("del final_assessment_estimates_slow.txt");
        system("del performance_measures_slow.txt");
    system("del terminal_estimates_slow.txt");
    system("del obs_est_slow.txt");

      ofstream ofs_head("performance_measures_head_slow.txt");
      ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << " R_type" <<"
"<< " SA_interval"<< " " << "data_lag"<< " " <<"acor_switch" <<" "<<"abc_switch"
<< " " << "proj_switch" << " " << "rp_switch" << " "  <<"cr" << " "<<  "eh" << " "
<< "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<  "true_S"
```

```
<<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<" est_Sproxy" << "
"<< "true_R"<<" "<< "est_R"<< " " << "F " << " " << "est_F"<< " "<<"Fmsy" << " "
<< "Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<
"C_act"<<  " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<  endl;
          }

           //system("rm final_assessment_estimates.txt");



   get_brps();

   //sa = 2;
   //R_sigma = R_sigmas(sa);


  // the model run (mr) loop - for assessment error and
  mr_max = 1;
  tvp_int = 0;

  for(mr = 1; mr <= mr_max; mr++)
  {
     if(mr_max==1) // set the model runs
     {
        proj_switch = 1; // set to 0 (no projections) or 1 (projections) over the
assessment interval //changed to 1 and 2 from 0 and 1 Asylvia
        rp_switch = 1;   // set to 1 or 2 (1 = use the estimated recrutiment to calc the
OFL; 2 = use the mean recruitment)
        abc_switch = 1;  // set to 1 or 2 (1 = use the estimated ABC; 2 = use ABC
averaging
        R_sigma = R_sigmas(1); // set to 1 (R_sigma = 0.77) or 2 (= 1.2)
        sa = 1;  // set to 1 or 2 for low (1) or high(2) assessment error
        acor_switch = 0; // keep at 0; this was used for other work
        R_type = 1; // keep at 1 for now.
     }

     else
     {
        proj_switch = proj_vec(mr);
        rp_switch = rp_vec(mr);
        abc_switch = abc_vec(mr);
        R_sigma = R_sigma_vec(mr);
        //R_type = R_type_vec(mr);
        R_type = 1;
        sa = sa_err_vec(mr);
        acor_switch = 0;
```

```
    }

    // this is used for printing out the time-varying parameters - only done under
certain conditions since they don't vary across most scenarios
    //if(mr==1) tvp_int==1;
    //else if(mr==5) tvp_int=1;


   // Loop over different assessment intervals
   for(sa_int = 1; sa_int <= int_runs; sa_int++)
   {
      if(int_runs ==1) sa_int = 1; // pick an interval if only using 1 value

      //SA_interval = 2 * int_vec(sa_int);
      SA_interval =  int_vec(sa_int);   //Asylvia what is the plus one doing, not
running anual assessments as is

      total_SAs = ceil((years - first_SA_yr + 1.) / SA_interval);

      if(total_SAs < 1) total_SAs = 1;


      for(lag_int = 1; lag_int <= lag_runs; lag_int++)
      {
         if(lag_runs == 1)lag_int = 1; // pick a data lag if only using 1 value


         data_lag = lag_vec(lag_int);
         data_yrs = years - first_data_yr - data_lag + 1;

         first_est_yr(lag_int) = first_SA_yr - data_lag;
         final_est_yr(lag_int) = years - data_lag;

         if(proj_switch==1) proj_years = 1 + data_lag;
         else if(proj_switch==2) proj_years = data_lag + SA_interval;

         for(cr = 1; cr <= cr_runs; cr++)
         {
            if(cr_runs == 1) cr = 4; // Use this to override a control rule index
            // cr = 1 -> The ABC = OFL
            // cr = 2 -> MAFMC P* control rule with CV = 0.38
            // cr = 3 -> MAFMC P* control rule with CV = 0.70
            // cr = 4 -> MAFMC P* control rule with CV = 1.0
            // cr = 5 -> Target P* = 0.4 with CV = 0.38
            // cr = 6 -> Target P* = 0.4 with CV = 0.7
            // cr = 7 -> Target P* = 0.4 with CV = 1.0
```

```
// cr = 8 -> Fish at 75% of F_lim


seed = 106;   // Initialize random number seed

if(cr <=3) ps_int = cr;
else if(cr<=6) ps_int = cr - 3;
else ps_int = 1;

for(i=1; i<=i_max; i++) // Loop over model iterations
{

   //cout << "Now on model iteration " << i << endl;

   // fill in the random numbers
   random_number_generator rng(seed);

   // create standard normal deviates
   R_error.fill_randn(rng);
   C_error.fill_randn(rng);
   I_error.fill_randn(rng);
   M_error.fill_randn(rng);
   sf50_error.fill_randn(rng);
   F_error.fill_randn(rng);
   q_error.fill_randn(rng);

   // create uniform deviates
   pC_error_lg.fill_randu(rng);
   pI_error_lg.fill_randu(rng);
   pC_error_sm.fill_randu(rng);
   pI_error_sm.fill_randu(rng);

   Rboom_error.fill_randu(rng);

   //cout << Rboom_error << endl;

   seed+=i; // update seed

   if(i <= i_max/3.) eh=1;
   else if(i <= 2.*i_max/3.) eh=2;
   else eh = 3;

   Fmax = Fmsy * Fmult(eh);


   SA_yr = first_SA_yr;
```

```
SA_num = 1;

for(t=1; t<=years; t++) //Loop over years
{
   //cout << cr << " " << i << " " << t << endl;

   if(t==1)
   {
      M_t(t) = M*exp(M_sigma * M_error(t) - 0.5*square(M_sigma));
      sf50_t(t) = sf50*exp(s_sigma * sf50_error(t) -
0.5*square(s_sigma));
      qt(t) = q_surv;
   }


   else
   {
      M_t(t) = M*exp((M_cor(sa)*M_sigma*M_error(t-1)+square(1-
M_cor(sa))*M_sigma*M_error(t))-0.5*square(M_sigma));
      sf50_t(t) = sf50*exp((s_cor(sa)*s_sigma*sf50_error(t-1)+square(1-
s_cor(sa))*s_sigma*sf50_error(t))-0.5*square(s_sigma));

      qt(t) = qt(t-1) * exp(q_sigma(sa) * q_error(t-1));

   }

   if(M_t(t)< M * rel_M_bounds(1)) M_t(t) = M * rel_M_bounds(1);
   else if(M_t(t) > M * rel_M_bounds(2)) M_t(t) = M *
rel_M_bounds(2);

   if(sf50_t(t)< sf50 * rel_s_bounds(1)) sf50_t(t) = sf50 *
rel_s_bounds(1);
   else if(sf50_t(t) > sf50 * rel_s_bounds(2)) sf50_t(t) = sf50 *
rel_s_bounds(2);



   for(a=a_r; a<=amax; a++) // Loop over ages
   {
      sf_t(t,a) = 1. / (1.+exp(-(a-sf50_t(t))/sf_slope));

      if(t<=a_r)
      {
         N(t,a) = N_init(a);

      }
```

```
                else if(t > a_r)
                {
                   if(a==a_r)
                   {
                      if((t - a_r) < 1)
                      {
                         N(t,a) = N_init(a_r);
                      }
                      else
                      {
                         //if(t==1) N(t,a) = S(t-a_r) / (alpha + beta * S(t-
a_r))*mfexp(R_sigma * R_error(t) - 0.5 * square(R_sigma));

                         if(R_type==1) // no boom recruitment events
                         {
                            N(t,a) =  N(t,a) = S(t-a_r) / (alpha + beta * S(t-a_r))
*exp((R_cor*R_sigma*R_error(t-1)+square(1-R_cor)*R_sigma*R_error(t))-
0.5*square(R_sigma));
                         }

                         else if(R_type==2)
                         {
                            // add if statement here for if some condition is met.

                            if(boom_prob > Rboom_error(t))
                            {
                               N(t,a) =  Rboom_mult * S(t-a_r) / (alpha + beta * S(t-
a_r));

                               Rboom_vec(t) = 1;
                            }
                            else
                            {
                               N(t,a) =  N(t,a) = S(t-a_r) / (alpha + beta * S(t-a_r))
*exp((R_cor*R_sigma*R_error(t-1)+square(1-R_cor)*R_sigma*R_error(t))-
0.5*square(R_sigma));
                               Rboom_vec(t) = 0;
                            }
                         }

                      }
                   }


                else if(a < amax) N(t,a)=N(t-1,a-1)*exp(-Z(t-1,a-1));
```

```
                    else N(t,a) = N(t-1,a-1)*exp(-Z(t-1,a-1)) + N(t-1,a)*exp(-Z(t-
1,a));
                }
            } // end the a loop


if(t <= first_SA_yr) F(t) = relF(t) * Fmax * exp(Finit_sd * F_error(t) - 0.5 *
square(Finit_sd));
            else
            {
               if(acor_switch==0)
               {

                  if(ABC(t) <= 100.0)
                  {
                     C_temp = 100.0;  // ensure a minimum catch
                  }

                  else
                  {
                     C_temp = ABC(t);

                  }

                  // prevent the catch from exceeding the exploitable biomass in
extreme cases
                  if(C_temp >= sum(elem_prod(elem_prod(N(t),W),sf_t(t)))) C_temp
= 0.75 *sum(elem_prod(elem_prod(N(t),W),sf_t(t)));

                  F(t) = estimate_F(N(t), sf_t(t),  W, C_temp, M_t(t));
               }

               else
               {
                  //F(t) = Fmax * exp(Finit_sd * F_error(t) - 0.5 * square(Finit_sd));
                  F(t) = F(t-1) * exp((Fcor*Finit_sd*F_error(t-1)+pow(1-
Fcor,2)*Finit_sd*F_error(t))-0.5*pow(Finit_sd,2));
               }
            }

            Z(t) = M_t(t) + sf_t(t) * F(t);
            S(t) = sum(elem_prod(N(t),elem_prod(m,W)));
            R(t) = N(t,a_r);

            Zmsy = M_t(t) + sf_t(t) * Fmsy;
            Zproxy = M_t(t) + sf_t(t) * F_proxy;
```

```
		C_act(t) = elem_prod(N(t),elem_prod(elem_div(sf_t(t)*F(t),Z(t)),(1.-
exp(-Z(t)))));
		CW_tot(t) = sum(elem_prod(C_act(t),W));
		Flim_OFL(t) =
sum(elem_prod(W,elem_prod(N(t),elem_prod(elem_div(sf_t(t)*F_proxy,Zproxy),(1.
-exp(-Zproxy))))));
		Fmsy_OFL(t) =
sum(elem_prod(W,elem_prod(N(t),elem_prod(elem_div(sf_t(t)*Fmsy,Zmsy),(1.-
exp(-Zmsy))))));

		I_obs_tot(t) = qt(t) * sum(elem_prod(ss, N(t)) * exp(I_sigma(sa) *
I_error(t) - 0.5 * square(I_sigma(sa))));


		if(sa==1)
		{
		   pC_obs(t) = multinomial(ESS(sa), pC_error_lg(t), C_act(t) /
sum(C_act(t)));
		   pI_obs(t) = multinomial(ESS(sa), pI_error_lg(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

		}

		else if(sa==2)
		{
		   pC_obs(t) = multinomial(ESS(sa), pC_error_sm(t), C_act(t) /
sum(C_act(t)));
		   pI_obs(t) = multinomial(ESS(sa), pI_error_sm(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

		}

		C_obs(t) = pC_obs(t) * sum(C_act(t))*exp(C_sigma* C_error(t) - 0.5 *
square(C_sigma));
		I_obs(t) = I_obs_tot(t) * pI_obs(t);

		   //cout << ESS(sa) << endl;
		   //cout << pI_error(t)(1,10) << endl;
		   //cout << "-----------------------------" << endl;
		   //cout << pI_error(t)(1,20) << endl;

		//cout << max_ESS << endl;
		//cout << multinomial(ESS(sa), pI_error(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t)))) << endl;
```

```
            //cout << multinomial(ESS(sa), pI_error(t)(1,ESS(sa)),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t)))) << endl;


            //cout << t << " " << C_obs(t) << endl;

                        // do an assessment
                        if(t==SA_yr)
            {

               system("del em_cf.dat");
               ofstream ofs("em_cf.dat"); // Print out the necessary info to the dat file
               {
ofs << "# life history index" << endl;
                  ofs << lh << endl;
                  ofs << "# control rule index" << endl;
                  ofs << cr << endl;
                  ofs << "# iteration" << endl;
                  ofs << i << endl;
                  ofs << "# stock assessment number" << endl;
                  ofs << SA_num << endl;
                  ofs << "#total number of assessments"<< endl;
                  ofs << total_SAs<< endl;
                  ofs << "# t" << endl;
                  ofs << t-data_lag << endl;
                  ofs << "# first data years" << endl;
                  ofs << first_data_yr << endl;
                  ofs << "#proj_years" << endl;
                  ofs << proj_years << endl;
                  ofs << "#proj_switch" << endl;
                  ofs << proj_switch << endl;
                  ofs << "#rec_proj_switch" << endl;
                  ofs << rp_switch << endl;
                  ofs << "#OFL years" << endl;
                  ofs << SA_interval << endl;
                  ofs << "#maximum age" << endl;
                  ofs << amax << endl;
                  ofs << "# Recruitment age" << endl;
                  ofs << a_r << endl;
                  ofs << "# M" << endl;
                  ofs << M << endl;
                  ofs << "# Weight at age" << endl;
                  ofs << W << endl;
                  ofs << "# Maturity at age" << endl;
                  ofs << m << endl;
                  ofs << "# Observed Catch" << endl;
```

```
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << C_obs(temp) << endl;
}
ofs << "# Observed Index " << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << I_obs(temp) << endl;
}
ofs << "# Catch variation" << endl;
ofs << C_var_obs << endl;
ofs << "# Index variation" << endl;
ofs << I_var_obs(sa) << endl;
ofs << "# Effective sample size" << endl;
ofs << ESS(sa) << endl;
ofs << "# log(True Recruitment)" << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << log(R(temp)) << " " ;
   if(temp==(t-data_lag)) ofs << endl;
}
ofs << "# log(F)" << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << log(F(temp)) << " " ;
   if(temp==(t-data_lag)) ofs << endl;
}
ofs << "# S " << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << S(temp) << " " ;
   if(temp==(t-data_lag)) ofs << endl;
}

ofs << "# log(Init N)" << endl;
for(temp=(a_r+1);temp<=amax;temp++)
{
   ofs << log(N(first_data_yr,temp)) << " " ;
   if(temp==amax) ofs << endl;
}
ofs << "# log(true q)" << endl;
ofs << log(q_surv) << endl;
ofs << "# true a50 in fishery" << endl;
ofs << log(sf50) << endl;
ofs << "# true a50 in survey" << endl;
ofs << log(ss50) << endl;
```

```
      ofs << "# true com fishery selex" << endl;
      ofs << log(sf_slope) << endl;
      ofs << "# true survey selex" << endl;
      ofs << log(ss_slope) << endl;
      ofs << "# SPR limit" << endl;
      ofs << SPR_lim << endl;
      ofs << "# EOF flag" << endl;
      ofs << EOF_flag_1 << endl;
    }

                                //system("em_cf.exe -nohess -nox >>junk.txt");   //
Call the assessment model
    system("em_cf.exe -nohess -nox >junk.txt");

    /* Have it print out
     gmax
     more than 1 year of F, S and R estimates
     sf50
     sf_slope estimates
     ss_50
     ss_slope

    */

    ifstream sao("term_est.txt");
    {

      // when multiple years are read-in, the 1st # read in is the most
recent estimate, and it moves backward in time
      for(fi=1; fi<=SA_interval; fi++)
      {
        sao >> est_termS(t-data_lag - fi +1);
        sao >> est_termR(t-data_lag - fi +1);
        sao >> est_termF(t-data_lag - fi +1);

        if((t-data_lag - fi +1) < first_est_yr(lag_int))
        {
          est_termS(t-data_lag - fi +1) = 0;
          est_termR(t-data_lag - fi +1) = 0;
          est_termF(t-data_lag - fi +1) = 0;
        }


        else if((t-data_lag - fi +1) > final_est_yr(lag_int))
        {
          est_termS(t-data_lag - fi +1) = 0;
```

```
                    est_termR(t-data_lag - fi +1) = 0;
                    est_termF(t-data_lag - fi +1) = 0;
                  }
                  //cout << fi << " " << est_termS(t-data_lag-fi+1) << " " <<
est_termS(t-data_lag) << endl;
               /*
                  }

                  for(fi=1; fi<=SA_interval; fi++)
                  {

                  }
                  for(fi=1; fi<=SA_interval; fi++)
                  {

                  }
                   */


                  sao >> Starg_temp;
                  sao >> Flim_temp;

                  for(fi=1;fi<=SA_interval;fi++)
                  {
                     sao >> OFL_in(fi);

                     if(t+fi <=years)  OFL(t+fi) = OFL_in(fi);


                     //cout << OFL_in(fi) << " " << OFL(t+fi) << " " <<  est_termS(t-
data_lag - fi +1) << " " <<  est_termS(t-data_lag) << " " <<  est_termF(t-data_lag - fi
+1) << " " <<  est_termR(t-data_lag - fi +1) << endl;
                  }


                  sao >> est_sf50_temp;
                  sao >> est_sf_slope_temp;
                  sao >> gmax_temp;

               }



               for(fi = 1; fi<=SA_interval; fi++)
               {
                  est_Flim(t-data_lag - fi +1) = Flim_temp;
```

```
                    est_Starg(t-data_lag - fi +1) = Starg_temp;
                    est_sf50(t-data_lag - fi +1) = est_sf50_temp;
                    est_sf_slope(t-data_lag - fi +1) = est_sf_slope_temp;
                    gmax(t-data_lag-fi+1) = gmax_temp;

                    if((t-data_lag - fi +1) < first_est_yr(lag_int))
                    {
                       est_Flim(t-data_lag - fi +1) = 0;
                       est_Starg(t-data_lag - fi +1) = 0;
                       est_sf50(t-data_lag - fi +1) = 0;
                       est_sf_slope(t-data_lag - fi +1) = 0;
                       gmax(t-data_lag-fi+1) = 0;

                    }

                    else if((t-data_lag - fi +1) > final_est_yr(lag_int))
                    {
                       est_Flim(t-data_lag - fi +1) = 0;
                       est_Starg(t-data_lag - fi +1) = 0;
                       est_sf50(t-data_lag - fi +1) = 0;
                       est_sf_slope(t-data_lag - fi +1) = 0;
                       gmax(t-data_lag-fi+1) = 0;

                    }
                    //cout << t-data_lag + fi -1 << " " << est_Flim(t-data_lag + fi -1) <<
" " << est_Starg(t-data_lag + fi -1) << " " << endl;
                    //cout << t-data_lag + fi -1 << " " << est_sf50(t-data_lag + fi -1) <<
" " << est_sf_slope(t-data_lag + fi -1) << " " << gmax(t-data_lag + fi -1) << endl;
                 }


                 Flim = Flim_temp;
                 Starg = Starg_temp;

                 //Increment year for next stock assessment
                 SA_yr = SA_yr + SA_interval;


                 if(SA_yr < years) tt_max = SA_yr;  //(check to make sure we're not
beyond the                                          years of the
simulation)
                 else tt_max = years;

           //Implement declining P* control rules (in the pstar part of the function)

                 if(cr==1)
```

```
                {
                   pstar_imp = 1.0;

                }

                else if(cr <= 4)
                {
                   if(est_termS(t-data_lag)>Starg) pstar_imp=pstar_table(ps_int,100);
//B>Bmsy                                                                    so
at asymptotic pstar
                   else
                   {
                      t1=(100.*est_termS(t-data_lag)/Starg)-floor(100.*est_termS(t-
data_lag)/Starg);
                      t2=100.*est_termS(t-data_lag)/Starg -t1;

                      pstar_imp=pstar_table(ps_int,t2)*(1.-
t1)+pstar_table(ps_int,t2+1)*(t1);  //B<Bmsy so modify P star

                      //cout << est_termS(t-data_lag) << " " << Starg << " " << t1 << "
"<< t2 << " " << pstar_imp << endl;


                   }

                }

             // Implement fixed P* control rule (but for different CVs)
                else if(cr<=7)
                {
                   pstar_imp=pstar_table(ps_int,100);
                }

                else if(cr==8)
                {
                   pstar_imp = 1.0;
                }

                     /*
                     cout << est_termS(t)/Starg << endl;
                     cout << pstar_imp << endl;
                     */
                     //OFL_buffer = Pstar_C_adjust(CV, pstar_imp,50);  //Calculate
OFL buffer

                     //calculate ABC for each year of the inter assessment period
```

```
                for(tt=(t+1);tt<=tt_max;tt++)
                {


                   if(SA_num==1)
                   {
                      ABC(tt) = abc_mult(abc_switch) * pstar_imp * OFL(tt) + (1. -
abc_mult(abc_switch)) * CW_tot(t);

                   }

                   else
                   {
                      ABC(tt) = abc_mult(abc_switch) * pstar_imp * OFL(tt) + (1. -
abc_mult(abc_switch)) * ABC(t);
                   }

                   //cout << OFL_in << endl;
                   //cout << cr << " " << t << " " << tt  << " " << pstar_imp << " " <<
ABC(tt) << " " << OFL(tt) << endl;
                  // cout << SA_num  << " " << abc_switch << " " <<
abc_mult(abc_switch) << " " << OFL(tt) << " " << ABC(tt)  << endl;

                }

                    // Call function to output stock asessment estimates for each
assessmnet
                                                        //write_SA_estimates();


                   SA_num = SA_num++;


                } // end t==SA_yr

                   // Create a vector called R_ref that is used to calculate S_proxy
                //if(t >= first_data_yr && t <= (years-data_lag)) R_ref(t) = R(t);


             } // end t loop


                /*
                cout << N << endl;
                cout << "------------------------------"<< endl;
                cout << R << endl;
```

```
                cout << "-------------------------------"<< endl;
                cout << est_termR << endl;
                cout << "-------------------------------"<< endl;
                */

                /*
                cout << extract_column(N,a_r) << endl;
                cout << "-------------------------------"<< endl;
                cout << R << endl;
                cout << "-------------------------------"<< endl;
                cout << R_ref << endl;
                cout << data_yrs << endl;
                 */

              //cout << R << endl;
              //cout << R(first_data_yr,years-data_lag) << endl;
              S_proxy = sum(R(first_data_yr,years-data_lag))/data_yrs * SPR_proxy;

            write_final_output();
            if(tvp_int==1) write_time_varying_params();


              cout<<"Sim: " << i << "; CR: " << cr << ";  lag_int: " <<lag_int << ";
SA_int: " << sa_int  << ";  mr: " << mr << endl;

          }//end the i loop

          tvp_int++;

       } // end the cr_loop
     } // end the lag data loop
   } // end the assessment interval loop
  } // end the mr loop



FUNCTION get_brps

  Finc = 0.01;
  minF = 0.0;
  F_iters = 200;

  for(f=1; f<=F_iters; f++)
  {
     F_brp = minF + (f-1) * Finc;
```

```
    for(a=a_r;a<=amax;a++)
    {
     Z_brp(a) = M + sf(a) * F_brp;
     if(a==a_r) N_brp(a) = 1.;
     else if(a < amax) N_brp(a) = N_brp(a-1) * exp(-Z_brp(a-1));
     else N_brp(a) =  N_brp(a-1) * exp(-Z_brp(a-1)) / (1.-exp(-Z_brp(a)));
    }
    /*
    cout << mu_sf << endl;
    cout << Z_brp << endl;
    cout << F_brp << " " << N_brp << endl;
    */
    YPR = sum(elem_prod(N_brp,elem_prod(W,elem_prod(elem_div(sf *
F_brp,Z_brp),(1-exp(-Z_brp))))));
    S_brp = sum(elem_prod(N_brp,elem_prod(m,W)));

    if(f==1) Sbrp_0 = S_brp;
    SPR = S_brp / Sbrp_0;
    R_brp = (S_brp - alpha)/(beta*S_brp);
    Y_brp = YPR * R_brp;

    if(f==1)
    {
      F_proxy = F_brp;
      Fmsy = F_brp;
      MSY = Y_brp;
      Smsy = S_brp * R_brp;
    }
    else
    {
       if(Y_brp > MSY)
       {
         Fmsy = F_brp;
         MSY = Y_brp;
         Smsy = S_brp * R_brp;
       }

       if(SPR >= SPR_lim)
       {
         F_proxy = F_brp;
         SPR_proxy = S_brp;
       }
    }
```

```
    //cout << f <<" " << F_brp << " " <<YPR << " " << S_brp << " " << SPR << " "
<< R_brp << " " << Y_brp << " " << MSY << " " << Fmsy << " " << F_proxy << " "
<< Smsy << endl;
  } // end the f loop



// Estimate the F associated with a particular catch.  This function uses the golden
section search.
FUNCTION double estimate_F(dvector N_vec, dvector s_vec, dvector W_vec,
double Total_C, double Est_M)
        double lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;
        double FL;
        double FU;
        double F1;
        double F2;
        double Est_F;
        double C1;
        double C2;
        double CF1;
        double CF2;
        double CU;
        double C_dev;
        double C_dev_targ;

        FL = 0.;
        FU = 2.5;
        C_dev_targ = 0.01;
        C_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        CU =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*FU,M+s_vec*FU),W_vec),N
_vec),(1.-exp(-M-s_vec*FU))));

                //if the highest est catch is still < the observed value, set F = FU
                if(CU <= Total_C)
                {
                        Est_F = FU;
                }

                else if(CU > Total_C){
```

```
for(int j=1; j<=jmax; j++){

C1 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F1,M+s_vec*F1),W_vec),N_
vec),(1.-exp(-M-s_vec*F1))));
C2 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F2,M+s_vec*F2),W_vec),N_
vec),(1.-exp(-M-s_vec*F2))));

CF2 = square(Total_C - C2);
CF1 = square(Total_C - C1);
C_dev = CF1 + CF2;

if(CF2 > CF1) {
        Est_F = F1;
        FU = F2;
        FL = FL;
        F2 = F1;
        F1 = FL + (1.-lambda)*(FU-FL);
        }

else if(CF1 > CF2) {
        Est_F = F2;
        FU = FU;
        FL = F1;
        F1 = F2;
        F2 = FU - (1.-lambda)*(FU - FL);
        }

else{
        if(C_dev > C_dev_targ){
                Est_F = F1;
                FU = 1.1*FU;
                FL = 0.9 * FL;
                F2 = FU - (1.-lambda)*(FU-FL);
                F1 = FL + (1.-lambda)*(FU-FL);
                }

        else{
                Est_F = F1;
                break;
                }
        }

if(C_dev <= C_dev_targ){
```

```
                                        Est_F = F1;
                                        break;
                                        }

                        //cout <<t << " " << j <<" "<< sum(C_obs(t)) << " " << C1 <<
" " << C2 <<" " << C_dev << " " << Est_F << endl;
                        //cout << sum(N(t))/sum(N_est(t))<<endl;
                        //cout <<j<<" " << sum(C_obs(t)) <<" " <<F1 << " " << C1
<<" " << F2 << " " << C2 << " " << C_dev << endl;

                        }//end for loop
                }

        return(Est_F);


 /*
// Estimate the F resulting from a particular catch
FUNCTION dvariable estimate_F(dvector N_vec, dvector s_vec, dvector W_vec,
dvariable Total_C, prevariable Est_M)
        dvariable lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;
        dvariable FL;
        dvariable FU;
        dvariable F1;
        dvariable F2;
        dvariable Est_F;
        dvariable C1;
        dvariable C2;
        dvariable CF1;
        dvariable CF2;
        dvariable CU;
        dvariable C_dev;
        dvariable C_dev_targ;

        FL = 0.;
        FU = 2.5;
        C_dev_targ = 0.01;
        C_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        CU =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*FU,M+s_vec*FU),W_vec),N
_vec),(1.-exp(-M-s_vec*FU))));
```

```
//if the highest est catch is still < the observed value, set F = FU
if(CU <= Total_C)
{
        Est_F = FU;
}

else if(CU > Total_C){

        for(int j=1; j<=jmax; j++){

                C1 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F1,M+s_vec*F1),W_vec),N_
vec),(1.-exp(-M-s_vec*F1))));
                C2 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F2,M+s_vec*F2),W_vec),N_
vec),(1.-exp(-M-s_vec*F2))));

                CF2 = square(Total_C - C2);
                CF1 = square(Total_C - C1);
                C_dev = CF1 + CF2;

                if(CF2 > CF1) {
                        Est_F = F1;
                        FU = F2;
                        FL = FL;
                        F2 = F1;
                        F1 = FL + (1.-lambda)*(FU-FL);
                        }

                else if(CF1 > CF2) {
                        Est_F = F2;
                        FU = FU;
                        FL = F1;
                        F1 = F2;
                        F2 = FU - (1.-lambda)*(FU - FL);
                        }

                else{
                        if(C_dev > C_dev_targ){
                                Est_F = F1;
                                FU = 1.1*FU;
                                FL = 0.9 * FL;
                                F2 = FU - (1.-lambda)*(FU-FL);
                                F1 = FL + (1.-lambda)*(FU-FL);
                                }
```

```
                                        else{
                                                Est_F = F1;
                                                break;
                                                }
                                        }

                                if(C_dev <= C_dev_targ){
                                        Est_F = F1;
                                        break;
                                        }

                        //cout <<t << " " << j <<" "<< sum(C_obs(t)) << " " << C1 <<
" " << C2 <<" " << C_dev << " " << Est_F << endl;
                        //cout << sum(N(t))/sum(N_est(t))<<endl;
                        //cout <<j<<" " << sum(C_obs(t)) <<" " <<F1 << " " << C1
<<" " << F2 << " " << C2 << " " << C_dev << endl;

                        }//end for loop
                }

        return(Est_F);

      */




FUNCTION dvector multinomial(long SS, dvector rand_uniform, dvector p)

        int min_p;
        int dim_p;

        min_p = p.indexmin();
        dim_p = p.indexmax();

        //define local variables to use in this function
        dvector sample(min_p,dim_p);  //create vector for sample from multinomial
        dvector sample_p(min_p,dim_p);
        double p1;
        double p2;
        int ii;
        int jj;
        sample.initialize();
        rand_uniform=sort(rand_uniform);  //sort random numbers
```

```
        p1=0;
        p2=p(min_p);
        jj=1;
        for (ii=min_p;ii<=dim_p;ii++)
        {
           //cout << "p1=" << p1 << " p2=" << p2 << endl;
           sample(ii)=0;
           //assign random uniform numbers to proper bins for multinomial
           while (jj <= SS && rand_uniform(jj) > p1 && rand_uniform(jj) <= p2)
           {
              sample(ii)++;
              jj++;
           }
           if (ii < dim_p)
           {
              p1=p2;
              p2+=p(ii+1);
           }
        }
        sample_p=sample/sum(sample);

        return(sample_p);


FUNCTION double size(dvector obs)
  return(double(obs.indexmax()-obs.indexmin()+1));

FUNCTION double size(param_init_bounded_dev_vector obs)  //overload size
function
  return(double(obs.indexmax()-obs.indexmin()+1));



FUNCTION write_final_output



        /*   ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << "
R_type" <<" "<< " SA_interval"<< " " << "data_lag"<< " " <<"cr" << " "<<  "eh" <<
" " << "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<
"true_S" <<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<"
est_Sproxy" << " "<< "R"<<" " << "F " << " " << "est_F"<< " "<<"Fmsy" << " " <<
"Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<  "C_act"<<
" " << "C_obs" << " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<
endl;*/
```

```
        if(lh==1)
        {

           ofstream ofs_pm("performance_measures_fast.txt",ios::app);

           for(ttt = 1; ttt<=years; ttt++)
           {
              ofs_pm << lh <<" "<< sa <<" "<< R_sigma << " " << R_type << " " <<
SA_interval << " " << data_lag << " " << acor_switch << " " << abc_switch << " "
<< proj_switch << " " << rp_switch << " " << cr << " " << eh << " " << i << " " << ttt
<< " " << M_t(ttt) << " " << sf50_t(ttt) << " " << qt(ttt) << " " << S(ttt) << " " <<
est_termS(ttt) << " " << Smsy << " " << S_proxy << " " << est_Starg(ttt) << " " <<
R(ttt)<< " " << est_termR(ttt) << " " << F(ttt) << " " << est_termF(ttt) << " " <<
Fmsy << " " << F_proxy << " " <<est_Flim(ttt)<< " "  << OFL(ttt) << " " <<
ABC(ttt) << " " << CW_tot(ttt) << " "  << est_sf50(ttt) << " " << est_sf_slope(ttt) <<
" " << gmax(ttt) << endl;
           }
        }

        if(lh==2)
        {
           ofstream ofs_pm("performance_measures_medium.txt",ios::app);

           for(ttt = 1; ttt<=years; ttt++)
           {
              ofs_pm << lh <<" "<< sa <<" "<< R_sigma << " " << R_type << " " <<
SA_interval << " " << data_lag << " " << acor_switch << " " << abc_switch << " "
<< proj_switch << " " << rp_switch << " " << cr << " " << eh << " " << i << " " << ttt
<< " " << M_t(ttt) << " " << sf50_t(ttt) << " " << qt(ttt) << " " << S(ttt) << " " <<
est_termS(ttt) << " " << Smsy << " " << S_proxy << " " << est_Starg(ttt) << " " <<
R(ttt)<< " " << est_termR(ttt) << " " << F(ttt) << " " << est_termF(ttt) << " " <<
Fmsy << " " << F_proxy << " " <<est_Flim(ttt)<< " "  << OFL(ttt) << " " <<
ABC(ttt) << " " << CW_tot(ttt) << " "  << est_sf50(ttt) << " " << est_sf_slope(ttt) <<
" " << gmax(ttt) << endl;
           }
        }


        if(lh==3)
        {
           ofstream ofs_pm("performance_measures_slow.txt",ios::app);

           for(ttt = 1; ttt<=years; ttt++)
           {
```

```
            ofs_pm << lh <<" "<< sa <<" "<< R_sigma << " " << R_type << " " <<
SA_interval << " " << data_lag << " " << acor_switch << " " << abc_switch << " "
<< proj_switch << " " << rp_switch << " " << cr << " " << eh << " " << i << " " << ttt
<< " " << M_t(ttt) << " " << sf50_t(ttt) << " " << qt(ttt) << " " << S(ttt) << " " <<
est_termS(ttt) << " " << Smsy << " " << S_proxy << " " << est_Starg(ttt) << " " <<
R(ttt)<< " " << est_termR(ttt) << " " << F(ttt) << " " << est_termF(ttt) << " " <<
Fmsy << " " << F_proxy << " " <<est_Flim(ttt)<< " "  << OFL(ttt) << " " <<
ABC(ttt) << " " << CW_tot(ttt) << " "  << est_sf50(ttt) << " " << est_sf_slope(ttt) <<
" " << gmax(ttt) << endl;
          }
        }


FUNCTION write_time_varying_params

        if(lh==1)
        {
          ofstream tvp("time_varying_params_fast.txt",ios::app);
          {
            for(ttt = 1; ttt<=years; ttt++)
            {
              tvp << sa << " " << i << " " << ttt  << " " <<  M_t(ttt) << " " <<
sf50_t(ttt) << " " <<sf_slope << " " << qt(ttt) << endl;
            }

          }

        }

        if(lh==2)
        {
          ofstream tvp("time_varying_params_medium.txt",ios::app);
          {
            for(ttt = 1; ttt<=years; ttt++)
            {
              tvp << sa << " " << i << " " << ttt  << " " <<  M_t(ttt) << " " <<
sf50_t(ttt) << " " <<sf_slope << " " << qt(ttt) << endl;
            }

          }

        }

        if(lh==3)
        {
          ofstream tvp("time_varying_params_slow.txt",ios::app);
```

```
            {
               for(ttt = 1; ttt<=years; ttt++)
               {
                  tvp << sa << " " << i << " " << ttt  << " " <<  M_t(ttt) << " " <<
sf50_t(ttt) << " " <<sf_slope << " " << qt(ttt) << endl;
               }

            }

         }

//FUNCTION int round_to_whole(double val)
//           return(ceil(val - 0.5));

REPORT_SECTION
```

Code I. 2: Chapter 1 ADMB code- Stock assessment model: The stock assessment model was called at regular intervals to estimate stock biomass and reference points for management using a statistical catch-at-age model.

```
//John Wiedenmann and Andrea Sylvia
//
//
//

DATA_SECTION

  //
**********************************************************************
*****
  //  DATA READ IN FROM  em_cf.dat
  //
**********************************************************************
*****

  init_int lh                                       // life history index
  init_int cr
  init_int iter                                     // which iteration we are on in the
assessment model
  init_int sa_num                                   // which stock assessment we are
conducting (within each iteration)
  init_int final_sa                         // how many assessments are to be done?
  init_int years                            // # of years in the model
  init_int first_data_yr                    // first year of data collection
  init_int proj_years
  init_int proj_switch
  init_int rp_switch
  init_int OFL_years
 // !! cout<<OFL_years<<endl;
 // !! exit(1);
  init_int amax                             // # of age classes in model
  init_int a_r                              // # age at recruitment to pop
  init_number M
                  // Natural mortality
  init_vector W(a_r,amax)                   // Weight-at-age
  init_vector m(a_r,amax)                   // maturity-at-age

  init_matrix Catch(first_data_yr,years,a_r,amax)                   // Observed catch
at age
  init_matrix N_Index(first_data_yr,years,a_r,amax)          // Observed adbundance
at age
  init_number C_var                         // Var. in catch
```

```
 init_number I_var                          // Var. in index
 init_int ESS                               // survey ESS
 init_vector log_R_true(first_data_yr,years)          // True log(recruitments)
 init_vector log_F_true(first_data_yr,years)          // True log(recruitments)
 init_vector S_true(first_data_yr,years)
 init_vector log_Ninit_true(a_r+1,amax)       // True log(initial population
size)
 init_number log_q_true                     // true log(catchability)
 init_number log_sf50_true                  // age @ 50% selectivity in
fishery
 init_number log_ss50_true                  // age @ 50% selectivity in the
survey
 init_number log_sf_slope                   // age @ 50% selectivity in
fishery
 init_number log_ss_slope                   // age @ 50% selectivity in
fishery
 //init_vector sf_true(1,amax)              // true selectivity
 //init_vector ss_true(1,amax)
 init_number SPR_lim
 init_int EOF_flag

 int t                                       // year index
 int tt
 int a
 int p_int                                  // age index
 int SA_interval
 !! SA_interval = OFL_years;

 //!! cout << "2"<< endl;

// Change the phases to 1 for all.
// feed in F values and start pop at true values
// take out variable C_obs and I_obs


PARAMETER_SECTION
 init_bounded_number log_mean_R(0.,20.,1)          //log mean recruitment
 init_bounded_dev_vector log_R_dev(first_data_yr,years,-20.,20.,1)  // log
deviations for mean recruitment

 init_bounded_vector log_Ninit(a_r+1,amax,0.,20.,1)          // mean initial pop
size (ages
 //init_bounded_dev_vector log_Ninit_dev(2,amax,-20.,20.,1)

 init_bounded_number log_mean_F(-10.,5.,1)          //log of mean F
```

```
 init_bounded_dev_vector log_F_dev(first_data_yr,years,-10,10,1)//log deviations
from mean F
 init_bounded_number log_qest(-15.,-5.,1)                    // log(estimated q)

 init_bounded_number log_sf_50(-2,3,1)                  // log(fishery age @ 50%
selectivity)
 init_bounded_number log_sf_slope(-5.,5.,1)             // log(fishery selectivity
slope)
 init_bounded_number log_ss_50(-2.,3,1)                 // log(survey age @ 50%
selectivity)
 init_bounded_number log_ss_slope(-5.,5.,1)             // log(fishery selectivity
slope)

 matrix N_est(first_data_yr,years,a_r,amax)             // Est. abundance at age
 vector log_Rest(first_data_yr,years)
 matrix C_obs(first_data_yr,years,a_r,amax)                  // Est. catch at age
 matrix C_est(first_data_yr,years,a_r,amax)                  // Est. catch at age
 matrix I_obs(first_data_yr,years,a_r,amax)                  // Est. abundance
index
 matrix I_est(first_data_yr,years,a_r,amax)                  // Est. abundance
index
 matrix pI_est(first_data_yr,years,a_r,amax)            // Est. proportion of Index
abundance
 matrix pC_est(first_data_yr,years,a_r,amax)            // Est. proportion of catch
 matrix pI_obs(first_data_yr,years,a_r,amax)            // Proportion of Index
abundance
 matrix pC_obs(first_data_yr,years,a_r,amax)            // Proportion of catch

 vector Ninit(a_r+1,amax)                               // initial pop size (ages 2-
 vector F_est(first_data_yr,years)                      // Estimated F
 vector ESS_C(first_data_yr,years)                      // Effective sample size
for catch at age
 vector ESS_I(first_data_yr,years)                      // Effective sample size
for index at age
 vector log_R(first_data_yr,years)

 vector I_est_tot(first_data_yr,years)                       // Est. abundance
index
 vector C_est_tot(first_data_yr,years)                       // Est. total catch
 vector C_obs_tot(first_data_yr,years)                       // Est. total catch
 vector S_est(first_data_yr,years)                           // Est. spawning
biomass
 vector ss(a_r,amax)                                         // selex at age in
survey
 vector sf(a_r,amax)                                         // selex at age in
fishery
```

```
vector OFL(1,OFL_years)
vector ctarg(1,OFL_years)

vector N_spr(a_r,amax)
number SPR
number Slim
number FL
number FU
number F1
number F2
number Est_F
number SPR_1
number SPR_2
number DF1
number DF2
number D_dev
number D_dev_targ
number SPR_0
number Flim
number Slim_temp
number Flim_temp
number OFL_est
number Ftarg
number mu_Rec

sdreport_number S_term     // terminal spawning biomass

objective_function_value NLL


        LOCAL_CALCS

                if(EOF_flag != 12345)
                {
                        cout << "----------------------------------------"<<endl;
                        cout <<" Data not read in properly to em_cf.tpl !"<< endl;
                        cout << "----------------------------------------"<<endl;
        cout << "# iteration" << endl;
                        cout << iter << endl;
                        cout << "# stock assessment number" << endl;
                        cout << sa_num << endl;
                        cout <<"# end yr "<< endl;
                        cout << years << endl;
                        cout <<"# first year of data  "<< endl;
                        cout << first_data_yr << endl;
```

```
                cout <<" # max age "<< endl;
                cout << amax << endl;
                cout <<" # age at recruitment  "<< endl;
                cout << a_r << endl;
                cout <<" # Nat. mort  "<< endl;
                cout << M << endl;
                cout <<" # Weight "<< endl;
                cout << W << endl;
                cout <<" #  maturity"<< endl;
                cout << m << endl;
                cout <<" # Catch "<< endl;
                cout << Catch << endl;
                cout <<" # N index "<< endl;
                cout << N_Index << endl;
                cout <<" #  Var in C"<< endl;
                cout << C_var << endl;
                cout <<" # Var in I "<< endl;
                cout << I_var << endl;
                cout <<" # ESS "<< endl;
                cout << ESS << endl;
                cout <<" #  log (R)"<< endl;
                cout << log_R_true << endl;
                cout <<" #  log (F)"<< endl;
                cout << log_F_true << endl;
                cout <<" #  log (N init)"<< endl;
                cout << log_Ninit_true << endl;
                cout <<" #  log(q)"<< endl;
                cout << log_q_true << endl;
                cout <<" #  age @ 50% selex in fishery"<< endl;
                cout << log_sf50_true << endl;
                cout <<" #  age at 50% selex in survey"<< endl;
                cout << log_ss50_true << endl;
                cout <<" #  fishery selex"<< endl;
                cout << log_sf_slope << endl;
                cout <<" #  survey selex"<< endl;
                cout << log_ss_slope << endl;
                cout <<" #  EOF flag"<< endl;
                cout << EOF_flag << endl;
                exit(1);
        }

//cout << log_R_true << endl;
//cout << mfexp(log_R_true(years-1,years)) << endl;

ESS_C = ESS;
ESS_I = ESS;
```

```
                log_mean_R=sum(log_R_true)/double(years-first_data_yr+1.);
                log_R_dev = log_R_true-log_mean_R;
                log_qest = log_q_true;
                //log_mean_F = -0.35;
        log_mean_F = sum(log_F_true)/double(years-first_data_yr+1.);
                log_F_dev = log_F_true-log_mean_F;
        log_Ninit = log_Ninit_true;

                //log_sf_50 = log(sf50_true);
                //log_ss_50 = log(ss50_true);

                log_sf_50 = log_sf50_true;
                log_ss_50 = log_ss50_true;



                get_data();



        END_CALCS



PROCEDURE_SECTION

        get_selectivities();
        get_estimates();
        evaluate_objective_function();
        //cout << Flim << " " << OFL_est << " " << mfexp(log_F_est(years)) << " "
<< sum(Catch(years)) << endl;


FUNCTION get_selectivities

  for(a=a_r; a<=amax; a++)
   {
      ss(a) = 1. / (1.+exp(-(double(a)-mfexp(log_ss_50))/mfexp(log_ss_slope)));
      sf(a) = 1. / (1.+exp(-(double(a)-mfexp(log_sf_50))/mfexp(log_sf_slope)));

        }
```

```
FUNCTION get_data
        for(t=first_data_yr;t<=years;t++)
   {
                C_obs(t) = Catch(t);
                I_obs(t) = N_Index(t);
                //log_R(t) = log_R_true(t);
      C_obs_tot(t) = sum(elem_prod(C_obs(t),W));
      pI_obs(t) = I_obs(t) / sum(I_obs(t));
                pC_obs(t) = C_obs(t) / sum(C_obs(t));
        }

FUNCTION get_estimates


        F_est=exp(log_mean_F+log_F_dev);
        log_Rest=(log_mean_R+log_R_dev);

   N_est(first_data_yr,a_r) = exp(log_Rest(first_data_yr));
   //cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   N_est(first_data_yr)((a_r+1),amax) = exp(log_Ninit);
    //  cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   //exit(1);
   //cout << exp(log_Ninit) << endl;

   S_est(first_data_yr) = sum(elem_prod(elem_prod(N_est(first_data_yr),m),W));
   C_est(first_data_yr) =
elem_prod(N_est(first_data_yr),elem_prod(elem_div(sf*F_est(first_data_yr),M+sf*F
_est(first_data_yr)),(1.-exp(-M-sf*F_est(first_data_yr)))));
   I_est(first_data_yr) = exp(log_qest) * elem_prod(ss, N_est(first_data_yr));
   C_est_tot(first_data_yr) = C_est(first_data_yr) * W;
   pI_est(first_data_yr) =  I_est(first_data_yr) / sum(I_est(first_data_yr));
   pC_est(first_data_yr) = C_est(first_data_yr) / sum(C_est(first_data_yr));


   for(t=first_data_yr+1; t<=years; t++){

      for(a=a_r; a<=amax; a++)
      {
         if(a==a_r) N_est(t,a) = exp(log_Rest(t));
         else
         {
            //if(t==first_data_yr) N_est(t,a) = Ninit(a);
            //else
            {
               if(a < amax) N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1));
```

```
            else  N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1)) + N_est(t-
1,a)*exp(-M - sf(a)*F_est(t-1));

          }

        }


      }

          S_est(t) = sum(elem_prod(elem_prod(N_est(t),m),W));
          C_est(t) =
elem_prod(N_est(t),elem_prod(elem_div(sf*F_est(t),M+sf*F_est(t)),(1.-exp(-M-
sf*F_est(t)))));
          I_est(t) = exp(log_qest) * elem_prod(ss, N_est(t));
          C_est_tot(t) = sum(elem_prod(C_est(t),W));

          pI_est(t) =  I_est(t) / sum(I_est(t));
          pC_est(t) = C_est(t) / sum(C_est(t));
        }
     //cout << N_est << endl;
     //exit(1);
       S_term = S_est(years);



FUNCTION evaluate_objective_function


        NLL = lognorm_nll(C_obs_tot, C_est_tot, C_var) +
lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) +
          multinom_nll(pC_obs, pC_est, ESS_C) + multinom_nll(pI_obs, pI_est,
ESS_I);
     //cout << "neg LL 1 " << lognorm_nll(C_obs_tot, C_est_tot, C_var) << endl <<
"neg LL 2 " << lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) << endl
      //    << "neg LL 3 " <<  multinom_nll(pC_obs, pC_est, ESS_C) << endl << "neg
LL 4 " <<  multinom_nll(pI_obs, pI_est, ESS_I) << endl;

        // lognormal NLL = n*ln(sigma)+0.5∑[(ln(obs)-ln(est))^2 / sigma^2]
        // where sigma^2 = ln(CV^2+1)
        // multionomial NLL = -ESS * ∑obs_p * ln(est_p)

FUNCTION get_spr_brps

        double lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;
```

```
        dvariable SPR_targ;
        SPR_targ = 0.35;
   //SPR_targ = SPR_lim;
        FL = 0.;
        FU = 2.;
        D_dev_targ = pow(0.0005,2);
        D_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        N_spr(a_r) = 1.;

        for(a = a_r+1; a <=amax; a++)
   {
                if(a<amax) N_spr(a) = N_spr(a-1) * exp(-M);
      else N_spr(a) = N_spr(a-1) * exp(-M) / (1.-exp(-M));
   }

   //N_spr(amax) += (1.-exp(-M));

        SPR_0 = sum(elem_prod(W,elem_prod(N_spr,m)));

        for(int j=1; j<=jmax; j++){

                for(a = a_r+1; a <=amax; a++)
      {
        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F1);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1) * F1) / (1.-exp(-M -
sf(amax)*F1));
      }


      SPR_1 = sum(elem_prod(W,elem_prod(N_spr,m)));

                //cout << "1" << " " << a <<" " <<  N_spr(a) << endl;


                for(a = a_r+1; a <=amax; a++)
      {
                        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2) / (1.-exp(-M -
sf(amax)*F2));
        //cout << "1" << " " << a <<" " <<  " " << sf(a) << " " << N_spr(a) << endl;
```

```
}

//cout <<F2 << " " <<  N_spr << endl;

//N_spr(amax) /= (1.-exp(-M -sf(amax)*F2));

//cout << "2" << " " << a <<" " <<  N_spr(a) << endl;

SPR_2 = sum(elem_prod(W,elem_prod(N_spr,m)));


                DF2 = pow(SPR_targ - SPR_2/SPR_0,2);
                DF1 = pow(SPR_targ - SPR_1/SPR_0,2);

                D_dev = DF1 + DF2;

                        if(DF2 > DF1) {
                                Est_F = F1;
                                SPR = SPR_1;
                                FU = F2;
                                FL = FL;
                                F2 = F1;
                                F1 = FL + (1.-lambda)*(FU-FL);
                                }

                        else if(DF1 > DF2) {
                                Est_F = F2;
                                SPR = SPR_2;
                                FU = FU;
                                FL = F1;
                                F1 = F2;
                                F2 = FU - (1.-lambda)*(FU - FL);
                                }

                        else{
                                if(D_dev > D_dev_targ){
                                        Est_F = F1;
                                        SPR = SPR_1;
                                        FU = 1.1*FU;
                                        FL = 0.9 * FL;
                                        F2 = FU - (1.-lambda)*(FU-FL);
                                        F1 = FL + (1.-lambda)*(FU-FL);
                                        }

                                else{
                                        Est_F = F1;
```

```
                        break;
                        }
                }

        if(D_dev <= D_dev_targ){
                if(DF1 < DF2){
                        Est_F = F1;
                        SPR = SPR_1;
                        }
                else{
                        Est_F = F2;
                        SPR = SPR_2;
                        }
                break;
                }

        //cout <<j<<" " <<F1 << " " << SPR_1/SPR_0 <<" " << F2 <<
" " << SPR_2/SPR_0 << " " << Est_F << endl;

        }//end for loop

        Flim = Est_F;
        Slim = SPR * sum(mfexp(log_Rest))/size(log_Rest);



FUNCTION dvariable ofl_calc(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Est_Flim)

        dvar_vector N_ofl(a_r,amax);
        dvariable OFL;

        N_ofl(a_r) = Rec;

        for(a=a_r+1; a<=amax; a++)
        {
                if(a < amax) N_ofl(a) =  N_term(a-1) * exp(-M - sel_f(a-1) * F_cur);
      else N_ofl(a) = N_term(a-1) * exp(-M - sel_f(a-1) * F_cur) + N_term(a) * exp(-
M - sel_f(a) * F_cur);
   }

        OFL =
sum(elem_prod(W,elem_prod(N_ofl,elem_prod(elem_div(sel_f*Est_Flim,M+sel_f*
Est_Flim),(1.-mfexp(-M-sel_f*Est_Flim))))));
        /*
        cout << F_cur << endl;
```

```
        cout << Est_Flim << endl;
        cout << M << endl;
        cout << W << endl;
        cout << sel_f << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << OFL << endl;
        */
        return(OFL);

      cout << "OFL"<<OFL<<endl;

//FUNCTION dvariable ofl_proj(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Frec_cur, dvariable Est_Flim, dvariable Est_Ftarg,
data_int pyrs, int proj_switch)
FUNCTION dvar_vector ofl_proj(dvar_vector N_term, named_dvar_vector sel_f,
prevariable Rec, prevariable F_cur, prevariable Est_Flim, data_int pyrs, data_int
proj_switch, data_int rp)
     //changed to dvar_vector ASylvia
   //cout << pyrs << " " << proj_years << endl;

        dvar_matrix N_ofl(1,proj_years,a_r,amax);
        dvar_vector OFL_vec(1,proj_years);
        dvar_vector ctarg_vec(1,proj_years);
   dvar_vector Zt(a_r,amax);
   dvar_vector Z_cur(a_r,amax);
        dvar_vector Z_lim(a_r,amax);
        dvar_vector Z_targ(a_r,amax);
   dvariable OFL_cur;

        Z_cur =  M + sel_f * F_cur;
        Z_lim =  M + sel_f * Est_Flim;
      //cout<<"1"<<endl;
        //Z_targ =  M + sel_f * Est_Ftarg;

   //cout << Z_cur << endl;
   //cout << Z_lim << endl;
   //cout << Z_targ << endl;
   //cout << 3 << endl;
   //OFL_cur =
sum(elem_prod(W,elem_prod(N_term,elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
   //cout << N_term << endl;
   // replace the terminal estimate of recruitment with the mean value to minimize the
effects of uncertainty
```

```
   if(rp>=2)
   {
      N_term(a_r) = Rec;
   }


   //cout << "pyrs: " << pyrs << endl;
   for(t = 1; t<=pyrs; t++)
   {
                  N_ofl(t,a_r) = Rec;


         for(a=a_r+1; a<=amax; a++)
         {

                        if(t==1)
                        {
                              Zt = Z_cur;
                        if(a < amax) N_ofl(t,a) =  N_term(a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_term(a-1) * exp(-Zt(a-1)) +  N_term(a) *
exp(-Zt(a));
                        }
                        else
                        {
                        Zt = Z_lim;
                        if(a < amax) N_ofl(t,a) =  N_ofl(t-1,a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_ofl(t-1,a-1) * exp(-Zt(a-1)) +  N_ofl(t-1,a)
* exp(-Zt(a));
                        }
         }


                  OFL_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
                  //ctarg_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Ftarg,Z_targ),
(1.-mfexp(-Z_targ))))));
      //cout << t << N_ofl(t) << endl;
      //cout << Est_Flim << " " << OFL_vec(t) << endl;
   }
      //cout <<  N_ofl << endl;
      //cout << "est_flim: " << Est_Flim << " OFL_vec " << OFL_vec << endl;

    // cout<<"2"<<endl;
         p_int = OFL_years;
```

```
        //cout<<proj_years<<endl;
        for(t = 1; t<= OFL_years; t++)
        {
       //cout<<"p_int "<<p_int<<endl;
                if(proj_switch <=1)
                {
                        OFL(t) = OFL_vec(pyrs);
                        //ctarg(t) = ctarg_vec(pyrs);
                }
                else if(proj_switch==2)
                {
                        OFL(t) = OFL_vec(p_int);
                        //ctarg(t) = ctarg(p_int);
                }

                p_int--;
        }
  //    cout<<"3"<<endl;
        //cout << proj_switch << " " << proj_years << " " << OFL << endl;


        /*
        cout << Est_Flim << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << sum(elem_prod(elem_prod(N_ofl,m),W))<< " " << S_est(years) <<
endl;
        cout << sum(elem_prod(W,N_ofl)) <<  endl;
        cout << (1.-exp(-M-sel_f*Est_Flim)) << endl;
        cout << elem_div(sel_f*Est_Flim,M+sel_f*Est_Flim) << endl;
        cout << sf << endl;
        cout << sf_true << endl;
        cout << N_term << endl << N_ofl << endl;
        */
        //cout << OFL << endl;
        return(OFL);  //turned back into code ASylvia


FUNCTION write_term_output
   ofstream sae("term_est.txt");
   {
      for(tt = 0; tt <=(SA_interval-1);tt++)
      {
         sae << S_est(years-tt) << " " <<  exp(log_Rest(years-tt)) << " " <<
F_est(years-tt) << endl;
```

```
    }
    sae <<Slim<< endl;
    sae <<Flim<< endl;
    sae <<OFL<< endl;
    sae << mfexp(log_sf_50) << " " << mfexp(log_sf_slope) << endl;
    sae << objective_function_value::gmax << endl;
  }


FUNCTION write_final_estimates

          if(lh==1)
          {
               ofstream ofs_fe("final_assessment_estimates_fast.txt",ios::app);
               {
               if(iter==1)
               {
             ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
               }

                    for(t=first_data_yr;t<=years;t++)
                {

                         ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
               }

          }


        if(lh==2)
          {
               ofstream ofs_fe("final_assessment_estimates_medium.txt",ios::app);
               {
               if(iter==1)
               {
             ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
               }
```

```
                    for(t=first_data_yr;t<=years;t++)
                {

                            ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
                }

            }

          if(lh==3)
            {
                ofstream ofs_fe("final_assessment_estimates_slow.txt",ios::app);
                {
                if(iter==1)
                {
              ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                }

                    for(t=first_data_yr;t<=years;t++)
                {

                            ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
                }

            }


FUNCTION dvariable lognorm_nll(dvar_vector obs, dvar_vector est, data_number
var)    //Function to calculate a lognormal negative log likelihood
  dvariable L;
  L=norm2(log(obs+0.1)-log(est+0.1));
  L = 0.5*(size(obs))*log(var)+0.5*L/var;



  //cout << endl << L << endl<< endl;
  return(L);
```

```
FUNCTION dvariable multinom_nll(dvar_matrix obs_p, dvar_matrix est_p,
dvar_vector eff_size)            // multionomial NLL = -ESS * ∑obs_p * ln(est_p)
        dvariable ML;
        ML = 0.0;
   // make this sum and not rowsum
   // -eff_size change to a value and then change rowsum to sum
        ML = -eff_size * rowsum(elem_prod(obs_p,log(est_p+.001)));

   //cout << ML << endl;
   return(ML);

FUNCTION double size(dvar_vector obs)
// function to return the number of elements in vector
  return(double(obs.indexmax()-obs.indexmin()+1));

FINAL_SECTION


   get_spr_brps();
  // cout<<"1.0"<<endl;
   if(cr <=7) Ftarg = Flim;
   else if(cr==8) Ftarg = 0.75*Flim;

   //cout << rp_switch << endl;
   //cout << mfexp(log_Rest) << endl;
   //cout << mfexp(log_Rest(years-5,years-1)) << endl;

   if(rp_switch<=2) mu_Rec = sum(mfexp(log_Rest))/size(log_Rest);
   else if(rp_switch==3) mu_Rec = sum(mfexp(log_Rest(years-5,years-1)))/5.;
  // cout<<"1.2"<<endl;


//ofl_calc(N_est(years),sf,sum(mfexp(log_Rest))/size(log_Rest),F_est(years),Flim);
   //
cout<<"N_est"<<N_est(years)<<endl<<"sf"<<sf<<endl<<"mu_Rec"<<mu_Rec<<en
dl<<"F_est"<<F_est(years)<<endl<<"Ftarg"<<Ftarg<<endl<<"proj_years"<<proj_ye
ars<<endl<<"proj_switch"<<proj_switch<<endl<<"rp_switch"<<rp_switch<<endl;
   // cout<<"proj_years " << proj_years<<endl;

OFL=ofl_proj(N_est(years),sf,mu_Rec,F_est(years),Ftarg,proj_years,proj_switch,rp_
switch);
    //added OFL= in front of ofl_proj function
   // cout<<"1.1"<<endl;
   write_term_output();
   //write_r_est();
   //if(sa_num==final_sa) write_final_estimates();
```

```
  // get rid of output here, and combine with other files
  /*
  ofstream brp("brp.txt");
  {
     brp << Flim << endl;
     brp << Slim << endl;
  }
  */

  //cout << "estimates" << endl;
  //cout << Flim << " " << Slim << endl;

  //cout << OFL_est << endl;
  //write_ofl();
        //write_sa_output();
        //write_r_est();
        //write_selex();

REPORT_SECTION


        //write_obs_est();
        //write_init_N();
        //write_SR_output();
        //cout << "est model done!" << endl;
        //write_ssb_output();


RUNTIME_SECTION
  maximum_function_evaluations 4000  // increase the # of evaluations
```

Code III.3: Chapter 2 ADMB Code- Operating model: the operating model represented the true dynamics of the stock using an age-structured population model and implemented the management portion of the simulation by applying the ABC to the stock.

```
// Code created by:
// John Wiedenmann and Andrea Sylvia
// Operating model for the ABC MSE
// Last modified March 10th, 2015

// ############   NOTES  ###################
//
// This file is the operating model. It is the core of the MSE simulation
// and calls additional executables for various purposes.
//
// ########################################

// ========== TO DO ======================
// Add in control rule specifics
//
// Add projections?

  //TOP_OF_MAIN_SECTION
  //time(&start);
  //arrmblsize = 2880117328;
  //gradient_structure::set_GRADSTACK_BUFFER_SIZE(1.e7);
  //gradient_structure::set_CMPDIF_BUFFER_SIZE(1.e7);
  //gradient_structure::set_MAX_NVAR_OFFSET(5000);
  //gradient_structure::set_NUM_DEPENDENT_VARIABLES(5000);

DATA_SECTION


  //
**********************************************************************
*****
  //  DATA READ IN FROM:
  //       • Main parameters are in rec_opmodel.dat
  //       • Life history specific parameters are in
  //                 - slow life history -> slow_lh.dat
  //                 - medium life history -> med_lh.dat
  //       - fast life history -> fast_lh.dat
  //
**********************************************************************
*****
```

```
 init_int i_max           // total iterations
 init_int lag_runs        // # of different data lag and assessment intervals (ex: 3 la
 init_vector lag_vec(1,3)
 init_int int_runs        // # of different
 init_vector int_vec(1,3)  //
 init_int m_runs          // # of sa error and recruitment runs
 init_int years                  // years in the model
 init_int first_data_yr // year when data first collected
 init_int first_SA_yr            // year when assessment first done
 init_int cr_runs                        // control rule runs
 init_int lh                             // life history index
 init_int pd_switch           //partial data switch edited:ALS
 init_vector amax_vec(1,3)                       // age classes in model
 init_vector h_vec(1,3)
 init_vector M_vec(1,3)
 init_number t0                          // age at length = 0
 init_number L_inf                       // L infinity
 init_number a_LW                        // L-W scale
 init_number b_LW                        // L-W exponent
 init_number SPR_lim       // SPR limit (i.e. F35%)
 init_number SPR_targ      // SPR target
 init_number q_surv       // catchability of the survey
 init_vector q_sigma(1,2)
 init_vector ESS(1,2)             // effective sample size
 init_number terminal_ESS       //edited:ALS
 init_int F_shape          // controls the shape of the initial F pattern
 init_vector Fmult(1,3)     //
 init_number Finit_sd       // s.d. of initial F
 init_number M_sigma        // variability in M
 init_vector M_cor(1,2)     // autocorrelation
 init_vector rel_M_bounds(1,2)  // relative bounds on sf50
 init_number s_sigma        // variability in sf50
 init_vector s_cor(1,2)         // autocorrelation in sf50
 init_vector rel_s_bounds(1,2)         // relative bounds on sf50
 init_number C_pse         // PSE in catch estimates
 init_vector I_pse(1,2)     // PSE in index
 init_int acor_switch
 init_number Fcor
 init_number R0                          // unfished recruitment
 init_number R_cor
 init_vector R_sigmas(1,2)    // values for S-R variability
 init_number Rboom_yrs_mult // avg number of years for a boom recruitment to
occur   init_number Rboom_mult     // dtermines the size of the boom recruitment
event
```

```
 init_number M_k_ratio      // ratio of M / k
 init_number M_mat_mult     // product of M and age at maturity
 init_number surv_sel_mult  // multiplier relating survey a50 to fishery a50
 init_vector abc_mult(1,2)  // multiplier ofr calculating the ABC
 init_matrix pstar_table(1,3,1,100)  //table of P star values

 init_int EOF_flag_1                    // end of file flag

 //!!ad_comm::change_datafile_name("med_lh.dat");
 //init_int a_r                         // age at recruitment to pop
 //init_number h                              // steepness
 //init_number M                              // Natural mortality
 //init_int EOF_flag_2                  // end of file flag


 int a_r
 int amax
 int i                                  // integer index
 int t                                  // year index
 int tt                           // year index used to set future catches
 int tt_max                       // final year catches are projected (following and
assessment)
 int ttt                          // year index used for printing out results
 int temp                 // int used for printing out data to be read into assessment
model
 int a                                  // age index
 int sa
 int seed                               // random number seed
 int last_SA_yr                   // year when last assessment is done
 int SA_yr                              // year when an assessment is to be conducted
 int SA_num                       // counter of the number of assessments
 int total_SAs        // total nuber of assessment
 int file_iters           // number of data points read in for a particular variable
 int fi                           // index used for reading in output from assessment
 int f              // index for calculating lognormal densitites
 int xmax
 int F_iters          // iterations for the
 int fm
 int eh                                 // index for different exploitation histories
 int rv                                 // index for recrutiment variability
 int iv                                 // index for survey index variability
 int RR
 int r
 int rr
 int cnt
 int lag_int
```

```
 int sa_int
 int mr
 int mr_max
 int cr
 int ps_int
 int data_yrs
 int proj_years
 int max_ESS
 int min_ESS
 int R_type
 int tvp_int
 int rp_switch
 int proj_switch
 int abc_switch
 int aa
 int ps
 int rp
 int max_SA_interval
 int SA_interval
 int data_lag
 int mlv //ASylvia  maximum lag

 !! max_ESS = ESS(1);
 !! min_ESS = ESS(2);
 !! max_SA_interval = int_vec(6)+1;  //Asylvia added +1
 !! mlv=max(lag_vec);     //added by Asylvia
 //!! cout << max_SA_interval << endl;
 //!! data_yrs = years - first_data_yr - data_lag + 1;

 // Vectors and matrices used to generate random variables
 vector R_error(1,years)                                    // recruitment deviations
 vector Rboom_error(1,years)                                // recruitment deviations
 vector C_error(1,years)              // Error in observed catch
 vector I_error(1,years)                                    // Abundance index error
 vector M_error(1,years)                                    // M variability
 vector q_error(1,years)                                    // M variability
 vector sf50_error(1,years)                    // variability in age @ 50% selex in
fishery
 matrix pC_error_lg(1,years,1,max_ESS)    // rv's used to calculate multinomial r.v.
for catch
 matrix pI_error_lg(1,years,1,max_ESS)     // rv's used to calculate multinomial r.v.
for index
 matrix pC_error_sm(1,years,1,min_ESS)
// rv's used to calculate multinomial r.v. for catch  edited:ALS
 matrix pI_error_sm(1,years,1,min_ESS)     // rv's used to calculate multinomial r.v.
for index
```

```
  matrix pC_error_lg_t(1,years,1,terminal_ESS)// rv's used to calculate multinomial
r.v. for catch
  matrix pI_error_lg_t(1,years,1,terminal_ESS)// rv's used to calculate multinomial
r.v. for index
  matrix pC_error_sm_t(1,years,1,terminal_ESS)
// rv's used to calculate multinomial r.v. for catch  edited:ALS
  matrix pI_error_sm_t(1,years,1,terminal_ESS)
// rv's used to calculate multinomial r.v. for index  edited:ALS


  vector F_error(1,years)
  number Fmax;
  number pstar_imp  //implemented pstar value based on B/Bmsy

  vector sa_err_vec(1,16)
  vector R_sigma_vec(1,16)
  vector R_type_vec(1,16)
  vector proj_vec(1,16)
  vector abc_vec(1,16)
  vector rp_vec(1,16)

  vector first_est_yr(1,lag_runs)
  vector final_est_yr(1,lag_runs)

  vector est_Flim(1,years)
  vector est_Starg(1,years)
  vector est_termR(1,years)
  vector est_termS(1,years)
  vector est_termF(1,years)
  vector est_OFL(1,years)
  vector OFL_in(1,max_SA_interval)

  //
  number h
  number M
  number ss50
  number sf50
  number m50
  number k
  number ss_slope
  number sf_slope
  number m_slope
  number boom_prob

  !! amax = amax_vec(lh);
  !! h = h_vec(lh);
```

```
!! M = M_vec(lh);
!! m50 = M_mat_mult / M;
!! sf50 = m50;
!! ss50 = sf50 * surv_sel_mult;
!! a_r = floor(ss50);
!! if(a_r < 1) a_r = 1;
!! boom_prob = 1.0 / (amax * Rboom_yrs_mult);

//Previously from the Parameter Section
number S0                       // Unfished spawning biomass
number alpha                    // stock-recruit alpha (beverton-holt model)
number beta                     // stock-recruit beta (beverton-holt model)

number C_var_obs      // Variance in commercial catch
number C_sigma        // s.d. of commercial catch
vector I_var_obs(1,2)           // Variance in index
vector I_sigma(1,2)             //  // variability in recruitment deviations
number R_sigma

// These parameters control the initial F
// If F_shape = 0 it plateaus, if = 1 it's dome-shaped
number F_0                      //
number F_end          //
number F_slope_up               //
number F_slope_down   //
number F_plat                   //
number F_int_up                 //
number F_int_down     //
number Yr_plat                  //

// Biological Reference Points
number Flim                     // Limit fishing mortality rate (Fmsy of Fx% proxy)
number Starg                    // target spawning biomass (Smsy)
number Flim_temp                // temp estimate of Flim read in from assessment
number Starg_temp               // temp estimate of Starg read in from assessment
number S_temp
number R_temp
number F_temp
number eq_Fmult
number Fmult_msy
number Fmult_spr
vector Zmsy(a_r,amax)
vector Zproxy(a_r,amax)

number S_ratio
number term_S
```

```
  number OFL_buffer
  number C_temp;

  vector L(a_r,amax)              // Length at age
  vector W(a_r,amax)              // Weight at age
  vector m(a_r,amax)              // maturity at age
  vector ss(a_r,amax)             // selex at age in survey
  vector sf(a_r,amax)       // selex at age in com fishery
  vector M_t(1,years)             // M as a function of t
  vector sf50_t(1,years)          // slectivity as a function of t
  vector mu_sf(a_r,amax)

  matrix N(1,years,a_r,amax)  // Numerical abundance
  matrix Z(1,years,a_r,amax)  // Total mortality
  matrix sf_t(1,years,a_r,amax)          // fishery selectivity

  matrix C_act(1,years,a_r,amax)       // Actual catch at age in fishery
  matrix C_obs(1,years,a_r,amax)       // Obs. catch at age in fishery

  matrix I_obs(1,years,a_r,amax)       // Obs. abundance index at age
  matrix pI_obs(1,years,a_r,amax)      // Proportion of Index abundance
  matrix pC_obs(1,years,a_r,amax)       // Proportion of catch

  vector N_init(a_r,amax)             // Initial abundance
  vector N_0(a_r+1,amax)               // Initial abundance in the assessment
  vector R(1,years)                 // Recruits
  vector Rboom_vec(1,years)           // Recruits
  vector S(1,years)                 // Spawning Biomass by year
  vector relF(1,first_SA_yr)          // relative Fishing mortality
  vector F_init(1,first_SA_yr)         // Fishing mortality
  vector F(1,years)                 // Fishing mortality
  vector Fcom(1,years)                // Fishing mortality
  vector Flim_OFL(1,years)
  vector Fmsy_OFL(1,years)
  vector CW_tot(1,years)              // catch in weight
  vector I_obs_tot(1,years)                                 // Total index
  vector qt(1,years)                                        // Effective sample size for index
at age

  vector est_F(1,years)     // estimated F in fishery
  vector est_S(1,years)           // estimated Spawning biomass
  vector est_R(1,years)                   // estimated Recruitment
  vector ABC(1,years)                     // ABC
  vector OFL(1,years+mlv)                  // OFL ASylvia edit
  vector est_sf50(1,years)
  vector est_sf_slope(1,years)
```

```
vector gmax(1,years)

number est_sf50_temp
number est_sf_slope_temp
number gmax_temp

// Variable used in the calculation of SPR-based BRPs
number mu_sf50
vector N_brp(a_r,amax)
vector Z_brp(a_r,amax)
number F_brp
number S_brp
number SPR
number SPR_proxy
number Sbrp_0
number YPR
number R_brp
number Y_brp

number minF
number maxF
number Finc
number MSY
number Smsy
number Fmsy
number F_proxy
number S_proxy

number Fmax_in
number Fmult_in

//intermediate numbers for calculating Pstar
number t1
number t2


LOCAL_CALCS

        if(EOF_flag_1 != 12345)
        {

                // If the last value read in from the .dat file
                // doesn't match up, print the values to check
                // where the error might be
                cout << "-------------------------------------------" << endl;
```

```
                cout <<" Data not read in properly to abc_opmodel.tpl!"<<
endl;
                cout << "-------------------------------------------" << endl;
                cout << "i_max" <<" " << i_max << endl;
                cout <<"years" << " "<< years << endl;
                cout <<"first_data_yr" << " "  << first_data_yr << endl;
                cout <<"first_SA_yr" << " "  << first_SA_yr << endl;
        cout <<"SA_interval" << " "  << SA_interval << endl;
                cout <<"data_lag" << " " << data_lag << endl;
                cout <<"proj_switch" << " " << proj_switch << endl;
                cout <<"rp_switch" << " " << rp_switch << endl;
                cout <<"cr_runs" << " " << cr_runs << endl;
                cout <<"q_surv" << " " << q_surv << endl;
                cout <<"ESS" << " " << ESS << endl;
                cout <<"F_shape" << " " << F_shape << endl;
                cout <<"EOF_flag_1" << " " << EOF_flag_1 << endl;
                cout << "lh" << " " << lh << endl;
                cout <<"amax" << " " << amax << endl;
                cout <<"a_r" << " " << a_r << endl;
                cout <<"R0" << " " << R0 << endl;
        cout <<"R_sigmas " << " " << R_sigmas << endl;
                cout <<"h" << " " << h << endl;
                cout <<"M" << " " << M << endl;
                cout <<"t0" << " " << t0 << endl;
                cout <<"k" << " " << k << endl;
                cout <<"L_inf" << " " << L_inf << endl;
                cout <<"a_LW" << " " << a_LW << endl;
                cout <<"b_LW" << " " << b_LW << endl;
                cout <<"m50" << " " << m50 << endl;
                cout <<"m_slope" << " " << m_slope << endl;
                cout <<"sf50" << " " << sf50 << endl;
                cout <<"sf_slope" << " " << sf_slope << endl;
                cout <<"ss50" << " " << ss50 << endl;
                cout <<"ss_slope" << " " << ss_slope << endl;
        cout <<"SPR_lim" << " " << SPR_lim << endl;
                cout <<"SPR_targ" << " " << SPR_targ << endl;

        }


 END_CALCS

PARAMETER_SECTION
```

```
objective_function_value NLL

        LOCAL_CALCS


   //2 x 2 x 2 = 8 x 2 = 16 x 3 = 48 x 2 x 2


     cnt = 1;

     for(sa=1; sa<=2; sa++)
     {
        for(aa=1; aa<=2; aa++)
        {
           for(ps=1; ps<=2; ps++)
           {
              for(rp=1; rp<=2; rp++)
              {
                 sa_err_vec(cnt) = sa;
                 R_sigma_vec(cnt) = R_sigmas(sa);
                 abc_vec(cnt) = aa;
                 proj_vec(cnt) = ps;
                 rp_vec(cnt) = rp;

                 cnt++;


              }
           }
        }
     }

       // cout << sa_err_vec << endl;
       // cout << R_sigma_vec << endl;
       // cout << R_type_vec << endl;
       // cout << abc_vec << endl;
       // cout << proj_vec << endl;
       // cout << rp_vec << endl;




              // Calculate the total number of stock assessments

       // Uncertainty in commercial landings and survey estimates
```

```
                C_var_obs = log(square(C_pse)+1.);
                C_sigma = sqrt(C_var_obs);
                I_var_obs = log(square(I_pse)+1.);
                I_sigma = sqrt(I_var_obs);

        // How much recruitment vasriability do you want to use?

                // Calculate years for projections

      //  Calculate selectivity, maturity and growth params

      m_slope = 1.0;
      ss_slope = 1.0;
      sf_slope = 1.0;
      k = M / M_k_ratio;
      //cout << m50 << " " << sf50 << " " << ss50 << " " << floor(ss50) << " " << k
<< endl;
          // cout<<"1"<<endl;
                get_init_vals();
          // cout<<"2"<<endl;
                get_init_F();
          // cout<<"3"<<endl;
                get_numbers();
            //cout<<"4"<<endl;

      cout << "Model finished!" << endl;
      exit(0);

        END_CALCS


PROCEDURE_SECTION




FUNCTION get_init_vals

 for(a=a_r; a<=amax; a++)
 {

        // Calc initial abundance
        if(a==a_r) N_init(a) = R0;
        else if(a < amax) N_init(a) = N_init(a-1)*exp(-M);
        else N_init(a) = N_init(a-1)*exp(-M)/(1.-exp(-M));
```

```
        // calc age-specific vectors (length, weight,selex,mat)
        L(a) = L_inf*(1.-exp(-k*(a - t0)));
        W(a) = a_LW * pow(L(a),b_LW);
        m(a) = 1. / (1.+exp(-(a-m50)/m_slope));
        ss(a) = 1. / (1.+exp(-(a-ss50)/ss_slope));
        sf(a) = 1. / (1.+exp(-(a-sf50)/sf_slope));

}
  //cout << ss << endl;
  //cout << sf << endl;
  //cout << m << endl;

  // Calculate S0 and B-H params
        S0 = sum(elem_prod(N_init,elem_prod(m,W)));
        alpha = S0 * (1.-h)/(4.*h*R0);
        beta = (5.*h-1.)/(4.*h*R0);

        //cout << alpha << " " << beta << endl;

FUNCTION get_init_F

        // These values are all relative
        F_0 = 0.05;
        F_end = 0.6;
        Yr_plat = 18.;
        F_plat = 1;
        F_slope_up = (F_plat - F_0)/(Yr_plat-1.);
        F_slope_down = (F_end - F_plat) / (first_SA_yr - 1.);
      F_int_up = F_0 - F_slope_up * 1.;
        F_int_down = F_plat - F_slope_down * Yr_plat;

        for(t=1; t<=first_SA_yr; t++)
        {
                if(F_shape==0)  // intitial F plateaus
                {
                        if(t <= Yr_plat) relF(t) = F_slope_up * t + F_int_up;
                        else relF(t) = F_plat;
                }

                else if(F_shape==1)
                {
                        if(t <= Yr_plat) relF(t) = F_slope_up * t + F_int_up;
                        else  relF(t) = F_slope_down * t + F_int_down;
        }
```

```
        }

        F_init = relF;




FUNCTION get_numbers

        // Remove the output files before running

        if(lh==1)
        {
    system("rm final_assessment_estimates_fast.txt");
         system("rm performance_measures_fast.txt");
         system("rm terminal_estimates_fast.txt");
    system("rm obs_est_fast.txt");


     ofstream ofs_head("performance_measures_head_fast.txt");
     ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << " R_type" <<"
"<< " SA_interval"<< " " << "data_lag"<< " " <<"acor_switch" <<" "<<"abc_switch"
<< " " << "proj_switch" << " " << "rp_switch" << " "  <<"cr" << " "<<  "eh" << " "
<< "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<  "true_S"
<<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<" est_Sproxy" << "
"<< "true_R"<<" "<< "est_R"<< " " << "F " << " " << "est_F"<< " "<<"Fmsy" << " "
<< "Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<
"C_act"<<  " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<  endl;

   }

        else if(lh==2)
        {
      system("del final_assessment_estimates_medium.txt");
      system("del performance_measures_medium.txt");
      system("del terminal_estimates_medium.txt");
      system("del obs_est_medium.txt");

      ofstream ofs_head("performance_measures_head_medium.txt");
      ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << " R_type" <<"
"<< " SA_interval"<< " " << "data_lag"<< " " <<"acor_switch" <<" "<<"abc_switch"
<< " " << "proj_switch" << " " << "rp_switch" << " "  <<"cr" << " "<<  "eh" << " "
<< "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<  "true_S"
```

```
<<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<" est_Sproxy" << "
"<< "true_R"<<" "<< "est_R"<< " " << "F " << " " << "est_F"<< " "<<"Fmsy" << " "
<< "Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<
"C_act"<<  " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<  endl;
          }

          else if(lh==3)
          {
    system("del final_assessment_estimates_slow.txt");
           system("del performance_measures_slow.txt");
    system("del terminal_estimates_slow.txt");
    system("del obs_est_slow.txt");

       ofstream ofs_head("performance_measures_head_slow.txt");
       ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << " R_type" <<"
"<< " SA_interval"<< " " << "data_lag"<< " " <<"acor_switch" <<" "<<"abc_switch"
<< " " << "proj_switch" << " " << "rp_switch" << " "  <<"cr" << " "<<  "eh" << " "
<< "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<  "true_S"
<<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<" est_Sproxy" << "
"<< "true_R"<<" "<< "est_R"<< " " << "F " << " " << "est_F"<< " "<<"Fmsy" << " "
<< "Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<
"C_act"<<  " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<  endl;
          }

           //system("rm final_assessment_estimates.txt");


   //cout << "1.1" << endl;
  // cout<<"3.1"<<endl;
   get_brps();
   // cout<<"3.2"<<endl;
   //sa = 2;
   //R_sigma = R_sigmas(sa);


  // the model run (mr) loop - for assessment error and
  mr_max = 1;
  tvp_int = 0;

  for(mr = 1; mr <= mr_max; mr++)
  {
     if(mr_max==1) // set the model runs
     {
        proj_switch = 1; // set to 0 (no projections) or 1 (projections) over the
assessment interval //changed to 1 and 2 from 0 and 1 Asylvia
```

```
        rp_switch = 1;   // set to 1 or 2 (1 = use the estimated recrutiment to calc the
OFL; 2 = use the mean recruitment)
        abc_switch = 1;  // set to 1 or 2 (1 = use the estimated ABC; 2 = use ABC
averaging
        R_sigma = R_sigmas(1); // set to 1 (R_sigma = 0.77) or 2 (= 1.2)
        sa = 1;  // set to 1 or 2 for low (1) or high(2) assessment error
        acor_switch = 0; // keep at 0; this was used for other work
        R_type = 1; // keep at 1 for now.
    }

    else
    {
        proj_switch = proj_vec(mr);
        rp_switch = rp_vec(mr);
        abc_switch = abc_vec(mr);
        R_sigma = R_sigma_vec(mr);
        //R_type = R_type_vec(mr);
        R_type = 1;
        sa = sa_err_vec(mr);
        acor_switch = 0;
    }

    // this is used for printing out the time-varying parameters - only done under
certain conditions since they don't vary across most scenarios
    //if(mr==1) tvp_int==1;
    //else if(mr==5) tvp_int=1;


   // Loop over different assessment intervals
   for(sa_int = 1; sa_int <= int_runs; sa_int++)
   {
      if(int_runs ==1) sa_int = 2; // pick an interval if only using 1 value

      //SA_interval = 2 * int_vec(sa_int);
      SA_interval =  int_vec(sa_int);   //Asylvia what is the plus one doing, not
running anual assessments as is

      total_SAs = ceil((years - first_SA_yr + 1.) / SA_interval);

      if(total_SAs < 1) total_SAs = 1;


      for(lag_int = 1; lag_int <= lag_runs; lag_int++)
      {
         if(lag_runs == 1)lag_int = 3; // pick a data lag if only using 1 value
```

```
        data_lag = lag_vec(lag_int);
        data_yrs = years - first_data_yr - data_lag + 1;

        first_est_yr(lag_int) = first_SA_yr - data_lag;
        final_est_yr(lag_int) = years - data_lag;

        if(proj_switch==1) proj_years = 1 + data_lag;
        else if(proj_switch==2) proj_years = data_lag + SA_interval;

        for(cr = 1; cr <= cr_runs; cr++)
        {
           if(cr_runs == 1) cr = 4; // Use this to override a control rule index - pick 1-8
based on list below
           // cr = 1 -> The ABC = OFL
           // cr = 2 -> MAFMC P* control rule with CV = 0.38
           // cr = 3 -> MAFMC P* control rule with CV = 0.70
           // cr = 4 -> MAFMC P* control rule with CV = 1.0
           // cr = 5 -> Target P* = 0.4 with CV = 0.38
           // cr = 6 -> Target P* = 0.4 with CV = 0.7
           // cr = 7 -> Target P* = 0.4 with CV = 1.0
           // cr = 8 -> Fish at 75% of F_lim


           seed = 106;   // Initialize random number seed

           if(cr <=3) ps_int = cr;
           else if(cr<=6) ps_int = cr - 3;
           else ps_int = 1;

           for(i=1; i<=i_max; i++) // Loop over model iterations
           {

             // cout << "last_SA_yr " << i << endl;
                // cout << last_SA_yr  << i << endl;
              // fill in the random numbers
              random_number_generator rng(seed);

              // create standard normal deviates
              R_error.fill_randn(rng);
              C_error.fill_randn(rng);
              I_error.fill_randn(rng);
              M_error.fill_randn(rng);
              sf50_error.fill_randn(rng);
              F_error.fill_randn(rng);
              q_error.fill_randn(rng);
```

```
// create uniform deviates
pC_error_lg.fill_randu(rng);
pI_error_lg.fill_randu(rng);
pC_error_sm.fill_randu(rng);
pI_error_sm.fill_randu(rng);     //edited:ALS
pC_error_lg_t.fill_randu(rng);
pI_error_lg_t.fill_randu(rng);
pC_error_sm_t.fill_randu(rng);
pI_error_sm_t.fill_randu(rng);
Rboom_error.fill_randu(rng);

//cout << Rboom_error << endl;

seed+=i; // update seed

if(i <= i_max/3.) eh=1;
else if(i <= 2.*i_max/3.) eh=2;
else eh = 3;

Fmax = Fmsy * Fmult(eh);


SA_yr = first_SA_yr;
SA_num = 1;

for(t=1; t<=years; t++) //Loop over years
{
   //cout << cr << " " << i << " " << t << endl;

   if(t==1)
   {
      M_t(t) = M*exp(M_sigma * M_error(t) - 0.5*square(M_sigma));
      sf50_t(t) = sf50*exp(s_sigma * sf50_error(t) -
0.5*square(s_sigma));
      qt(t) = q_surv;
   }


   else
   {
      M_t(t) = M*exp((M_cor(sa)*M_sigma*M_error(t-1)+square(1-
M_cor(sa))*M_sigma*M_error(t))-0.5*square(M_sigma));
      sf50_t(t) = sf50*exp((s_cor(sa)*s_sigma*sf50_error(t-1)+square(1-
s_cor(sa))*s_sigma*sf50_error(t))-0.5*square(s_sigma));
```

```
            qt(t) = qt(t-1) * exp(q_sigma(sa) * q_error(t-1));

         }

         if(M_t(t)< M * rel_M_bounds(1)) M_t(t) = M * rel_M_bounds(1);
         else if(M_t(t) > M * rel_M_bounds(2)) M_t(t) = M *
rel_M_bounds(2);

         if(sf50_t(t)< sf50 * rel_s_bounds(1)) sf50_t(t) = sf50 *
rel_s_bounds(1);
         else if(sf50_t(t) > sf50 * rel_s_bounds(2)) sf50_t(t) = sf50 *
rel_s_bounds(2);



         for(a=a_r; a<=amax; a++) // Loop over ages
         {
            sf_t(t,a) = 1. / (1.+exp(-(a-sf50_t(t))/sf_slope));

            if(t<=a_r)
            {
               N(t,a) = N_init(a);

            }
            else if(t > a_r)
            {
               if(a==a_r)
               {
                  if((t - a_r) < 1)
                  {
                     N(t,a) = N_init(a_r);
                  }
                  else
                  {
                     //if(t==1) N(t,a) = S(t-a_r) / (alpha + beta * S(t-
a_r))*mfexp(R_sigma * R_error(t) - 0.5 * square(R_sigma));

                     if(R_type==1) // no boom recruitment events
                     {
                        N(t,a) =  N(t,a) = S(t-a_r) / (alpha + beta * S(t-a_r))
*exp((R_cor*R_sigma*R_error(t-1)+square(1-R_cor)*R_sigma*R_error(t))-
0.5*square(R_sigma));
                     }

                     else if(R_type==2)
                     {
```

```
                                    // add if statement here for if some condition is met.

                                    if(boom_prob > Rboom_error(t))
                                    {
                                       N(t,a) =  Rboom_mult * S(t-a_r) / (alpha + beta * S(t-
a_r));

                                       Rboom_vec(t) = 1;
                                    }
                                    else
                                    {
                                       N(t,a) =  N(t,a) = S(t-a_r) / (alpha + beta * S(t-a_r))
*exp((R_cor*R_sigma*R_error(t-1)+square(1-R_cor)*R_sigma*R_error(t))-
0.5*square(R_sigma));
                                       Rboom_vec(t) = 0;
                                    }
                                 }

                              }
                           }


                           else if(a < amax) N(t,a)=N(t-1,a-1)*exp(-Z(t-1,a-1));

                           else N(t,a) = N(t-1,a-1)*exp(-Z(t-1,a-1)) + N(t-1,a)*exp(-Z(t-
1,a));
                        }
                     } // end the a loop


if(t <= first_SA_yr) F(t) = relF(t) * Fmax * exp(Finit_sd * F_error(t) - 0.5 *
square(Finit_sd));
                  else
                  {
                     if(acor_switch==0)
                     {

                        if(ABC(t) <= 100.0)
                        {
                           C_temp = 100.0;  // ensure a minimum catch
                        }

                        else
                        {
                           C_temp = ABC(t);
```

```
                    }

                    // prevent the catch from exceeding the exploitable biomass in
extreme cases
                    if(C_temp >= sum(elem_prod(elem_prod(N(t),W),sf_t(t)))) C_temp
= 0.75 *sum(elem_prod(elem_prod(N(t),W),sf_t(t)));

                    F(t) = estimate_F(N(t), sf_t(t),  W, C_temp, M_t(t));
                  }

                  else
                  {
                    //F(t) = Fmax * exp(Finit_sd * F_error(t) - 0.5 * square(Finit_sd));
                    F(t) = F(t-1) * exp((Fcor*Finit_sd*F_error(t-1)+pow(1-
Fcor,2)*Finit_sd*F_error(t))-0.5*pow(Finit_sd,2));
                  }
                }

                Z(t) = M_t(t) + sf_t(t) * F(t);
                S(t) = sum(elem_prod(N(t),elem_prod(m,W)));
                R(t) = N(t,a_r);

                Zmsy = M_t(t) + sf_t(t) * Fmsy;
                Zproxy = M_t(t) + sf_t(t) * F_proxy;


                C_act(t) = elem_prod(N(t),elem_prod(elem_div(sf_t(t)*F(t),Z(t)),(1.-
exp(-Z(t)))));
                CW_tot(t) = sum(elem_prod(C_act(t),W));
                Flim_OFL(t) =
sum(elem_prod(W,elem_prod(N(t),elem_prod(elem_div(sf_t(t)*F_proxy,Zproxy),(1.
-exp(-Zproxy))))));
                Fmsy_OFL(t) =
sum(elem_prod(W,elem_prod(N(t),elem_prod(elem_div(sf_t(t)*Fmsy,Zmsy),(1.-
exp(-Zmsy))))));

                I_obs_tot(t) = qt(t) * sum(elem_prod(ss, N(t)) * exp(I_sigma(sa) *
I_error(t) - 0.5 * square(I_sigma(sa))));
                 // cout<<"3.3"<<endl;
               //cout<<"1"<<endl;
              // cout<<t<<" "<<SA_yr<<" "<<data_lag<<endl;
               if(pd_switch==1)//edited:ALS
                {
                  if(sa==1)
                  {
```

```
                pC_obs(t) = multinomial(ESS(sa), pC_error_lg(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_lg(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

              }

              else if(sa==2)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_sm(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_sm(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

              }
             }
            if(pd_switch==2)//edited:ALS
             {
              if(sa==1)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_lg(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_lg(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

              }

              else if(sa==2)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_sm(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_sm(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

              }
            }


            if(pd_switch==3)//edited:ALS
            {
              if(sa==1)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_lg(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_lg(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));
```

```
                pC_obs(SA_yr-data_lag)= multinomial(terminal_ESS,
pC_error_lg_t(SA_yr-data_lag), C_act(SA_yr-data_lag) / sum(C_act(t)));
//edited:ALS 02/07
              }

              //I think it is the ESS terminal that is the problem need to figure out what
to do with that I think that the multinomail function needs to be changed

              else if(sa==2)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_sm(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_sm(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));
                pC_obs(SA_yr-data_lag)= multinomial(terminal_ESS,
pC_error_sm_t(SA_yr-data_lag), C_act(SA_yr-data_lag) / sum(C_act(t)));
              }
             }

            if(pd_switch==4)//edited:ALS
            {
              if(sa==1)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_lg(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_lg(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));
                pC_obs(SA_yr-data_lag)= multinomial(terminal_ESS,
pC_error_lg_t(SA_yr-data_lag), C_act(SA_yr-data_lag) / sum(C_act(t)));
                pI_obs(SA_yr-data_lag)= multinomial(terminal_ESS,
pI_error_lg_t(SA_yr-data_lag),elem_prod(ss, N(SA_yr-data_lag))/sum(elem_prod(ss,
N(t))));
                //cout<<pC_obs(SA_yr-data_lag)<<endl;
                //cout<<pI_obs(SA_yr-data_lag)<<endl;
              }

              //I think it is the ESS terminal that is the problem need to figure out what
to do with that I think that the multinomail function needs to be changed

              else if(sa==2)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_sm(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_sm(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));
```

```
                pC_obs(SA_yr-data_lag)= multinomial(terminal_ESS,
pC_error_sm_t(SA_yr-data_lag), C_act(SA_yr-data_lag) / sum(C_act(t)));
                pI_obs(SA_yr-data_lag)= multinomial(terminal_ESS,
pI_error_sm_t(SA_yr-data_lag),elem_prod(ss, N(SA_yr-
data_lag))/sum(elem_prod(ss, N(t))));

               // cout<<pC_obs(SA_yr-data_lag)<<endl;
                //cout<<pI_obs(SA_yr-data_lag)<<endl;

              }

             }

           if(pd_switch==5)//edited:ALS
            {
             if(sa==1)
             {
                pC_obs(t) = multinomial(ESS(sa), pC_error_lg(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_lg(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

              }

              else if(sa==2)
              {
                pC_obs(t) = multinomial(ESS(sa), pC_error_sm(t), C_act(t) /
sum(C_act(t)));
                pI_obs(t) = multinomial(ESS(sa), pI_error_sm(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t))));

              }
            }
             //cout<<"2"<<endl;
               // cout<<"3.5"<<endl;
            // for(t=first_data_yr; t<=years-1; t++)
             //{
             C_obs(t) = pC_obs(t) * sum(C_act(t))*exp(C_sigma* C_error(t) - 0.5 *
square(C_sigma));
             I_obs(t) = I_obs_tot(t) * pI_obs(t);
            // }
             //C_obs(years) = pC_obs(years) * sum(C_act(t))*exp(C_sigma*
C_error(t) - 0.5 * square(C_sigma));
             //I_obs(years) = I_obs_tot(t) * pI_obs(years);

           /*  cout<<"C_obs(t)"<<endl;
```

```
                cout<<C_obs(years-1)<<endl;
                cout<<"I_obs(t)"<<endl;
                cout<<I_obs(years-1)<<endl;

               for(temp=first_data_yr;temp<=(t-data_lag);temp++)
                {
                 cout<<"C_obs(temp)"<<endl;
                 cout<<C_obs(temp)<<endl;
                 cout<<"I_obs(temp)"<<endl;
                 cout<<I_obs(temp)<<endl;
                }
                */
                //cout<<"3"<<endl;
                   //cout << ESS(sa) << endl;
                   //cout << pI_error(t)(1,10) << endl;
                   //cout << "-----------------------------" << endl;
                   //cout << pI_error(t)(1,20) << endl;

                //cout << max_ESS << endl;
                //cout << multinomial(ESS(sa), pI_error(t),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t)))) << endl;
                //cout << multinomial(ESS(sa), pI_error(t)(1,ESS(sa)),elem_prod(ss,
N(t))/sum(elem_prod(ss, N(t)))) << endl;


                //cout << t << " " << C_obs(t) << endl;

                                // do an assessment
                                if(t==SA_yr)
                {

                   system("del em_cf.dat");
                   ofstream ofs("em_cf.dat"); // Print out the necessary info to the dat file
                   {
ofs << "# life history index" << endl;
                      ofs << lh << endl;
                      ofs << "# control rule index" << endl;
                      ofs << cr << endl;
                      ofs << "# iteration" << endl;
                      ofs << i << endl;
                      ofs << "# stock assessment number" << endl;
                      ofs << SA_num << endl;
                      ofs << "#total number of assessments"<< endl;
                      ofs << total_SAs<< endl;
                      ofs << "# t" << endl;
                      ofs << t-data_lag << endl;
```

```
ofs << "# first data years" << endl;
ofs << first_data_yr << endl;
ofs << "#proj_years" << endl;
ofs << proj_years << endl;
ofs << "#proj_switch" << endl;
ofs << proj_switch << endl;
ofs << "#rec_proj_switch" << endl;
ofs << rp_switch << endl;
ofs << "#OFL years" << endl;
ofs << SA_interval << endl;
ofs << "#maximum age" << endl;
ofs << amax << endl;
ofs << "# Recruitment age" << endl;
ofs << a_r << endl;
ofs << "# M" << endl;
ofs << M << endl;
ofs << "# Weight at age" << endl;
ofs << W << endl;
ofs << "# Maturity at age" << endl;
ofs << m << endl;
ofs << "# Observed Catch" << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << C_obs(temp) << endl;
}
ofs << "# Observed Index " << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << I_obs(temp) << endl;
}
ofs << "# Catch variation" << endl;
ofs << C_var_obs << endl;
ofs << "# Index variation" << endl;
ofs << I_var_obs(sa) << endl;
ofs << "# Effective sample size" << endl;
ofs << ESS(sa) << endl;
ofs << "# log(True Recruitment)" << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << log(R(temp)) << " " ;
   if(temp==(t-data_lag)) ofs << endl;
}
ofs << "# log(F)" << endl;
for(temp=first_data_yr;temp<=(t-data_lag);temp++)
{
   ofs << log(F(temp)) << " " ;
```

```
      if(temp==(t-data_lag)) ofs << endl;
    }
    ofs << "# S " << endl;
    for(temp=first_data_yr;temp<=(t-data_lag);temp++)
    {
      ofs << S(temp) << " " ;
      if(temp==(t-data_lag)) ofs << endl;
    }

    ofs << "# log(Init N)" << endl;
    for(temp=(a_r+1);temp<=amax;temp++)
    {
      ofs << log(N(first_data_yr,temp)) << " " ;
      if(temp==amax) ofs << endl;
    }
    ofs << "# log(true q)" << endl;
    ofs << log(q_surv) << endl;
    ofs << "# true a50 in fishery" << endl;
    ofs << log(sf50) << endl;
    ofs << "# true a50 in survey" << endl;
    ofs << log(ss50) << endl;
    ofs << "# true com fishery selex" << endl;
    ofs << log(sf_slope) << endl;
    ofs << "# true survey selex" << endl;
    ofs << log(ss_slope) << endl;
    ofs << "# SPR limit" << endl;
    ofs << SPR_lim << endl;
    ofs << "# EOF flag" << endl;
    ofs << EOF_flag_1 << endl;
   // cout <<"pI_obs"<<endl;
   // cout <<t<<" "<<pI_obs(SA_yr)<<endl;

   // cout<<"4"<<endl;

  }


                        //system("em_cf.exe -nohess -nox >>junk.txt");   //
Call the assessment model
        system("em_cf.exe -nohess -nox >junk.txt");

        /* Have it print out
         gmax
         more than 1 year of F, S and R estimates
         sf50
         sf_slope estimates
```

```
ss_50
ss_slope

*/

ifstream sao("term_est.txt");
{

   // when multiple years are read-in, the 1st # read in is the most
recent estimate, and it moves backward in time
   for(fi=1; fi<=SA_interval; fi++)
   {
      sao >> est_termS(t-data_lag - fi +1);
      sao >> est_termR(t-data_lag - fi +1);
      sao >> est_termF(t-data_lag - fi +1);

      if((t-data_lag - fi +1) < first_est_yr(lag_int))
      {
         est_termS(t-data_lag - fi +1) = 0;
         est_termR(t-data_lag - fi +1) = 0;
         est_termF(t-data_lag - fi +1) = 0;
      }


      else if((t-data_lag - fi +1) > final_est_yr(lag_int))
      {
         est_termS(t-data_lag - fi +1) = 0;
         est_termR(t-data_lag - fi +1) = 0;
         est_termF(t-data_lag - fi +1) = 0;
      }
      //cout << fi << " " << est_termS(t-data_lag-fi+1) << " " <<
est_termS(t-data_lag) << endl;

   }

   /*
   for(fi=1; fi<=SA_interval; fi++)
   {

   }
   for(fi=1; fi<=SA_interval; fi++)
   {

   }
    */
```

```
      sao >> Starg_temp;
      sao >> Flim_temp;

      for(fi=1;fi<=SA_interval;fi++)
      {
         sao >> OFL_in(fi);

         if(t+fi <=years)  OFL(t+fi) = OFL_in(fi);


         //cout << OFL_in(fi) << " " << OFL(t+fi) << " " <<  est_termS(t-
data_lag - fi +1) << " " <<  est_termS(t-data_lag) << " " <<  est_termF(t-data_lag - fi
+1) << " " <<  est_termR(t-data_lag - fi +1) << endl;
      }


      sao >> est_sf50_temp;
      sao >> est_sf_slope_temp;
      sao >> gmax_temp;


   }




   for(fi = 1; fi<=SA_interval; fi++)
   {
      est_Flim(t-data_lag - fi +1) = Flim_temp;
      est_Starg(t-data_lag - fi +1) = Starg_temp;
      est_sf50(t-data_lag - fi +1) = est_sf50_temp;
      est_sf_slope(t-data_lag - fi +1) = est_sf_slope_temp;
      gmax(t-data_lag-fi+1) = gmax_temp;

      if((t-data_lag - fi +1) < first_est_yr(lag_int))
      {
         est_Flim(t-data_lag - fi +1) = 0;
         est_Starg(t-data_lag - fi +1) = 0;
         est_sf50(t-data_lag - fi +1) = 0;
         est_sf_slope(t-data_lag - fi +1) = 0;
         gmax(t-data_lag-fi+1) = 0;

      }
```

```
                    else if((t-data_lag - fi +1) > final_est_yr(lag_int))
                    {
                       est_Flim(t-data_lag - fi +1) = 0;
                       est_Starg(t-data_lag - fi +1) = 0;
                       est_sf50(t-data_lag - fi +1) = 0;
                       est_sf_slope(t-data_lag - fi +1) = 0;
                       gmax(t-data_lag-fi+1) = 0;

                    }
                    //cout << t-data_lag + fi -1 << " " << est_Flim(t-data_lag + fi -1) <<
" " << est_Starg(t-data_lag + fi -1) << " " << endl;
                    //cout << t-data_lag + fi -1 << " " << est_sf50(t-data_lag + fi -1) <<
" " << est_sf_slope(t-data_lag + fi -1) << " " << gmax(t-data_lag + fi -1) << endl;
                 }



                 Flim = Flim_temp;
                 Starg = Starg_temp;

                 //Increment year for next stock assessment
                 SA_yr = SA_yr + SA_interval;


                 if(SA_yr < years) tt_max = SA_yr;  //(check to make sure we're not
beyond the years of the simulation)
                 else tt_max = years;

           //Implement declining P* control rules (in the pstar part of the function)

                 if(cr==1)
                 {
                    pstar_imp = 1.0;

                 }

                 else if(cr <= 4)
                 {
                    if(est_termS(t-data_lag)>Starg) pstar_imp=pstar_table(ps_int,100);
//B>Bmsy so at asymptotic pstar
                    else
                    {
                       t1=(100.*est_termS(t-data_lag)/Starg)-floor(100.*est_termS(t-
data_lag)/Starg);
                       t2=100.*est_termS(t-data_lag)/Starg -t1;
```

```
                        pstar_imp=pstar_table(ps_int,t2)*(1.-
t1)+pstar_table(ps_int,t2+1)*(t1);  //B<Bmsy so modify P star

                        //cout << est_termS(t-data_lag) << " " << Starg << " " << t1 << "
"<< t2 << " " << pstar_imp << endl;


                     }

                  }

               // Implement fixed P* control rule (but for different CVs)
                  else if(cr<=7)
                  {
                     pstar_imp=pstar_table(ps_int,100);
                  }

                  else if(cr==8)
                  {
                     pstar_imp = 1.0;
                  }

                        /*
                        cout << est_termS(t)/Starg << endl;
                        cout << pstar_imp << endl;
                        */
                        //OFL_buffer = Pstar_C_adjust(CV, pstar_imp,50);  //Calculate
OFL buffer

                        //calculate ABC for each year of the inter assessment period
                  for(tt=(t+1);tt<=tt_max;tt++)
                  {


                     if(SA_num==1)
                     {
                        ABC(tt) = abc_mult(abc_switch) * pstar_imp * OFL(tt) + (1. -
abc_mult(abc_switch)) * CW_tot(t);

                     }

                     else
                     {
                        ABC(tt) = abc_mult(abc_switch) * pstar_imp * OFL(tt) + (1. -
abc_mult(abc_switch)) * ABC(t);
                     }
```

```
                    //cout << OFL_in << endl;
                    //cout << cr << " " << t << " " << tt  << " " << pstar_imp << " " <<
ABC(tt) << " " << OFL(tt) << endl;
                   // cout << SA_num  << " " << abc_switch << " " <<
abc_mult(abc_switch) << " " << OFL(tt) << " " << ABC(tt)  << endl;

                }



                //cout << "3.7" << endl;
                      // Call function to output stock asessment estimates for each
assessmnet
                                                                  //write_SA_estimates();


                   SA_num = SA_num++;


                } // end t==SA_yr

                   // Create a vector called R_ref that is used to calculate S_proxy
                //if(t >= first_data_yr && t <= (years-data_lag)) R_ref(t) = R(t);


            } // end t loop


                /*
                cout << N << endl;
                cout << "-------------------------------"<< endl;
                cout << R << endl;
                cout << "-------------------------------"<< endl;
                cout << est_termR << endl;
                cout << "-------------------------------"<< endl;
                */

                /*
                cout << extract_column(N,a_r) << endl;
                cout << "-------------------------------"<< endl;
                cout << R << endl;
                cout << "-------------------------------"<< endl;
                cout << R_ref << endl;
                cout << data_yrs << endl;
```

```
                */


                 //cout << R << endl;
                 //cout << R(first_data_yr,years-data_lag) << endl;
                 S_proxy = sum(R(first_data_yr,years-data_lag))/data_yrs * SPR_proxy;

               write_final_output();
               if(tvp_int==1) write_time_varying_params();


                 cout<<"Sim: " << i << "; CR: " << cr << ";  lag_int: " <<lag_int << ";
SA_int: " << sa_int  << ";  mr: " << mr << endl;
                 //cout<<"pC_obs(years)"<<endl;
                 //cout<<pC_obs(years)<<endl;
                 //cout<<"pI_obs(years)"<<endl;
                 //cout<<pI_obs(years)<<endl;

                  //cout<<"pC_obs terminal"<<endl;
                 //cout<<pC_obs(t)<<endl;
                 //cout<<pC_obs(years)<<endl;
                 //cout<<"pI_obs terminal"<<endl;
                // cout<<pI_obs(t)<<endl;
                 //cout<<pI_obs(years)<<endl;

          }//end the i loop

          tvp_int++;

       } // end the cr_loop
     } // end the lag data loop
   } // end the assessment interval loop
  } // end the mr loop


FUNCTION get_brps

  Finc = 0.01;
  minF = 0.0;
  F_iters = 200;

  for(f=1; f<=F_iters; f++)
  {
     F_brp = minF + (f-1) * Finc;

     for(a=a_r;a<=amax;a++)
```

```
    {
      Z_brp(a) = M + sf(a) * F_brp;
      if(a==a_r) N_brp(a) = 1.;
      else if(a < amax) N_brp(a) = N_brp(a-1) * exp(-Z_brp(a-1));
      else N_brp(a) =  N_brp(a-1) * exp(-Z_brp(a-1)) / (1.-exp(-Z_brp(a)));
    }
    /*
    cout << mu_sf << endl;
    cout << Z_brp << endl;
    cout << F_brp << " " << N_brp << endl;
    */
    YPR = sum(elem_prod(N_brp,elem_prod(W,elem_prod(elem_div(sf *
F_brp,Z_brp),(1-exp(-Z_brp))))));
    S_brp = sum(elem_prod(N_brp,elem_prod(m,W)));

    if(f==1) Sbrp_0 = S_brp;
    SPR = S_brp / Sbrp_0;
    R_brp = (S_brp - alpha)/(beta*S_brp);
    Y_brp = YPR * R_brp;

    if(f==1)
    {
      F_proxy = F_brp;
      Fmsy = F_brp;
      MSY = Y_brp;
      Smsy = S_brp * R_brp;
    }
    else
    {
       if(Y_brp > MSY)
       {
         Fmsy = F_brp;
         MSY = Y_brp;
         Smsy = S_brp * R_brp;
       }

       if(SPR >= SPR_lim)
       {
         F_proxy = F_brp;
         SPR_proxy = S_brp;
       }
    }

   //cout << f <<" " << F_brp << " " <<YPR << " " << S_brp << " " << SPR << " "
<< R_brp << " " << Y_brp << " " << MSY << " " << Fmsy << " " << F_proxy << " "
<< Smsy << endl;
```

```
  } // end the f loop



// Estimate the F associated with a particular catch.  This function uses the golden
section search.
FUNCTION double estimate_F(dvector N_vec, dvector s_vec, dvector W_vec,
double Total_C, double Est_M)
        double lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;
        double FL;
        double FU;
        double F1;
        double F2;
        double Est_F;
        double C1;
        double C2;
        double CF1;
        double CF2;
        double CU;
        double C_dev;
        double C_dev_targ;

        FL = 0.;
        FU = 2.5;
        C_dev_targ = 0.01;
        C_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        CU =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*FU,M+s_vec*FU),W_vec),N
_vec),(1.-exp(-M-s_vec*FU))));

                //if the highest est catch is still < the observed value, set F = FU
                if(CU <= Total_C)
                {
                        Est_F = FU;
                }

                else if(CU > Total_C){

                        for(int j=1; j<=jmax; j++){
```

```
C1 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F1,M+s_vec*F1),W_vec),N_
vec),(1.-exp(-M-s_vec*F1))));
C2 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F2,M+s_vec*F2),W_vec),N_
vec),(1.-exp(-M-s_vec*F2))));

CF2 = square(Total_C - C2);
CF1 = square(Total_C - C1);
C_dev = CF1 + CF2;

if(CF2 > CF1) {
        Est_F = F1;
        FU = F2;
        FL = FL;
        F2 = F1;
        F1 = FL + (1.-lambda)*(FU-FL);
        }

else if(CF1 > CF2) {
        Est_F = F2;
        FU = FU;
        FL = F1;
        F1 = F2;
        F2 = FU - (1.-lambda)*(FU - FL);
        }

else{
        if(C_dev > C_dev_targ){
                Est_F = F1;
                FU = 1.1*FU;
                FL = 0.9 * FL;
                F2 = FU - (1.-lambda)*(FU-FL);
                F1 = FL + (1.-lambda)*(FU-FL);
                }

        else{
                Est_F = F1;
                break;
                }
        }

if(C_dev <= C_dev_targ){
        Est_F = F1;
        break;
```

```
                                        }

                                //cout <<t << " " << j <<" "<< sum(C_obs(t)) << " " << C1 <<
" " << C2 <<" " << C_dev << " " << Est_F << endl;
                                //cout << sum(N(t))/sum(N_est(t))<<endl;
                                //cout <<j<<" " << sum(C_obs(t)) <<" " <<F1 << " " << C1
<<" " << F2 << " " << C2 << " " << C_dev << endl;

                                }//end for loop
                        }

        return(Est_F);


 /*
// Estimate the F resulting from a particular catch
FUNCTION dvariable estimate_F(dvector N_vec, dvector s_vec, dvector W_vec,
dvariable Total_C, prevariable Est_M)
        dvariable lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;
        dvariable FL;
        dvariable FU;
        dvariable F1;
        dvariable F2;
        dvariable Est_F;
        dvariable C1;
        dvariable C2;
        dvariable CF1;
        dvariable CF2;
        dvariable CU;
        dvariable C_dev;
        dvariable C_dev_targ;

        FL = 0.;
        FU = 2.5;
        C_dev_targ = 0.01;
        C_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        CU =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*FU,M+s_vec*FU),W_vec),N
_vec),(1.-exp(-M-s_vec*FU))));

                //if the highest est catch is still < the observed value, set F = FU
```

```
if(CU <= Total_C)
{
        Est_F = FU;
}

else if(CU > Total_C){

        for(int j=1; j<=jmax; j++){

                C1 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F1,M+s_vec*F1),W_vec),N_
vec),(1.-exp(-M-s_vec*F1))));
                C2 =
sum(elem_prod(elem_prod(elem_prod(elem_div(s_vec*F2,M+s_vec*F2),W_vec),N_
vec),(1.-exp(-M-s_vec*F2))));

                CF2 = square(Total_C - C2);
                CF1 = square(Total_C - C1);
                C_dev = CF1 + CF2;

                if(CF2 > CF1) {
                        Est_F = F1;
                        FU = F2;
                        FL = FL;
                        F2 = F1;
                        F1 = FL + (1.-lambda)*(FU-FL);
                        }

                else if(CF1 > CF2) {
                        Est_F = F2;
                        FU = FU;
                        FL = F1;
                        F1 = F2;
                        F2 = FU - (1.-lambda)*(FU - FL);
                        }

                else{
                        if(C_dev > C_dev_targ){
                                Est_F = F1;
                                FU = 1.1*FU;
                                FL = 0.9 * FL;
                                F2 = FU - (1.-lambda)*(FU-FL);
                                F1 = FL + (1.-lambda)*(FU-FL);
                                }

                        else{
```

```
                                        Est_F = F1;
                                        break;
                                        }
                                }

                        if(C_dev <= C_dev_targ){
                                Est_F = F1;
                                break;
                                }

                        //cout <<t << " " << j <<" "<< sum(C_obs(t)) << " " << C1 <<
" " << C2 <<" " << C_dev << " " << Est_F << endl;
                        //cout << sum(N(t))/sum(N_est(t))<<endl;
                        //cout <<j<<" " << sum(C_obs(t)) <<" " <<F1 << " " << C1
<<" " << F2 << " " << C2 << " " << C_dev << endl;

                        }//end for loop
                }

        return(Est_F);

      */



FUNCTION dvector multinomial(long SS, dvector rand_uniform, dvector p)

        int min_p;
        int dim_p;

        min_p = p.indexmin();
        dim_p = p.indexmax();

        //define local variables to use in this function
        dvector sample(min_p,dim_p);  //create vector for sample from multinomial
        dvector sample_p(min_p,dim_p);
        double p1;
        double p2;
        int ii;
        int jj;
        sample.initialize();
        rand_uniform=sort(rand_uniform);  //sort random numbers


        p1=0;
```

```
        p2=p(min_p);
        jj=1;
        for (ii=min_p;ii<=dim_p;ii++)
        {
           //cout << "p1=" << p1 << " p2=" << p2 << endl;
           sample(ii)=0;
           //assign random uniform numbers to proper bins for multinomial
           while (jj <= SS && rand_uniform(jj) > p1 && rand_uniform(jj) <= p2)
           {
              sample(ii)++;
              jj++;
           }
           if (ii < dim_p)
           {
              p1=p2;
              p2+=p(ii+1);
           }
        }
        sample_p=sample/sum(sample);

        return(sample_p);


FUNCTION double size(dvector obs)
  return(double(obs.indexmax()-obs.indexmin()+1));

FUNCTION double size(param_init_bounded_dev_vector obs)  //overload size
function
  return(double(obs.indexmax()-obs.indexmin()+1));



FUNCTION write_final_output



        /*   ofs_head <<"lh"<< " "<< "sa_err"<< " " << "R_sigma" << " " << "
R_type" <<" "<< " SA_interval"<< " " << "data_lag"<< " " <<"cr" << " "<<  "eh" <<
" " << "i" <<" " << "Year" << " "<<"M"<< " " << "sf50" <<" "<< "q"<< " " <<
"true_S" <<" " << "est_S" << " "<<" Smsy" << " " << "Sproxy" <<" "<<"
est_Sproxy" << " "<< "R"<<" " << "F " << " " << "est_F"<< " "<<"Fmsy" << " " <<
"Flim" << " " << "est_Flim"<< " " << "OFL" <<" " << "ABC" << " " <<  "C_act"<<
" " << "C_obs" << " " << "est_sf50" << " " << "sf_slope" << " " << "gmax" <<
endl;*/

        if(lh==1)
```

```
        {

           ofstream ofs_pm("performance_measures_fast.txt",ios::app);

           for(ttt = 1; ttt<=years; ttt++)
           {
              ofs_pm << lh <<" "<< sa <<" "<< R_sigma << " " << R_type << " " <<
SA_interval << " " << data_lag << " "<< pd_switch << " " << acor_switch << " " <<
abc_switch << " " << proj_switch << " " << rp_switch << " " << cr << " " << eh << "
" << i << " " << ttt << " " << M_t(ttt) << " " << sf50_t(ttt) << " " << qt(ttt) << " " <<
S(ttt) << " " << est_termS(ttt) << " " << Smsy << " " << S_proxy << " " <<
est_Starg(ttt) << " " << R(ttt)<< " " << est_termR(ttt) << " " << F(ttt) << " " <<
est_termF(ttt) << " " << Fmsy << " " << F_proxy << " " <<est_Flim(ttt)<< " "  <<
OFL(ttt) << " " << ABC(ttt) << " " << CW_tot(ttt) << " "  << est_sf50(ttt) << " " <<
est_sf_slope(ttt) << " " << gmax(ttt) << endl;
           }
        }

        if(lh==2)
        {
           ofstream ofs_pm("performance_measures_medium.txt",ios::app);

           for(ttt = 1; ttt<=years; ttt++)
           {
              ofs_pm << lh <<" "<< sa <<" "<< R_sigma << " " << R_type << " " <<
SA_interval << " " << data_lag << " " << pd_switch << " " << acor_switch << " " <<
abc_switch << " " << proj_switch << " " << rp_switch << " " << cr << " " << eh << "
" << i << " " << ttt << " " << M_t(ttt) << " " << sf50_t(ttt) << " " << qt(ttt) << " " <<
S(ttt) << " " << est_termS(ttt) << " " << Smsy << " " << S_proxy << " " <<
est_Starg(ttt) << " " << R(ttt)<< " " << est_termR(ttt) << " " << F(ttt) << " " <<
est_termF(ttt) << " " << Fmsy << " " << F_proxy << " " <<est_Flim(ttt)<< " "  <<
OFL(ttt) << " " << ABC(ttt) << " " << CW_tot(ttt) << " "  << est_sf50(ttt) << " " <<
est_sf_slope(ttt) << " " << gmax(ttt) << endl;
           }
        }


        if(lh==3)
        {
           ofstream ofs_pm("performance_measures_slow.txt",ios::app);

           for(ttt = 1; ttt<=years; ttt++)
           {
              ofs_pm << lh <<" "<< sa <<" "<< R_sigma << " " << R_type << " " <<
SA_interval << " " << data_lag << " " << pd_switch << " " << acor_switch << " " <<
abc_switch << " " << proj_switch << " " << rp_switch << " " << cr << " " << eh << "
```

```
" << i << " " << ttt << " " << M_t(ttt) << " " << sf50_t(ttt) << " " << qt(ttt) << " " <<
S(ttt) << " " << est_termS(ttt) << " " << Smsy << " " << S_proxy << " " <<
est_Starg(ttt) << " " << R(ttt)<< " " << est_termR(ttt) << " " << F(ttt) << " " <<
est_termF(ttt) << " " << Fmsy << " " << F_proxy << " " <<est_Flim(ttt)<< " "  <<
OFL(ttt) << " " << ABC(ttt) << " " << CW_tot(ttt) << " "  << est_sf50(ttt) << " " <<
est_sf_slope(ttt) << " " << gmax(ttt) << endl;
            }
        }


FUNCTION write_time_varying_params

        if(lh==1)
        {
           ofstream tvp("time_varying_params_fast.txt",ios::app);
           {
              for(ttt = 1; ttt<=years; ttt++)
              {
                 tvp << sa << " " << i << " " << ttt  << " " <<  M_t(ttt) << " " <<
sf50_t(ttt) << " " <<sf_slope << " " << qt(ttt) << endl;
              }

           }

        }

        if(lh==2)
        {
           ofstream tvp("time_varying_params_medium.txt",ios::app);
           {
              for(ttt = 1; ttt<=years; ttt++)
              {
                 tvp << sa << " " << i << " " << ttt  << " " <<  M_t(ttt) << " " <<
sf50_t(ttt) << " " <<sf_slope << " " << qt(ttt) << endl;
              }

           }

        }

        if(lh==3)
        {
           ofstream tvp("time_varying_params_slow.txt",ios::app);
           {
              for(ttt = 1; ttt<=years; ttt++)
              {
```

```
                    tvp << sa << " " << i << " " << ttt  << " " <<  M_t(ttt) << " " <<
sf50_t(ttt) << " " <<sf_slope << " " << qt(ttt) << endl;
                }

            }

        }

//FUNCTION int round_to_whole(double val)
//           return(ceil(val - 0.5));

REPORT_SECTION
```

Code III.4: Chapter 2 ADMB Code- Stock assessment model 1: The stock assessment model was called at regular intervals to estimate stock biomass and reference points for management using a statistical catch-at-age model. Stock assessment model 1 used full age composition in the terminal year and was used as control scenarios with both one year and two year data lags.

```
//Andrea Sylvia
//
// THis is ESTIMATION MODEL 1!!!!!!!!!!!
//

DATA_SECTION

  //
**********************************************************************
*****
  //  DATA READ IN FROM  em_cf.dat
  //
**********************************************************************
*****

  init_int lh                                   // life history index
  init_int cr
  init_int iter                                 // which iteration we are on in the
assessment model
  init_int sa_num                               // which stock assessment we are
conducting (within each iteration)
  init_int final_sa                     // how many assessments are to be done?
  init_int years                        // # of years in the model
  init_int first_data_yr                // first year of data collection
  init_int proj_years
  init_int proj_switch
  init_int rp_switch
  init_int OFL_years
 // !! cout<<OFL_years<<endl;
 // !! exit(1);
  init_int amax
          // # of age classes in model
  init_int a_r
          // # age at recruitment to pop
  init_number M
                  // Natural mortality
  init_vector W(a_r,amax)                                                  //
Weight-at-age
```

```
 init_vector m(a_r,amax)                                                    //
maturity-at-age

 init_matrix Catch(first_data_yr,years,a_r,amax)                  // Observed catch
at age
 init_matrix N_Index(first_data_yr,years,a_r,amax)          // Observed adbundance
at age
 init_number C_var
        // Var. in catch
 init_number I_var
        // Var. in index
 init_int ESS
        // survey ESS
 init_vector log_R_true(first_data_yr,years)               // True log(recruitments)
 init_vector log_F_true(first_data_yr,years)               // True log(recruitments)
 init_vector S_true(first_data_yr,years)
 init_vector log_Ninit_true(a_r+1,amax)                                 // True
log(initial population size)
 init_number log_q_true                                                     //
true log(catchability)
 init_number log_sf50_true
        // age @ 50% selectivity in fishery
 init_number log_ss50_true
        // age @ 50% selectivity in the survey
 init_number log_sf_slope                                                   //
age @ 50% selectivity in fishery
 init_number log_ss_slope                                                   //
age @ 50% selectivity in fishery
 //init_vector sf_true(1,amax)                                             // true sele
 //init_vector ss_true(1,amax)
 init_number SPR_lim
 init_int EOF_flag

 int t
                // year index
 int tt
 int a
 int p_int
                // age index
 int SA_interval
 !! SA_interval = OFL_years;

 //!! cout << "2"<< endl;

// Change the phases to 1 for all.
// feed in F values and start pop at true values
```

```
 // take out variable C_obs and I_obs


PARAMETER_SECTION
  init_bounded_number log_mean_R(0.,20.,1)                  //log mean recruitment
  init_bounded_dev_vector log_R_dev(first_data_yr,years,-20.,20.,1)         // log
deviations for mean recruitment

  init_bounded_vector log_Ninit(a_r+1,amax,0.,20.,1)                                //
mean initial pop size (ages
  //init_bounded_dev_vector log_Ninit_dev(2,amax,-20.,20.,1)

  init_bounded_number log_mean_F(-10.,5.,1)                 //log of mean F
  init_bounded_dev_vector log_F_dev(first_data_yr,years,-10,10,1)           //log
deviations from mean F
  init_bounded_number log_qest(-15.,-5.,1)                                          //
log(estimated q)

  init_bounded_number log_sf_50(-2,3,1)                                     //
log(fishery age @ 50% selectivity)
  init_bounded_number log_sf_slope(-5.,5.,1)                        // log(fishery
selectivity slope)
  init_bounded_number log_ss_50(-2.,3,1)                                    //
log(survey age @ 50% selectivity)
  init_bounded_number log_ss_slope(-5.,5.,1)                        // log(fishery
selectivity slope)

  matrix N_est(first_data_yr,years,a_r,amax)                        // Est. abundance
at age
  vector log_Rest(first_data_yr,years)
  matrix C_obs(first_data_yr,years,a_r,amax)                        // Est. catch at age
  matrix C_est(first_data_yr,years,a_r,amax)                        // Est. catch at age
  matrix I_obs(first_data_yr,years,a_r,amax)                        // Est. abundance
index
  matrix I_est(first_data_yr,years,a_r,amax)                        // Est. abundance
index
  matrix pI_est(first_data_yr,years,a_r,amax)                       // Est. proportion
of Index abundance
  matrix pC_est(first_data_yr,years,a_r,amax)                               // Est.
proportion of catch
  matrix pI_obs(first_data_yr,years,a_r,amax)                               //
Proportion of Index abundance
  matrix pC_obs(first_data_yr,years,a_r,amax)                               //
Proportion of catch
```

```
  vector Ninit(a_r+1,amax)
        // initial pop size (ages 2-
  vector F_est(first_data_yr,years)                                    //
Estimated F
  vector ESS_C(first_data_yr,years)                                    // Effective
sample size for catch at age
  vector ESS_I(first_data_yr,years)                                    // Effective
sample size for index at age
  vector log_R(first_data_yr,years)

  vector I_est_tot(first_data_yr,years)                              // Est. abundance
index
  vector C_est_tot(first_data_yr,years)                                // Est. total
catch
  vector C_obs_tot(first_data_yr,years)                                // Est. total
catch
  vector S_est(first_data_yr,years)                                    // Est.
spawning biomass
  vector ss(a_r,amax)
        // selex at age in survey
  vector sf(a_r,amax)
        // selex at age in fishery

  vector OFL(1,OFL_years)
  vector ctarg(1,OFL_years)

  vector N_spr(a_r,amax)
  number SPR
  number Slim
  number FL
  number FU
  number F1
  number F2
  number Est_F
  number SPR_1
  number SPR_2
  number DF1
  number DF2
  number D_dev
  number D_dev_targ
  number SPR_0
  number Flim
  number Slim_temp
  number Flim_temp
  number OFL_est
  number Ftarg
```

```
number mu_Rec

sdreport_number S_term     // terminal spawning biomass

objective_function_value NLL


        LOCAL_CALCS

                if(EOF_flag != 12345)
                {
                        cout << "---------------------------------------"<<endl;
                        cout <<" Data not read in properly to em_cf.tpl !"<< endl;
                        cout << "---------------------------------------"<<endl;
        cout << "# iteration" << endl;
                        cout << iter << endl;
                        cout << "# stock assessment number" << endl;
                        cout << sa_num << endl;
                        cout <<"# end yr "<< endl;
                        cout << years << endl;
                        cout <<"# first year of data  "<< endl;
                        cout << first_data_yr << endl;
                        cout <<" # max age "<< endl;
                        cout << amax << endl;
                        cout <<" # age at recruitment  "<< endl;
                        cout << a_r << endl;
                        cout <<" # Nat. mort  "<< endl;
                        cout << M << endl;
                        cout <<" # Weight "<< endl;
                        cout << W << endl;
                        cout <<" #  maturity"<< endl;
                        cout << m << endl;
                        cout <<" # Catch "<< endl;
                        cout << Catch << endl;
                        cout <<" # N index "<< endl;
                        cout << N_Index << endl;
                        cout <<" #  Var in C"<< endl;
                        cout << C_var << endl;
                        cout <<" # Var in I "<< endl;
                        cout << I_var << endl;
                        cout <<" # ESS "<< endl;
                        cout << ESS << endl;
                        cout <<" #  log (R)"<< endl;
                        cout << log_R_true << endl;
                        cout <<" #  log (F)"<< endl;
                        cout << log_F_true << endl;
```

```
                cout <<" #  log (N init)"<< endl;
                cout << log_Ninit_true << endl;
                cout <<" #  log(q)"<< endl;
                cout << log_q_true << endl;
                cout <<" #  age @ 50% selex in fishery"<< endl;
                cout << log_sf50_true << endl;
                cout <<" #  age at 50% selex in survey"<< endl;
                cout << log_ss50_true << endl;
                cout <<" #  fishery selex"<< endl;
                cout << log_sf_slope << endl;
                cout <<" #  survey selex"<< endl;
                cout << log_ss_slope << endl;
                cout <<" #  EOF flag"<< endl;
                cout << EOF_flag << endl;
                exit(1);
        }

//cout << log_R_true << endl;
//cout << mfexp(log_R_true(years-1,years)) << endl;

ESS_C = ESS;
ESS_I = ESS;

        log_mean_R=sum(log_R_true)/double(years-first_data_yr+1.);
        log_R_dev = log_R_true-log_mean_R;
        log_qest = log_q_true;
        //log_mean_F = -0.35;
log_mean_F = sum(log_F_true)/double(years-first_data_yr+1.);
        log_F_dev = log_F_true-log_mean_F;
log_Ninit = log_Ninit_true;

        //log_sf_50 = log(sf50_true);
        //log_ss_50 = log(ss50_true);

        log_sf_50 = log_sf50_true;
        log_ss_50 = log_ss50_true;



        get_data();



  END_CALCS
```

```
PROCEDURE_SECTION

        get_selectivities();
        get_estimates();
        evaluate_objective_function();
        //cout << Flim << " " << OFL_est << " " << mfexp(log_F_est(years)) << " "
<< sum(Catch(years)) << endl;


FUNCTION get_selectivities

  for(a=a_r; a<=amax; a++)
   {
      ss(a) = 1. / (1.+exp(-(double(a)-mfexp(log_ss_50))/mfexp(log_ss_slope)));
      sf(a) = 1. / (1.+exp(-(double(a)-mfexp(log_sf_50))/mfexp(log_sf_slope)));

        }


FUNCTION get_data
        for(t=first_data_yr;t<=years;t++)
   {
                C_obs(t) = Catch(t);
                I_obs(t) = N_Index(t);
                //log_R(t) = log_R_true(t);
      C_obs_tot(t) = sum(elem_prod(C_obs(t),W));
      pI_obs(t) = I_obs(t) / sum(I_obs(t));
                pC_obs(t) = C_obs(t) / sum(C_obs(t));
        }

FUNCTION get_estimates


        F_est=exp(log_mean_F+log_F_dev);
        log_Rest=(log_mean_R+log_R_dev);

   N_est(first_data_yr,a_r) = exp(log_Rest(first_data_yr));
   //cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   N_est(first_data_yr)((a_r+1),amax) = exp(log_Ninit);
     //  cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   //exit(1);
   //cout << exp(log_Ninit) << endl;

   S_est(first_data_yr) = sum(elem_prod(elem_prod(N_est(first_data_yr),m),W));
```

```
   C_est(first_data_yr) =
elem_prod(N_est(first_data_yr),elem_prod(elem_div(sf*F_est(first_data_yr),M+sf*F
_est(first_data_yr)),(1.-exp(-M-sf*F_est(first_data_yr)))));
   I_est(first_data_yr) = exp(log_qest) * elem_prod(ss, N_est(first_data_yr));
   C_est_tot(first_data_yr) = C_est(first_data_yr) * W;
   pI_est(first_data_yr) =  I_est(first_data_yr) / sum(I_est(first_data_yr));
   pC_est(first_data_yr) = C_est(first_data_yr) / sum(C_est(first_data_yr));


   for(t=first_data_yr+1; t<=years; t++){

      for(a=a_r; a<=amax; a++)
      {
         if(a==a_r) N_est(t,a) = exp(log_Rest(t));
         else
         {
            //if(t==first_data_yr) N_est(t,a) = Ninit(a);
            //else
            {
               if(a < amax) N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1));
               else  N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1)) + N_est(t-
1,a)*exp(-M - sf(a)*F_est(t-1));

            }

         }


      }

            S_est(t) = sum(elem_prod(elem_prod(N_est(t),m),W));
            C_est(t) =
elem_prod(N_est(t),elem_prod(elem_div(sf*F_est(t),M+sf*F_est(t)),(1.-exp(-M-
sf*F_est(t)))));
            I_est(t) = exp(log_qest) * elem_prod(ss, N_est(t));
            C_est_tot(t) = sum(elem_prod(C_est(t),W));

            pI_est(t) =  I_est(t) / sum(I_est(t));
            pC_est(t) = C_est(t) / sum(C_est(t));
         }
      //cout << N_est << endl;
      //exit(1);
         S_term = S_est(years);
```

```
FUNCTION evaluate_objective_function


        NLL = lognorm_nll(C_obs_tot, C_est_tot, C_var) +
lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) +
          multinom_nll(pC_obs, pC_est, ESS_C) + multinom_nll(pI_obs, pI_est,
ESS_I);
      //cout << "neg LL 1 " << lognorm_nll(C_obs_tot, C_est_tot, C_var) << endl <<
"neg LL 2 " << lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) << endl
       //    << "neg LL 3 " <<  multinom_nll(pC_obs, pC_est, ESS_C) << endl << "neg
LL 4 " <<  multinom_nll(pI_obs, pI_est, ESS_I) << endl;

        // lognormal NLL = n*ln(sigma)+0.5∑[(ln(obs)-ln(est))^2 / sigma^2]
        // where sigma^2 = ln(CV^2+1)
        // multionomial NLL = -ESS * ∑obs_p * ln(est_p)

FUNCTION get_spr_brps

        double lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;

        dvariable SPR_targ;
        SPR_targ = 0.35;
  //SPR_targ = SPR_lim;
        FL = 0.;
        FU = 2.;
        D_dev_targ = pow(0.0005,2);
        D_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        N_spr(a_r) = 1.;

        for(a = a_r+1; a <=amax; a++)
  {
                if(a<amax) N_spr(a) = N_spr(a-1) * exp(-M);
    else N_spr(a) = N_spr(a-1) * exp(-M) / (1.-exp(-M));
  }

  //N_spr(amax) += (1.-exp(-M));

        SPR_0 = sum(elem_prod(W,elem_prod(N_spr,m)));

        for(int j=1; j<=jmax; j++){
```

```
                for(a = a_r+1; a <=amax; a++)
        {
           if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F1);
           else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1) * F1) / (1.-exp(-M -
sf(amax)*F1));
        }


        SPR_1 = sum(elem_prod(W,elem_prod(N_spr,m)));

                //cout << "1" << " " << a <<" " <<  N_spr(a) << endl;


                for(a = a_r+1; a <=amax; a++)
        {
                        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2);
           else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2) / (1.-exp(-M -
sf(amax)*F2));
           //cout << "1" << " " << a <<" " <<  " " << sf(a) << " " << N_spr(a) << endl;

        }

        //cout <<F2 << " " <<  N_spr << endl;

        //N_spr(amax) /= (1.-exp(-M -sf(amax)*F2));

        //cout << "2" << " " << a <<" " <<  N_spr(a) << endl;

        SPR_2 = sum(elem_prod(W,elem_prod(N_spr,m)));


                        DF2 = pow(SPR_targ - SPR_2/SPR_0,2);
                        DF1 = pow(SPR_targ - SPR_1/SPR_0,2);

                        D_dev = DF1 + DF2;

                                if(DF2 > DF1) {
                                        Est_F = F1;
                                        SPR = SPR_1;
                                        FU = F2;
                                        FL = FL;
                                        F2 = F1;
                                        F1 = FL + (1.-lambda)*(FU-FL);
                                        }

                                else if(DF1 > DF2) {
```

```
                    Est_F = F2;
                    SPR = SPR_2;
                    FU = FU;
                    FL = F1;
                    F1 = F2;
                    F2 = FU - (1.-lambda)*(FU - FL);
                    }

                else{
                    if(D_dev > D_dev_targ){
                        Est_F = F1;
                        SPR = SPR_1;
                        FU = 1.1*FU;
                        FL = 0.9 * FL;
                        F2 = FU - (1.-lambda)*(FU-FL);
                        F1 = FL + (1.-lambda)*(FU-FL);
                        }

                    else{
                        Est_F = F1;
                        break;
                        }
                    }

                if(D_dev <= D_dev_targ){
                    if(DF1 < DF2){
                        Est_F = F1;
                        SPR = SPR_1;
                        }
                    else{
                        Est_F = F2;
                        SPR = SPR_2;
                        }
                    break;
                    }

                //cout <<j<<" " <<F1 << " " << SPR_1/SPR_0 <<" " << F2 <<
" " << SPR_2/SPR_0 << " " << Est_F << endl;

                }//end for loop

                Flim = Est_F;
                Slim = SPR * sum(mfexp(log_Rest))/size(log_Rest);
```

```
FUNCTION dvariable ofl_calc(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Est_Flim)

        dvar_vector N_ofl(a_r,amax);
        dvariable OFL;

        N_ofl(a_r) = Rec;

        for(a=a_r+1; a<=amax; a++)
        {
                if(a < amax) N_ofl(a) =  N_term(a-1) * exp(-M - sel_f(a-1) * F_cur);
      else N_ofl(a) = N_term(a-1) * exp(-M - sel_f(a-1) * F_cur) + N_term(a) * exp(-
M - sel_f(a) * F_cur);
   }

        OFL =
sum(elem_prod(W,elem_prod(N_ofl,elem_prod(elem_div(sel_f*Est_Flim,M+sel_f*
Est_Flim),(1.-mfexp(-M-sel_f*Est_Flim))))));
        /*
        cout << F_cur << endl;
        cout << Est_Flim << endl;
        cout << M << endl;
        cout << W << endl;
        cout << sel_f << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << OFL << endl;
        */
        return(OFL);

      cout << "OFL"<<OFL<<endl;

//FUNCTION dvariable ofl_proj(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Frec_cur, dvariable Est_Flim, dvariable Est_Ftarg,
data_int pyrs, int proj_switch)
FUNCTION dvar_vector ofl_proj(dvar_vector N_term, named_dvar_vector sel_f,
prevariable Rec, prevariable F_cur, prevariable Est_Flim, data_int pyrs, data_int
proj_switch, data_int rp)
     //changed to dvar_vector ASylvia
   //cout << pyrs << " " << proj_years << endl;

        dvar_matrix N_ofl(1,proj_years,a_r,amax);
        dvar_vector OFL_vec(1,proj_years);
        dvar_vector ctarg_vec(1,proj_years);
   dvar_vector Zt(a_r,amax);
   dvar_vector Z_cur(a_r,amax);
```

```
        dvar_vector Z_lim(a_r,amax);
        dvar_vector Z_targ(a_r,amax);
   dvariable OFL_cur;

        Z_cur =  M + sel_f * F_cur;
        Z_lim =  M + sel_f * Est_Flim;
      //cout<<"1"<<endl;
        //Z_targ =  M + sel_f * Est_Ftarg;

   //cout << Z_cur << endl;
   //cout << Z_lim << endl;
   //cout << Z_targ << endl;
   //cout << 3 << endl;
   //OFL_cur =
sum(elem_prod(W,elem_prod(N_term,elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
   //cout << N_term << endl;
   // replace the terminal estimate of recruitment with the mean value to minimize the
effects of uncertainty

   if(rp>=2)
   {
      N_term(a_r) = Rec;
   }


   //cout << "pyrs: " << pyrs << endl;
   for(t = 1; t<=pyrs; t++)
   {
                N_ofl(t,a_r) = Rec;


        for(a=a_r+1; a<=amax; a++)
        {

                        if(t==1)
                        {
                                Zt = Z_cur;
                        if(a < amax) N_ofl(t,a) =  N_term(a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_term(a-1) * exp(-Zt(a-1)) +  N_term(a) *
exp(-Zt(a));
                        }
                        else
                        {
                        Zt = Z_lim;
                        if(a < amax) N_ofl(t,a) =  N_ofl(t-1,a-1) * exp(-Zt(a-1));
```

```
                        else N_ofl(t,a) = N_ofl(t-1,a-1) * exp(-Zt(a-1)) +  N_ofl(t-1,a)
* exp(-Zt(a));
                        }
        }


                OFL_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
                //ctarg_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Ftarg,Z_targ),
(1.-mfexp(-Z_targ))))));
      //cout << t << N_ofl(t) << endl;
      //cout << Est_Flim << " " << OFL_vec(t) << endl;
   }
      //cout <<  N_ofl << endl;
      //cout << "est_flim: " << Est_Flim << " OFL_vec " << OFL_vec << endl;

   // cout<<"2"<<endl;
        p_int = OFL_years;
        //cout<<proj_years<<endl;
        for(t = 1; t<= OFL_years; t++)
        {
       //cout<<"p_int "<<p_int<<endl;
                if(proj_switch <=1)
                {
                        OFL(t) = OFL_vec(pyrs);
                        //ctarg(t) = ctarg_vec(pyrs);
                }
                else if(proj_switch==2)
                {
                        OFL(t) = OFL_vec(p_int);
                        //ctarg(t) = ctarg(p_int);
                }

                p_int--;
        }
  //    cout<<"3"<<endl;
        //cout << proj_switch << " " << proj_years << " " << OFL << endl;


        /*
        cout << Est_Flim << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
```

```
        cout << sum(elem_prod(elem_prod(N_ofl,m),W))<< " " << S_est(years) <<
endl;
        cout << sum(elem_prod(W,N_ofl)) <<  endl;
        cout << (1.-exp(-M-sel_f*Est_Flim)) << endl;
        cout << elem_div(sel_f*Est_Flim,M+sel_f*Est_Flim) << endl;
        cout << sf << endl;
        cout << sf_true << endl;
        cout << N_term << endl << N_ofl << endl;
        */
        //cout << OFL << endl;
        return(OFL);  //turned back into code ASylvia


FUNCTION write_term_output
  ofstream sae("term_est.txt");
  {
    for(tt = 0; tt <=(SA_interval-1);tt++)
    {
      sae << S_est(years-tt) << " " <<  exp(log_Rest(years-tt)) << " " <<
F_est(years-tt) << endl;


    }
    sae <<Slim<< endl;
    sae <<Flim<< endl;
    sae <<OFL<< endl;
    sae << mfexp(log_sf_50) << " " << mfexp(log_sf_slope) << endl;
    sae << objective_function_value::gmax << endl;
  }


FUNCTION write_final_estimates

          if(lh==1)
          {
              ofstream ofs_fe("final_assessment_estimates_fast.txt",ios::app);
              {
              if(iter==1)
              {
            ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
              }

                    for(t=first_data_yr;t<=years;t++)
               {
```

```
                                ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                    }
                    }

             }


           if(lh==2)
             {
                    ofstream ofs_fe("final_assessment_estimates_medium.txt",ios::app);
                    {
                    if(iter==1)
                    {
                 ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                    }

                            for(t=first_data_yr;t<=years;t++)
                     {

                                ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                    }
                    }

             }

           if(lh==3)
             {
                    ofstream ofs_fe("final_assessment_estimates_slow.txt",ios::app);
                    {
                    if(iter==1)
                    {
                 ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                    }

                            for(t=first_data_yr;t<=years;t++)
                     {
```

```
                                ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                    }
                   }

             }


FUNCTION dvariable lognorm_nll(dvar_vector obs, dvar_vector est, data_number
var)    //Function to calculate a lognormal negative log likelihood
  dvariable L;
  L=norm2(log(obs+0.1)-log(est+0.1));
  L = 0.5*(size(obs))*log(var)+0.5*L/var;



  //cout << endl << L << endl<< endl;
  return(L);

FUNCTION dvariable multinom_nll(dvar_matrix obs_p, dvar_matrix est_p,
dvar_vector eff_size)           // multionomial NLL = -ESS * ∑obs_p * ln(est_p)
          dvariable ML;
          ML = 0.0;
    // make this sum and not rowsum
    // -eff_size change to a value and then change rowsum to sum
          ML = -eff_size * rowsum(elem_prod(obs_p,log(est_p+.001)));

    //cout << ML << endl;
    return(ML);

FUNCTION double size(dvar_vector obs)               // function used to return the
number of elements in a vector
  return(double(obs.indexmax()-obs.indexmin()+1));

FINAL_SECTION


    get_spr_brps();
   // cout<<"1.0"<<endl;
    if(cr <=7) Ftarg = Flim;
    else if(cr==8) Ftarg = 0.75*Flim;

    //cout << rp_switch << endl;
    //cout << mfexp(log_Rest) << endl;
    //cout << mfexp(log_Rest(years-5,years-1)) << endl;
```

```
   if(rp_switch<=2) mu_Rec = sum(mfexp(log_Rest))/size(log_Rest);
   else if(rp_switch==3) mu_Rec = sum(mfexp(log_Rest(years-5,years-1)))/5.;
  // cout<<"1.2"<<endl;


//ofl_calc(N_est(years),sf,sum(mfexp(log_Rest))/size(log_Rest),F_est(years),Flim);
   //
cout<<"N_est"<<N_est(years)<<endl<<"sf"<<sf<<endl<<"mu_Rec"<<mu_Rec<<en
dl<<"F_est"<<F_est(years)<<endl<<"Ftarg"<<Ftarg<<endl<<"proj_years"<<proj_ye
ars<<endl<<"proj_switch"<<proj_switch<<endl<<"rp_switch"<<rp_switch<<endl;
   // cout<<"proj_years " << proj_years<<endl;

OFL=ofl_proj(N_est(years),sf,mu_Rec,F_est(years),Ftarg,proj_years,proj_switch,rp_
switch);
    //added OFL= in front of ofl_proj function
   // cout<<"1.1"<<endl;
   write_term_output();
   //write_r_est();
   //if(sa_num==final_sa) write_final_estimates();

   // get rid of output here, and combine with other files
   /*
   ofstream brp("brp.txt");
   {
      brp << Flim << endl;
      brp << Slim << endl;
   }
   */

   //cout << "estimates" << endl;
   //cout << Flim << " " << Slim << endl;

   //cout << OFL_est << endl;
   //write_ofl();
         //write_sa_output();
         //write_r_est();
         //write_selex();

REPORT_SECTION


         //write_obs_est();
         //write_init_N();
         //write_SR_output();
         //cout << "est model done!" << endl;
```

```
        //write_ssb_output();


RUNTIME_SECTION
  maximum_function_evaluations 4000  // increase the # of evaluations
```

Code III.5: Chapter 2 ADMB Code. Stock assessment model 2, code for data lag reduction method 1: The stock assessment model was called at regular intervals to estimate stock biomass and reference points for management using a statistical catch-at-age model. Stock assessment model 2 used no age composition in the terminal year of the catch.

```
//Andrea Sylvia
//
// THis is ESTIMATION MODEL 2!!!!!!!!!!
//

DATA_SECTION

  //
**********************************************************************
*****
  //  DATA READ IN FROM  em_cf.dat
  //
**********************************************************************
*****

  init_int lh                                   // life history index
  init_int cr
  init_int iter                                 // which iteration we are on in the
assessment model
  init_int sa_num                               // which stock assessment we are
conducting (within each iteration)
  init_int final_sa                     // how many assessments are to be done?
  init_int years                        // # of years in the model
  init_int first_data_yr                // first year of data collection
  init_int proj_years
  init_int proj_switch
  init_int rp_switch
  init_int OFL_years
 // !! cout<<OFL_years<<endl;
 // !! exit(1);
  init_int amax
          // # of age classes in model
  init_int a_r
          // # age at recruitment to pop
  init_number M
                  // Natural mortality
  init_vector W(a_r,amax)                                                  //
Weight-at-age
```

```
 init_vector m(a_r,amax)                                                    //
maturity-at-age

 init_matrix Catch(first_data_yr,years,a_r,amax)                  // Observed catch
at age
 init_matrix N_Index(first_data_yr,years,a_r,amax)          // Observed adbundance
at age
 init_number C_var
        // Var. in catch
 init_number I_var
        // Var. in index
 init_int ESS
        // survey ESS
 init_vector log_R_true(first_data_yr,years)               // True log(recruitments)
 init_vector log_F_true(first_data_yr,years)               // True log(recruitments)
 init_vector S_true(first_data_yr,years)
 init_vector log_Ninit_true(a_r+1,amax)                                 // True
log(initial population size)
 init_number log_q_true                                                     //
true log(catchability)
 init_number log_sf50_true
        // age @ 50% selectivity in fishery
 init_number log_ss50_true
        // age @ 50% selectivity in the survey
 init_number log_sf_slope                                                   //
age @ 50% selectivity in fishery
 init_number log_ss_slope                                                   //
age @ 50% selectivity in fishery
 //init_vector sf_true(1,amax)                                             // true sele
 //init_vector ss_true(1,amax)
 init_number SPR_lim
 init_int EOF_flag

 int t
                // year index
 int tt
 int a
 int p_int
                // age index
 int SA_interval
 !! SA_interval = OFL_years;

 //!! cout << "2"<< endl;

// Change the phases to 1 for all.
// feed in F values and start pop at true values
```

```
 // take out variable C_obs and I_obs


PARAMETER_SECTION
  init_bounded_number log_mean_R(0.,20.,1)                  //log mean recruitment
  init_bounded_dev_vector log_R_dev(first_data_yr,years,-20.,20.,1)         // log
deviations for mean recruitment

  init_bounded_vector log_Ninit(a_r+1,amax,0.,20.,1)                                //
mean initial pop size (ages
  //init_bounded_dev_vector log_Ninit_dev(2,amax,-20.,20.,1)

  init_bounded_number log_mean_F(-10.,5.,1)                 //log of mean F
  init_bounded_dev_vector log_F_dev(first_data_yr,years,-10,10,1)           //log
deviations from mean F
  init_bounded_number log_qest(-15.,-5.,1)                                          //
log(estimated q)

  init_bounded_number log_sf_50(-2,3,1)                                     //
log(fishery age @ 50% selectivity)
  init_bounded_number log_sf_slope(-5.,5.,1)                        // log(fishery
selectivity slope)
  init_bounded_number log_ss_50(-2.,3,1)                                    //
log(survey age @ 50% selectivity)
  init_bounded_number log_ss_slope(-5.,5.,1)                        // log(fishery
selectivity slope)

  matrix N_est(first_data_yr,years,a_r,amax)                        // Est. abundance
at age
  vector log_Rest(first_data_yr,years)
  matrix C_obs(first_data_yr,years,a_r,amax)                        // Est. catch at age
  matrix C_est(first_data_yr,years,a_r,amax)                        // Est. catch at age
  matrix I_obs(first_data_yr,years,a_r,amax)                        // Est. abundance
index
  matrix I_est(first_data_yr,years,a_r,amax)                        // Est. abundance
index
  matrix pI_est(first_data_yr,years,a_r,amax)                       // Est. proportion
of Index abundance
  matrix pC_est(first_data_yr,years,a_r,amax)                               // Est.
proportion of catch
  matrix pI_obs(first_data_yr,years,a_r,amax)                               //
Proportion of Index abundance
  matrix pC_obs(first_data_yr,years,a_r,amax)                               //
Proportion of catch
```

```
  vector Ninit(a_r+1,amax)
        // initial pop size (ages 2-
  vector F_est(first_data_yr,years)                                    //
Estimated F
  vector ESS_C(first_data_yr,years)                                    // Effective
sample size for catch at age
  vector ESS_I(first_data_yr,years)                                    // Effective
sample size for index at age
  vector log_R(first_data_yr,years)

  vector I_est_tot(first_data_yr,years)                              // Est. abundance
index
  vector C_est_tot(first_data_yr,years)                                // Est. total
catch
  vector C_obs_tot(first_data_yr,years)                                // Est. total
catch
  vector S_est(first_data_yr,years)                                    // Est.
spawning biomass
  vector ss(a_r,amax)
        // selex at age in survey
  vector sf(a_r,amax)
        // selex at age in fishery

  vector OFL(1,OFL_years)
  vector ctarg(1,OFL_years)

  vector N_spr(a_r,amax)
  number SPR
  number Slim
  number FL
  number FU
  number F1
  number F2
  number Est_F
  number SPR_1
  number SPR_2
  number DF1
  number DF2
  number D_dev
  number D_dev_targ
  number SPR_0
  number Flim
  number Slim_temp
  number Flim_temp
  number OFL_est
  number Ftarg
```

```
number mu_Rec

sdreport_number S_term     // terminal spawning biomass

objective_function_value NLL


        LOCAL_CALCS

                if(EOF_flag != 12345)
                {
                        cout << "----------------------------------------"<<endl;
                        cout <<" Data not read in properly to em_cf.tpl !"<< endl;
                        cout << "----------------------------------------"<<endl;
        cout << "# iteration" << endl;
                        cout << iter << endl;
                        cout << "# stock assessment number" << endl;
                        cout << sa_num << endl;
                        cout <<"# end yr "<< endl;
                        cout << years << endl;
                        cout <<"# first year of data  "<< endl;
                        cout << first_data_yr << endl;
                        cout <<" # max age "<< endl;
                        cout << amax << endl;
                        cout <<" # age at recruitment  "<< endl;
                        cout << a_r << endl;
                        cout <<" # Nat. mort  "<< endl;
                        cout << M << endl;
                        cout <<" # Weight "<< endl;
                        cout << W << endl;
                        cout <<" #  maturity"<< endl;
                        cout << m << endl;
                        cout <<" # Catch "<< endl;
                        cout << Catch << endl;
                        cout <<" # N index "<< endl;
                        cout << N_Index << endl;
                        cout <<" #  Var in C"<< endl;
                        cout << C_var << endl;
                        cout <<" # Var in I "<< endl;
                        cout << I_var << endl;
                        cout <<" # ESS "<< endl;
                        cout << ESS << endl;
                        cout <<" #  log (R)"<< endl;
                        cout << log_R_true << endl;
                        cout <<" #  log (F)"<< endl;
                        cout << log_F_true << endl;
```

```
                cout <<" #  log (N init)"<< endl;
                cout << log_Ninit_true << endl;
                cout <<" #  log(q)"<< endl;
                cout << log_q_true << endl;
                cout <<" #  age @ 50% selex in fishery"<< endl;
                cout << log_sf50_true << endl;
                cout <<" #  age at 50% selex in survey"<< endl;
                cout << log_ss50_true << endl;
                cout <<" #  fishery selex"<< endl;
                cout << log_sf_slope << endl;
                cout <<" #  survey selex"<< endl;
                cout << log_ss_slope << endl;
                cout <<" #  EOF flag"<< endl;
                cout << EOF_flag << endl;
                exit(1);
        }

//cout << log_R_true << endl;
//cout << mfexp(log_R_true(years-1,years)) << endl;

ESS_C = ESS;
ESS_I = ESS;

        log_mean_R=sum(log_R_true)/double(years-first_data_yr+1.);
        log_R_dev = log_R_true-log_mean_R;
        log_qest = log_q_true;
        //log_mean_F = -0.35;
log_mean_F = sum(log_F_true)/double(years-first_data_yr+1.);
        log_F_dev = log_F_true-log_mean_F;
log_Ninit = log_Ninit_true;

        //log_sf_50 = log(sf50_true);
        //log_ss_50 = log(ss50_true);

        log_sf_50 = log_sf50_true;
        log_ss_50 = log_ss50_true;



        get_data();



  END_CALCS
```

```
PROCEDURE_SECTION

        get_selectivities();
        get_estimates();
        evaluate_objective_function();
        //cout << Flim << " " << OFL_est << " " << mfexp(log_F_est(years)) << " "
<< sum(Catch(years)) << endl;


FUNCTION get_selectivities

  for(a=a_r; a<=amax; a++)
   {
      ss(a) = 1. / (1.+exp(-(double(a)-mfexp(log_ss_50))/mfexp(log_ss_slope)));
      sf(a) = 1. / (1.+exp(-(double(a)-mfexp(log_sf_50))/mfexp(log_sf_slope)));

        }


FUNCTION get_data
        for(t=first_data_yr;t<=years;t++)
  {
                C_obs(t) = Catch(t);
                I_obs(t) = N_Index(t);
                //log_R(t) = log_R_true(t);
     C_obs_tot(t) = sum(elem_prod(C_obs(t),W));
     pI_obs(t) = I_obs(t) / sum(I_obs(t));

        }
      for(t=first_data_yr; t<=years-1; t++)     //edited:ALS
        {

          pC_obs(t) = C_obs(t) / sum(C_obs(t));

        }

FUNCTION get_estimates


        F_est=exp(log_mean_F+log_F_dev);
        log_Rest=(log_mean_R+log_R_dev);

   N_est(first_data_yr,a_r) = exp(log_Rest(first_data_yr));
   //cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   N_est(first_data_yr)((a_r+1),amax) = exp(log_Ninit);
    //  cout << N_est(first_data_yr)((a_r+1),amax) << endl;
```

```
   //exit(1);
   //cout << exp(log_Ninit) << endl;

   S_est(first_data_yr) = sum(elem_prod(elem_prod(N_est(first_data_yr),m),W));
   C_est(first_data_yr) =
elem_prod(N_est(first_data_yr),elem_prod(elem_div(sf*F_est(first_data_yr),M+sf*F
_est(first_data_yr)),(1.-exp(-M-sf*F_est(first_data_yr)))));
   I_est(first_data_yr) = exp(log_qest) * elem_prod(ss, N_est(first_data_yr));
   C_est_tot(first_data_yr) = C_est(first_data_yr) * W;
   pI_est(first_data_yr) =  I_est(first_data_yr) / sum(I_est(first_data_yr));
   pC_est(first_data_yr) = C_est(first_data_yr) / sum(C_est(first_data_yr));


   for(t=first_data_yr+1; t<=years; t++){

      for(a=a_r; a<=amax; a++)
      {
         if(a==a_r) N_est(t,a) = exp(log_Rest(t));
         else
         {
            //if(t==first_data_yr) N_est(t,a) = Ninit(a);
            //else
            {
               if(a < amax) N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1));
               else  N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1)) + N_est(t-
1,a)*exp(-M - sf(a)*F_est(t-1));

            }

         }


      }

            S_est(t) = sum(elem_prod(elem_prod(N_est(t),m),W));
            C_est(t) =
elem_prod(N_est(t),elem_prod(elem_div(sf*F_est(t),M+sf*F_est(t)),(1.-exp(-M-
sf*F_est(t)))));
            I_est(t) = exp(log_qest) * elem_prod(ss, N_est(t));
            C_est_tot(t) = sum(elem_prod(C_est(t),W));

            pI_est(t) =  I_est(t) / sum(I_est(t));

          }
         for(t=first_data_yr; t<=years-1; t++)     //edited:ALS
          {
```

```
            pC_est(t) = C_est(t) / sum(C_est(t));

          }

       cout<<"pC_obs terminal"<<endl;
       cout<<pC_obs(t-1)<<endl;
          cout<<pC_obs(years)<<endl;
       cout<<"pC_est terminal"<<endl;
       cout<<pC_est(t-1)<<endl;
          cout<<pC_est(years)<<endl;
      // cout<<"pI_obs terminal"<<endl;
      // cout<<pI_obs(t-1)<<endl;
      //   cout<<pI_obs(years)<<endl;
       //cout << N_est << endl;
      //exit(1);
          S_term = S_est(years);



FUNCTION evaluate_objective_function


          NLL = lognorm_nll(C_obs_tot, C_est_tot, C_var) +
lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) +
             multinom_nll(pC_obs, pC_est, ESS_C) + multinom_nll(pI_obs, pI_est,
ESS_I);
       //cout << "neg LL 1 " << lognorm_nll(C_obs_tot, C_est_tot, C_var) << endl <<
"neg LL 2 " << lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) << endl
        //    << "neg LL 3 " <<  multinom_nll(pC_obs, pC_est, ESS_C) << endl << "neg
LL 4 " <<  multinom_nll(pI_obs, pI_est, ESS_I) << endl;

          // lognormal NLL = n*ln(sigma)+0.5∑[(ln(obs)-ln(est))^2 / sigma^2]
          // where sigma^2 = ln(CV^2+1)
          // multionomial NLL = -ESS * ∑obs_p * ln(est_p)

FUNCTION get_spr_brps

          double lambda = (sqrt(5.) - 1.)/2.;
          int jmax=200;

          dvariable SPR_targ;
          SPR_targ = 0.35;
    //SPR_targ = SPR_lim;
          FL = 0.;
          FU = 2.;
```

```
        D_dev_targ = pow(0.0005,2);
        D_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        N_spr(a_r) = 1.;

        for(a = a_r+1; a <=amax; a++)
   {
                if(a<amax) N_spr(a) = N_spr(a-1) * exp(-M);
      else N_spr(a) = N_spr(a-1) * exp(-M) / (1.-exp(-M));
   }

   //N_spr(amax) += (1.-exp(-M));

        SPR_0 = sum(elem_prod(W,elem_prod(N_spr,m)));

        for(int j=1; j<=jmax; j++){

                for(a = a_r+1; a <=amax; a++)
      {
        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F1);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1) * F1) / (1.-exp(-M -
sf(amax)*F1));
      }


      SPR_1 = sum(elem_prod(W,elem_prod(N_spr,m)));

                //cout << "1" << " " << a <<" " <<  N_spr(a) << endl;


                for(a = a_r+1; a <=amax; a++)
      {
                        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2) / (1.-exp(-M -
sf(amax)*F2));
        //cout << "1" << " " << a <<" " <<  " " << sf(a) << " " << N_spr(a) << endl;

      }

      //cout <<F2 << " " <<  N_spr << endl;

      //N_spr(amax) /= (1.-exp(-M -sf(amax)*F2));
```

```
//cout << "2" << " " << a <<" " <<  N_spr(a) << endl;

SPR_2 = sum(elem_prod(W,elem_prod(N_spr,m)));


                DF2 = pow(SPR_targ - SPR_2/SPR_0,2);
                DF1 = pow(SPR_targ - SPR_1/SPR_0,2);

                D_dev = DF1 + DF2;

                        if(DF2 > DF1) {
                                Est_F = F1;
                                SPR = SPR_1;
                                FU = F2;
                                FL = FL;
                                F2 = F1;
                                F1 = FL + (1.-lambda)*(FU-FL);
                                }

                        else if(DF1 > DF2) {
                                Est_F = F2;
                                SPR = SPR_2;
                                FU = FU;
                                FL = F1;
                                F1 = F2;
                                F2 = FU - (1.-lambda)*(FU - FL);
                                }

                        else{
                                if(D_dev > D_dev_targ){
                                        Est_F = F1;
                                        SPR = SPR_1;
                                        FU = 1.1*FU;
                                        FL = 0.9 * FL;
                                        F2 = FU - (1.-lambda)*(FU-FL);
                                        F1 = FL + (1.-lambda)*(FU-FL);
                                        }

                                else{
                                        Est_F = F1;
                                        break;
                                        }
                                }

                        if(D_dev <= D_dev_targ){
                                if(DF1 < DF2){
```

```
                                        Est_F = F1;
                                        SPR = SPR_1;
                                        }
                                else{
                                        Est_F = F2;
                                        SPR = SPR_2;
                                        }
                                break;
                                }

                        //cout <<j<<" " <<F1 << " " << SPR_1/SPR_0 <<" " << F2 <<
" " << SPR_2/SPR_0 << " " << Est_F << endl;

                        }//end for loop

                        Flim = Est_F;
                        Slim = SPR * sum(mfexp(log_Rest))/size(log_Rest);



FUNCTION dvariable ofl_calc(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Est_Flim)

        dvar_vector N_ofl(a_r,amax);
        dvariable OFL;

        N_ofl(a_r) = Rec;

        for(a=a_r+1; a<=amax; a++)
        {
                if(a < amax) N_ofl(a) =  N_term(a-1) * exp(-M - sel_f(a-1) * F_cur);
      else N_ofl(a) = N_term(a-1) * exp(-M - sel_f(a-1) * F_cur) + N_term(a) * exp(-
M - sel_f(a) * F_cur);
   }

        OFL =
sum(elem_prod(W,elem_prod(N_ofl,elem_prod(elem_div(sel_f*Est_Flim,M+sel_f*
Est_Flim),(1.-mfexp(-M-sel_f*Est_Flim))))));
        /*
        cout << F_cur << endl;
        cout << Est_Flim << endl;
        cout << M << endl;
        cout << W << endl;
        cout << sel_f << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
```

```
        cout << OFL << endl;
        */
        return(OFL);

      cout << "OFL"<<OFL<<endl;

//FUNCTION dvariable ofl_proj(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Frec_cur, dvariable Est_Flim, dvariable Est_Ftarg,
data_int pyrs, int proj_switch)
FUNCTION dvar_vector ofl_proj(dvar_vector N_term, named_dvar_vector sel_f,
prevariable Rec, prevariable F_cur, prevariable Est_Flim, data_int pyrs, data_int
proj_switch, data_int rp)
     //changed to dvar_vector ASylvia
   //cout << pyrs << " " << proj_years << endl;

        dvar_matrix N_ofl(1,proj_years,a_r,amax);
        dvar_vector OFL_vec(1,proj_years);
        dvar_vector ctarg_vec(1,proj_years);
   dvar_vector Zt(a_r,amax);
   dvar_vector Z_cur(a_r,amax);
        dvar_vector Z_lim(a_r,amax);
        dvar_vector Z_targ(a_r,amax);
   dvariable OFL_cur;

        Z_cur =  M + sel_f * F_cur;
        Z_lim =  M + sel_f * Est_Flim;
      //cout<<"1"<<endl;
        //Z_targ =  M + sel_f * Est_Ftarg;

   //cout << Z_cur << endl;
   //cout << Z_lim << endl;
   //cout << Z_targ << endl;
   //cout << 3 << endl;
   //OFL_cur =
sum(elem_prod(W,elem_prod(N_term,elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
   //cout << N_term << endl;
   // replace the terminal estimate of recruitment with the mean value to minimize the
effects of uncertainty

   if(rp>=2)
   {
      N_term(a_r) = Rec;
   }
```

```
   //cout << "pyrs: " << pyrs << endl;
   for(t = 1; t<=pyrs; t++)
   {
                N_ofl(t,a_r) = Rec;


        for(a=a_r+1; a<=amax; a++)
        {

                        if(t==1)
                        {
                                Zt = Z_cur;
                        if(a < amax) N_ofl(t,a) =  N_term(a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_term(a-1) * exp(-Zt(a-1)) +  N_term(a) *
exp(-Zt(a));
                        }
                        else
                        {
                        Zt = Z_lim;
                        if(a < amax) N_ofl(t,a) =  N_ofl(t-1,a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_ofl(t-1,a-1) * exp(-Zt(a-1)) +  N_ofl(t-1,a)
* exp(-Zt(a));
                        }
        }


                OFL_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
                //ctarg_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Ftarg,Z_targ),
(1.-mfexp(-Z_targ))))));
      //cout << t << N_ofl(t) << endl;
      //cout << Est_Flim << " " << OFL_vec(t) << endl;
   }
      //cout <<  N_ofl << endl;
      //cout << "est_flim: " << Est_Flim << " OFL_vec " << OFL_vec << endl;

    // cout<<"2"<<endl;
        p_int = OFL_years;
        //cout<<proj_years<<endl;
        for(t = 1; t<= OFL_years; t++)
        {
       //cout<<"p_int "<<p_int<<endl;
                if(proj_switch <=1)
                {
```

```
                OFL(t) = OFL_vec(pyrs);
                //ctarg(t) = ctarg_vec(pyrs);
            }
            else if(proj_switch==2)
            {
                OFL(t) = OFL_vec(p_int);
                //ctarg(t) = ctarg(p_int);
            }

            p_int--;
        }
  //    cout<<"3"<<endl;
        //cout << proj_switch << " " << proj_years << " " << OFL << endl;


        /*
        cout << Est_Flim << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << sum(elem_prod(elem_prod(N_ofl,m),W))<< " " << S_est(years) <<
endl;
        cout << sum(elem_prod(W,N_ofl)) <<  endl;
        cout << (1.-exp(-M-sel_f*Est_Flim)) << endl;
        cout << elem_div(sel_f*Est_Flim,M+sel_f*Est_Flim) << endl;
        cout << sf << endl;
        cout << sf_true << endl;
        cout << N_term << endl << N_ofl << endl;
        */
        //cout << OFL << endl;
        return(OFL);  //turned back into code ASylvia


FUNCTION write_term_output
   ofstream sae("term_est.txt");
   {
      for(tt = 0; tt <=(SA_interval-1);tt++)
      {
         sae << S_est(years-tt) << " " <<  exp(log_Rest(years-tt)) << " " <<
F_est(years-tt) << endl;


      }
      sae <<Slim<< endl;
      sae <<Flim<< endl;
      sae <<OFL<< endl;
      sae << mfexp(log_sf_50) << " " << mfexp(log_sf_slope) << endl;
```

```
    sae << objective_function_value::gmax << endl;
  }


FUNCTION write_final_estimates

          if(lh==1)
          {
               ofstream ofs_fe("final_assessment_estimates_fast.txt",ios::app);
               {
               if(iter==1)
               {
             ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
               }

                         for(t=first_data_yr;t<=years;t++)
                {

                              ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
               }

          }


        if(lh==2)
          {
               ofstream ofs_fe("final_assessment_estimates_medium.txt",ios::app);
               {
               if(iter==1)
               {
             ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
               }

                         for(t=first_data_yr;t<=years;t++)
                {

                              ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
```

```
                }
                }

            }

        if(lh==3)
            {
                ofstream ofs_fe("final_assessment_estimates_slow.txt",ios::app);
                {
                if(iter==1)
                {
              ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                }

                        for(t=first_data_yr;t<=years;t++)
                {

                            ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
                }

            }


FUNCTION dvariable lognorm_nll(dvar_vector obs, dvar_vector est, data_number
var)    //Function to calculate a lognormal negative log likelihood
  dvariable L;
  L=norm2(log(obs+0.1)-log(est+0.1));
  L = 0.5*(size(obs))*log(var)+0.5*L/var;



  //cout << endl << L << endl<< endl;
  return(L);

FUNCTION dvariable multinom_nll(dvar_matrix obs_p, dvar_matrix est_p,
dvar_vector eff_size)            // multionomial NLL = -ESS * ∑obs_p * ln(est_p)
        dvariable ML;
        ML = 0.0;
    // make this sum and not rowsum
    // -eff_size change to a value and then change rowsum to sum
        ML = -eff_size * rowsum(elem_prod(obs_p,log(est_p+.001)));
```

```
   //cout << ML << endl;
   return(ML);

FUNCTION double size(dvar_vector obs)              // function used to return the
number of elements in a vector
  return(double(obs.indexmax()-obs.indexmin()+1));

FINAL_SECTION


   get_spr_brps();
  // cout<<"1.0"<<endl;
   if(cr <=7) Ftarg = Flim;
   else if(cr==8) Ftarg = 0.75*Flim;

   //cout << rp_switch << endl;
   //cout << mfexp(log_Rest) << endl;
   //cout << mfexp(log_Rest(years-5,years-1)) << endl;

   if(rp_switch<=2) mu_Rec = sum(mfexp(log_Rest))/size(log_Rest);
   else if(rp_switch==3) mu_Rec = sum(mfexp(log_Rest(years-5,years-1)))/5.;
  // cout<<"1.2"<<endl;


//ofl_calc(N_est(years),sf,sum(mfexp(log_Rest))/size(log_Rest),F_est(years),Flim);
   //
cout<<"N_est"<<N_est(years)<<endl<<"sf"<<sf<<endl<<"mu_Rec"<<mu_Rec<<en
dl<<"F_est"<<F_est(years)<<endl<<"Ftarg"<<Ftarg<<endl<<"proj_years"<<proj_ye
ars<<endl<<"proj_switch"<<proj_switch<<endl<<"rp_switch"<<rp_switch<<endl;
   // cout<<"proj_years " << proj_years<<endl;

OFL=ofl_proj(N_est(years),sf,mu_Rec,F_est(years),Ftarg,proj_years,proj_switch,rp_
switch);
    //added OFL= in front of ofl_proj function
   // cout<<"1.1"<<endl;
   write_term_output();
   //write_r_est();
   //if(sa_num==final_sa) write_final_estimates();

   // get rid of output here, and combine with other files
   /*
   ofstream brp("brp.txt");
   {
      brp << Flim << endl;
      brp << Slim << endl;
```

```
  }
  */

  //cout << "estimates" << endl;
  //cout << Flim << " " << Slim << endl;

  //cout << OFL_est << endl;
  //write_ofl();
      //write_sa_output();
      //write_r_est();
      //write_selex();

REPORT_SECTION


      //write_obs_est();
      //write_init_N();
      //write_SR_output();
      //cout << "est model done!" << endl;
      //write_ssb_output();


RUNTIME_SECTION
  maximum_function_evaluations 4000  // increase the # of evaluations
```

Code III.5: Chapter 2 ADMB Code. Stock assessment model 3, code for alg reduction method 2: The stock assessment model was called at regular intervals to estimate stock biomass and reference points for management using a statistical catch-at-age model. Stock assessment model 3 used a decresed effective sample size for the composition in the terminal year of the catch.

```
//Andrea Sylvia
//
// THis is ESTIMATION MODEL 3!!!!!!!!!!!
//

DATA_SECTION

  //
**********************************************************************
*****
  //  DATA READ IN FROM  em_cf.dat
  //
**********************************************************************
*****

  init_int lh                                   // life history index
  init_int cr
  init_int iter                                 // which iteration we are on in the
assessment model
  init_int sa_num                               // which stock assessment we are
conducting (within each iteration)
  init_int final_sa                     // how many assessments are to be done?
  init_int years                        // # of years in the model
  init_int first_data_yr                // first year of data collection
  init_int proj_years
  init_int proj_switch
  init_int rp_switch
  init_int OFL_years
 // !! cout<<OFL_years<<endl;
 // !! exit(1);
  init_int amax
          // # of age classes in model
  init_int a_r
          // # age at recruitment to pop
  init_number M
                  // Natural mortality
  init_vector W(a_r,amax)                                                 //
Weight-at-age
  init_vector m(a_r,amax)                                                 //
maturity-at-age
```

```
 init_matrix Catch(first_data_yr,years,a_r,amax)                    // Observed catch
at age
 init_matrix N_Index(first_data_yr,years,a_r,amax)          // Observed adbundance
at age
 init_number C_var
        // Var. in catch
 init_number I_var
        // Var. in index
 init_int ESS
        // survey ESS
 init_vector log_R_true(first_data_yr,years)                // True log(recruitments)
 init_vector log_F_true(first_data_yr,years)                // True log(recruitments)
 init_vector S_true(first_data_yr,years)
 init_vector log_Ninit_true(a_r+1,amax)                                  // True
log(initial population size)
 init_number log_q_true                                                  //
true log(catchability)
 init_number log_sf50_true
        // age @ 50% selectivity in fishery
 init_number log_ss50_true
        // age @ 50% selectivity in the survey
 init_number log_sf_slope                                                //
age @ 50% selectivity in fishery
 init_number log_ss_slope                                                //
age @ 50% selectivity in fishery
 //init_vector sf_true(1,amax)                                           // true sele
 //init_vector ss_true(1,amax)
 init_number SPR_lim
 init_int EOF_flag

 int t
                // year index
 int tt
 int a
 int p_int
                // age index
 int SA_interval
 !! SA_interval = OFL_years;

 //!! cout << "2"<< endl;

// Change the phases to 1 for all.
// feed in F values and start pop at true values
// take out variable C_obs and I_obs
```

```
PARAMETER_SECTION
  init_bounded_number log_mean_R(0.,20.,1)                    //log mean recruitment
  init_bounded_dev_vector log_R_dev(first_data_yr,years,-20.,20.,1)         // log
deviations for mean recruitment

  init_bounded_vector log_Ninit(a_r+1,amax,0.,20.,1)                                  //
mean initial pop size (ages
  //init_bounded_dev_vector log_Ninit_dev(2,amax,-20.,20.,1)

  init_bounded_number log_mean_F(-10.,5.,1)                   //log of mean F
  init_bounded_dev_vector log_F_dev(first_data_yr,years,-10,10,1)           //log
deviations from mean F
  init_bounded_number log_qest(-15.,-5.,1)                                            //
log(estimated q)

  init_bounded_number log_sf_50(-2,3,1)                                       //
log(fishery age @ 50% selectivity)
  init_bounded_number log_sf_slope(-5.,5.,1)                        // log(fishery
selectivity slope)
  init_bounded_number log_ss_50(-2.,3,1)                                      //
log(survey age @ 50% selectivity)
  init_bounded_number log_ss_slope(-5.,5.,1)                        // log(fishery
selectivity slope)

  matrix N_est(first_data_yr,years,a_r,amax)                        // Est. abundance
at age
  vector log_Rest(first_data_yr,years)
  matrix C_obs(first_data_yr,years,a_r,amax)                        // Est. catch at age
  matrix C_est(first_data_yr,years,a_r,amax)                        // Est. catch at age
  matrix I_obs(first_data_yr,years,a_r,amax)                        // Est. abundance
index
  matrix I_est(first_data_yr,years,a_r,amax)                        // Est. abundance
index
  matrix pI_est(first_data_yr,years,a_r,amax)                       // Est. proportion
of Index abundance
  matrix pC_est(first_data_yr,years,a_r,amax)                                 // Est.
proportion of catch
  matrix pI_obs(first_data_yr,years,a_r,amax)                                 //
Proportion of Index abundance
  matrix pC_obs(first_data_yr,years,a_r,amax)                                 //
Proportion of catch

  vector Ninit(a_r+1,amax)
        // initial pop size (ages 2-
```

```
  vector F_est(first_data_yr,years)                                    //
Estimated F
  vector ESS_C(first_data_yr,years)                                    // Effective
sample size for catch at age
  vector ESS_I(first_data_yr,years)                                    // Effective
sample size for index at age
  vector log_R(first_data_yr,years)

  vector I_est_tot(first_data_yr,years)                              // Est. abundance
index
  vector C_est_tot(first_data_yr,years)                                // Est. total
catch
  vector C_obs_tot(first_data_yr,years)                                // Est. total
catch
  vector S_est(first_data_yr,years)                                    // Est.
spawning biomass
  vector ss(a_r,amax)
        // selex at age in survey
  vector sf(a_r,amax)
        // selex at age in fishery

  vector OFL(1,OFL_years)
  vector ctarg(1,OFL_years)

  vector N_spr(a_r,amax)
  number SPR
  number Slim
  number FL
  number FU
  number F1
  number F2
  number Est_F
  number SPR_1
  number SPR_2
  number DF1
  number DF2
  number D_dev
  number D_dev_targ
  number SPR_0
  number Flim
  number Slim_temp
  number Flim_temp
  number OFL_est
  number Ftarg
  number mu_Rec
```

```
sdreport_number S_term     // terminal spawning biomass

objective_function_value NLL


        LOCAL_CALCS

                if(EOF_flag != 12345)
                {
                        cout << "---------------------------------------"<<endl;
                        cout <<" Data not read in properly to em_cf.tpl !"<< endl;
                        cout << "---------------------------------------"<<endl;
        cout << "# iteration" << endl;
                        cout << iter << endl;
                        cout << "# stock assessment number" << endl;
                        cout << sa_num << endl;
                        cout <<"# end yr "<< endl;
                        cout << years << endl;
                        cout <<"# first year of data  "<< endl;
                        cout << first_data_yr << endl;
                        cout <<" # max age "<< endl;
                        cout << amax << endl;
                        cout <<" # age at recruitment  "<< endl;
                        cout << a_r << endl;
                        cout <<" # Nat. mort  "<< endl;
                        cout << M << endl;
                        cout <<" # Weight "<< endl;
                        cout << W << endl;
                        cout <<" #  maturity"<< endl;
                        cout << m << endl;
                        cout <<" # Catch "<< endl;
                        cout << Catch << endl;
                        cout <<" # N index "<< endl;
                        cout << N_Index << endl;
                        cout <<" #  Var in C"<< endl;
                        cout << C_var << endl;
                        cout <<" # Var in I "<< endl;
                        cout << I_var << endl;
                        cout <<" # ESS "<< endl;
                        cout << ESS << endl;
                        cout <<" #  log (R)"<< endl;
                        cout << log_R_true << endl;
                        cout <<" #  log (F)"<< endl;
                        cout << log_F_true << endl;
                        cout <<" #  log (N init)"<< endl;
                        cout << log_Ninit_true << endl;
```

```
                cout <<" #  log(q)"<< endl;
                cout << log_q_true << endl;
                cout <<" #  age @ 50% selex in fishery"<< endl;
                cout << log_sf50_true << endl;
                cout <<" #  age at 50% selex in survey"<< endl;
                cout << log_ss50_true << endl;
                cout <<" #  fishery selex"<< endl;
                cout << log_sf_slope << endl;
                cout <<" #  survey selex"<< endl;
                cout << log_ss_slope << endl;
                cout <<" #  EOF flag"<< endl;
                cout << EOF_flag << endl;
                exit(1);
        }

//cout << log_R_true << endl;
//cout << mfexp(log_R_true(years-1,years)) << endl;

ESS_C = ESS;
ESS_I = ESS;

        log_mean_R=sum(log_R_true)/double(years-first_data_yr+1.);
        log_R_dev = log_R_true-log_mean_R;
        log_qest = log_q_true;
        //log_mean_F = -0.35;
log_mean_F = sum(log_F_true)/double(years-first_data_yr+1.);
        log_F_dev = log_F_true-log_mean_F;
log_Ninit = log_Ninit_true;

        //log_sf_50 = log(sf50_true);
        //log_ss_50 = log(ss50_true);

        log_sf_50 = log_sf50_true;
        log_ss_50 = log_ss50_true;



        get_data();



   END_CALCS

PROCEDURE_SECTION
```

```
        get_selectivities();
        get_estimates();
        evaluate_objective_function();
        //cout << Flim << " " << OFL_est << " " << mfexp(log_F_est(years)) << " "
<< sum(Catch(years)) << endl;


FUNCTION get_selectivities

  for(a=a_r; a<=amax; a++)
   {
      ss(a) = 1. / (1.+exp(-(double(a)-mfexp(log_ss_50))/mfexp(log_ss_slope)));
      sf(a) = 1. / (1.+exp(-(double(a)-mfexp(log_sf_50))/mfexp(log_sf_slope)));

        }


FUNCTION get_data
        for(t=first_data_yr;t<=years;t++)
   {
                C_obs(t) = Catch(t);
                I_obs(t) = N_Index(t);
                //log_R(t) = log_R_true(t);
      C_obs_tot(t) = sum(elem_prod(C_obs(t),W));
      pI_obs(t) = I_obs(t) / sum(I_obs(t));
                pC_obs(t) = C_obs(t) / sum(C_obs(t));
        }

FUNCTION get_estimates


        F_est=exp(log_mean_F+log_F_dev);
        log_Rest=(log_mean_R+log_R_dev);

   N_est(first_data_yr,a_r) = exp(log_Rest(first_data_yr));
   //cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   N_est(first_data_yr)((a_r+1),amax) = exp(log_Ninit);
     //  cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   //exit(1);
   //cout << exp(log_Ninit) << endl;

   S_est(first_data_yr) = sum(elem_prod(elem_prod(N_est(first_data_yr),m),W));
   C_est(first_data_yr) =
elem_prod(N_est(first_data_yr),elem_prod(elem_div(sf*F_est(first_data_yr),M+sf*F
_est(first_data_yr)),(1.-exp(-M-sf*F_est(first_data_yr)))));
   I_est(first_data_yr) = exp(log_qest) * elem_prod(ss, N_est(first_data_yr));
```

```
   C_est_tot(first_data_yr) = C_est(first_data_yr) * W;
   pI_est(first_data_yr) =  I_est(first_data_yr) / sum(I_est(first_data_yr));
   pC_est(first_data_yr) = C_est(first_data_yr) / sum(C_est(first_data_yr));


   for(t=first_data_yr+1; t<=years; t++){

      for(a=a_r; a<=amax; a++)
      {
         if(a==a_r) N_est(t,a) = exp(log_Rest(t));
         else
         {
            //if(t==first_data_yr) N_est(t,a) = Ninit(a);
            //else
            {
               if(a < amax) N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1));
               else  N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1)) + N_est(t-
1,a)*exp(-M - sf(a)*F_est(t-1));

            }

         }


      }

            S_est(t) = sum(elem_prod(elem_prod(N_est(t),m),W));
            C_est(t) =
elem_prod(N_est(t),elem_prod(elem_div(sf*F_est(t),M+sf*F_est(t)),(1.-exp(-M-
sf*F_est(t)))));
            I_est(t) = exp(log_qest) * elem_prod(ss, N_est(t));
            C_est_tot(t) = sum(elem_prod(C_est(t),W));

            pI_est(t) =  I_est(t) / sum(I_est(t));
            pC_est(t) = C_est(t) / sum(C_est(t));
         }
      //cout << N_est << endl;
      //exit(1);
         S_term = S_est(years);



FUNCTION evaluate_objective_function
```

```
        NLL = lognorm_nll(C_obs_tot, C_est_tot, C_var) +
lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) +
          multinom_nll(pC_obs, pC_est, ESS_C) + multinom_nll(pI_obs, pI_est,
ESS_I);
      //cout << "neg LL 1 " << lognorm_nll(C_obs_tot, C_est_tot, C_var) << endl <<
"neg LL 2 " << lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) << endl
       //    << "neg LL 3 " <<  multinom_nll(pC_obs, pC_est, ESS_C) << endl << "neg
LL 4 " <<  multinom_nll(pI_obs, pI_est, ESS_I) << endl;

        // lognormal NLL = n*ln(sigma)+0.5∑[(ln(obs)-ln(est))^2 / sigma^2]
        // where sigma^2 = ln(CV^2+1)
        // multionomial NLL = -ESS * ∑obs_p * ln(est_p)

FUNCTION get_spr_brps

        double lambda = (sqrt(5.) - 1.)/2.;
        int jmax=200;

        dvariable SPR_targ;
        SPR_targ = 0.35;
  //SPR_targ = SPR_lim;
        FL = 0.;
        FU = 2.;
        D_dev_targ = pow(0.0005,2);
        D_dev = 1;

        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        N_spr(a_r) = 1.;

        for(a = a_r+1; a <=amax; a++)
  {
                if(a<amax) N_spr(a) = N_spr(a-1) * exp(-M);
     else N_spr(a) = N_spr(a-1) * exp(-M) / (1.-exp(-M));
  }

  //N_spr(amax) += (1.-exp(-M));

        SPR_0 = sum(elem_prod(W,elem_prod(N_spr,m)));

        for(int j=1; j<=jmax; j++){

                for(a = a_r+1; a <=amax; a++)
     {
        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F1);
```

```
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1) * F1) / (1.-exp(-M -
sf(amax)*F1));
      }


      SPR_1 = sum(elem_prod(W,elem_prod(N_spr,m)));

                  //cout << "1" << " " << a <<" " <<  N_spr(a) << endl;


                  for(a = a_r+1; a <=amax; a++)
      {
                        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2) / (1.-exp(-M -
sf(amax)*F2));
         //cout << "1" << " " << a <<" " <<  " " << sf(a) << " " << N_spr(a) << endl;

      }

      //cout <<F2 << " " <<  N_spr << endl;

      //N_spr(amax) /= (1.-exp(-M -sf(amax)*F2));

      //cout << "2" << " " << a <<" " <<  N_spr(a) << endl;

      SPR_2 = sum(elem_prod(W,elem_prod(N_spr,m)));


                        DF2 = pow(SPR_targ - SPR_2/SPR_0,2);
                        DF1 = pow(SPR_targ - SPR_1/SPR_0,2);

                        D_dev = DF1 + DF2;

                              if(DF2 > DF1) {
                                    Est_F = F1;
                                    SPR = SPR_1;
                                    FU = F2;
                                    FL = FL;
                                    F2 = F1;
                                    F1 = FL + (1.-lambda)*(FU-FL);
                                    }

                              else if(DF1 > DF2) {
                                    Est_F = F2;
                                    SPR = SPR_2;
                                    FU = FU;
```

```
                FL = F1;
                F1 = F2;
                F2 = FU - (1.-lambda)*(FU - FL);
                }

        else{
                if(D_dev > D_dev_targ){
                        Est_F = F1;
                        SPR = SPR_1;
                        FU = 1.1*FU;
                        FL = 0.9 * FL;
                        F2 = FU - (1.-lambda)*(FU-FL);
                        F1 = FL + (1.-lambda)*(FU-FL);
                        }

                else{
                        Est_F = F1;
                        break;
                        }
                }

        if(D_dev <= D_dev_targ){
                if(DF1 < DF2){
                        Est_F = F1;
                        SPR = SPR_1;
                        }
                else{
                        Est_F = F2;
                        SPR = SPR_2;
                        }
                break;
                }

        //cout <<j<<" " <<F1 << " " << SPR_1/SPR_0 <<" " << F2 <<
" " << SPR_2/SPR_0 << " " << Est_F << endl;

        }//end for loop

        Flim = Est_F;
        Slim = SPR * sum(mfexp(log_Rest))/size(log_Rest);



FUNCTION dvariable ofl_calc(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Est_Flim)
```

```
        dvar_vector N_ofl(a_r,amax);
        dvariable OFL;

        N_ofl(a_r) = Rec;

        for(a=a_r+1; a<=amax; a++)
        {
                if(a < amax) N_ofl(a) =  N_term(a-1) * exp(-M - sel_f(a-1) * F_cur);
      else N_ofl(a) = N_term(a-1) * exp(-M - sel_f(a-1) * F_cur) + N_term(a) * exp(-
M - sel_f(a) * F_cur);
   }

        OFL =
sum(elem_prod(W,elem_prod(N_ofl,elem_prod(elem_div(sel_f*Est_Flim,M+sel_f*
Est_Flim),(1.-mfexp(-M-sel_f*Est_Flim))))));
        /*
        cout << F_cur << endl;
        cout << Est_Flim << endl;
        cout << M << endl;
        cout << W << endl;
        cout << sel_f << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << OFL << endl;
        */
        return(OFL);

      cout << "OFL"<<OFL<<endl;

//FUNCTION dvariable ofl_proj(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Frec_cur, dvariable Est_Flim, dvariable Est_Ftarg,
data_int pyrs, int proj_switch)
FUNCTION dvar_vector ofl_proj(dvar_vector N_term, named_dvar_vector sel_f,
prevariable Rec, prevariable F_cur, prevariable Est_Flim, data_int pyrs, data_int
proj_switch, data_int rp)
     //changed to dvar_vector ASylvia
   //cout << pyrs << " " << proj_years << endl;

        dvar_matrix N_ofl(1,proj_years,a_r,amax);
        dvar_vector OFL_vec(1,proj_years);
        dvar_vector ctarg_vec(1,proj_years);
   dvar_vector Zt(a_r,amax);
   dvar_vector Z_cur(a_r,amax);
        dvar_vector Z_lim(a_r,amax);
        dvar_vector Z_targ(a_r,amax);
   dvariable OFL_cur;
```

```
        Z_cur =  M + sel_f * F_cur;
        Z_lim =  M + sel_f * Est_Flim;
      //cout<<"1"<<endl;
        //Z_targ =  M + sel_f * Est_Ftarg;

   //cout << Z_cur << endl;
   //cout << Z_lim << endl;
   //cout << Z_targ << endl;
   //cout << 3 << endl;
   //OFL_cur =
sum(elem_prod(W,elem_prod(N_term,elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
   //cout << N_term << endl;
   // replace the terminal estimate of recruitment with the mean value to minimize the
effects of uncertainty

   if(rp>=2)
   {
      N_term(a_r) = Rec;
   }


   //cout << "pyrs: " << pyrs << endl;
   for(t = 1; t<=pyrs; t++)
   {
                N_ofl(t,a_r) = Rec;


        for(a=a_r+1; a<=amax; a++)
        {

                        if(t==1)
                        {
                                Zt = Z_cur;
                        if(a < amax) N_ofl(t,a) =  N_term(a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_term(a-1) * exp(-Zt(a-1)) +  N_term(a) *
exp(-Zt(a));
                        }
                        else
                        {
                        Zt = Z_lim;
                        if(a < amax) N_ofl(t,a) =  N_ofl(t-1,a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_ofl(t-1,a-1) * exp(-Zt(a-1)) +  N_ofl(t-1,a)
* exp(-Zt(a));
                        }
```

```
        }


                OFL_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
                //ctarg_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Ftarg,Z_targ),
(1.-mfexp(-Z_targ))))));
      //cout << t << N_ofl(t) << endl;
      //cout << Est_Flim << " " << OFL_vec(t) << endl;
   }
      //cout <<  N_ofl << endl;
      //cout << "est_flim: " << Est_Flim << " OFL_vec " << OFL_vec << endl;

    // cout<<"2"<<endl;
        p_int = OFL_years;
        //cout<<proj_years<<endl;
        for(t = 1; t<= OFL_years; t++)
        {
       //cout<<"p_int "<<p_int<<endl;
                if(proj_switch <=1)
                {
                        OFL(t) = OFL_vec(pyrs);
                        //ctarg(t) = ctarg_vec(pyrs);
                }
                else if(proj_switch==2)
                {
                        OFL(t) = OFL_vec(p_int);
                        //ctarg(t) = ctarg(p_int);
                }

                p_int--;
        }
  //    cout<<"3"<<endl;
        //cout << proj_switch << " " << proj_years << " " << OFL << endl;


        /*
        cout << Est_Flim << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << sum(elem_prod(elem_prod(N_ofl,m),W))<< " " << S_est(years) <<
endl;
        cout << sum(elem_prod(W,N_ofl)) <<  endl;
        cout << (1.-exp(-M-sel_f*Est_Flim)) << endl;
```

```
        cout << elem_div(sel_f*Est_Flim,M+sel_f*Est_Flim) << endl;
        cout << sf << endl;
        cout << sf_true << endl;
        cout << N_term << endl << N_ofl << endl;
        */
        //cout << OFL << endl;
        return(OFL);  //turned back into code ASylvia


FUNCTION write_term_output
   ofstream sae("term_est.txt");
   {
      for(tt = 0; tt <=(SA_interval-1);tt++)
      {
         sae << S_est(years-tt) << " " <<  exp(log_Rest(years-tt)) << " " <<
F_est(years-tt) << endl;


      }
      sae <<Slim<< endl;
      sae <<Flim<< endl;
      sae <<OFL<< endl;
      sae << mfexp(log_sf_50) << " " << mfexp(log_sf_slope) << endl;
      sae << objective_function_value::gmax << endl;
   }


FUNCTION write_final_estimates

            if(lh==1)
            {
                  ofstream ofs_fe("final_assessment_estimates_fast.txt",ios::app);
                  {
                  if(iter==1)
                  {
               ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                  }

                        for(t=first_data_yr;t<=years;t++)
                  {

                              ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
```

```
                }
                }

            }


          if(lh==2)
            {
                ofstream ofs_fe("final_assessment_estimates_medium.txt",ios::app);
                {
                if(iter==1)
                {
              ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                }

                        for(t=first_data_yr;t<=years;t++)
                {

                                ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
                }

            }

          if(lh==3)
            {
                ofstream ofs_fe("final_assessment_estimates_slow.txt",ios::app);
                {
                if(iter==1)
                {
              ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                }

                        for(t=first_data_yr;t<=years;t++)
                {

                                ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                }
```

```
                }

            }


FUNCTION dvariable lognorm_nll(dvar_vector obs, dvar_vector est, data_number
var)    //Function to calculate a lognormal negative log likelihood
  dvariable L;
  L=norm2(log(obs+0.1)-log(est+0.1));
  L = 0.5*(size(obs))*log(var)+0.5*L/var;



  //cout << endl << L << endl<< endl;
  return(L);

FUNCTION dvariable multinom_nll(dvar_matrix obs_p, dvar_matrix est_p,
dvar_vector eff_size)            // multionomial NLL = -ESS * ∑obs_p * ln(est_p)
        dvariable ML;
        ML = 0.0;
    // make this sum and not rowsum
    // -eff_size change to a value and then change rowsum to sum
        ML = -eff_size * rowsum(elem_prod(obs_p,log(est_p+.001)));

    //cout << ML << endl;
    return(ML);

FUNCTION double size(dvar_vector obs)              // function used to return the
number of elements in a vector
  return(double(obs.indexmax()-obs.indexmin()+1));

FINAL_SECTION


    get_spr_brps();
   // cout<<"1.0"<<endl;
    if(cr <=7) Ftarg = Flim;
    else if(cr==8) Ftarg = 0.75*Flim;

    //cout << rp_switch << endl;
    //cout << mfexp(log_Rest) << endl;
    //cout << mfexp(log_Rest(years-5,years-1)) << endl;

    if(rp_switch<=2) mu_Rec = sum(mfexp(log_Rest))/size(log_Rest);
    else if(rp_switch==3) mu_Rec = sum(mfexp(log_Rest(years-5,years-1)))/5.;
   // cout<<"1.2"<<endl;
```

```
//ofl_calc(N_est(years),sf,sum(mfexp(log_Rest))/size(log_Rest),F_est(years),Flim);
   //
cout<<"N_est"<<N_est(years)<<endl<<"sf"<<sf<<endl<<"mu_Rec"<<mu_Rec<<en
dl<<"F_est"<<F_est(years)<<endl<<"Ftarg"<<Ftarg<<endl<<"proj_years"<<proj_ye
ars<<endl<<"proj_switch"<<proj_switch<<endl<<"rp_switch"<<rp_switch<<endl;
   // cout<<"proj_years " << proj_years<<endl;

OFL=ofl_proj(N_est(years),sf,mu_Rec,F_est(years),Ftarg,proj_years,proj_switch,rp_
switch);
    //added OFL= in front of ofl_proj function
   // cout<<"1.1"<<endl;
   write_term_output();
   //write_r_est();
   //if(sa_num==final_sa) write_final_estimates();

   // get rid of output here, and combine with other files
   /*
   ofstream brp("brp.txt");
   {
      brp << Flim << endl;
      brp << Slim << endl;
   }
   */

   //cout << "estimates" << endl;
   //cout << Flim << " " << Slim << endl;

   //cout << OFL_est << endl;
   //write_ofl();
         //write_sa_output();
         //write_r_est();
         //write_selex();

REPORT_SECTION


         //write_obs_est();
         //write_init_N();
         //write_SR_output();
         //cout << "est model done!" << endl;
         //write_ssb_output();


RUNTIME_SECTION
```

```
maximum_function_evaluations 4000  // increase the # of evaluations
```

Code III.6: Chapter 2 ADMB Code. Estimation model 4, code for Lag reduction method 3: The stock assessment model was called at regular intervals to estimate stock biomass and reference points for management using a statistical catch-at-age model. Stock assessment model 4 used a decresed effective sample size for the composition in the terminal year of the catch and the survey.

```
 //Andrea Sylvia
//
// This is ESTIMATION MODEL 4!!!!!!!!!!
//

DATA_SECTION

  //
**********************************************************************
*****
  //  DATA READ IN FROM  em_cf.dat
  //
**********************************************************************
*****

  init_int lh                                   // life history index
  init_int cr
  init_int iter                                 // which iteration we are on in the
assessment model
  init_int sa_num                               // which stock assessment we are
conducting (within each iteration)
  init_int final_sa                     // how many assessments are to be done?
  init_int years                        // # of years in the model
  init_int first_data_yr                // first year of data collection
  init_int proj_years
  init_int proj_switch
  init_int rp_switch
  init_int OFL_years
 // !! cout<<OFL_years<<endl;
 // !! exit(1);
  init_int amax                                 // # of age classes in model
  init_int a_r                                          // # age at recruitment to
pop
  init_number M                                         // Natural mortality
  init_vector W(a_r,amax)                                       // Weight-at-age
  init_vector m(a_r,amax)                               // maturity-at-age

  init_matrix Catch(first_data_yr,years,a_r,amax)               // Observed catch
at age
```

```
 init_matrix N_Index(first_data_yr,years,a_r,amax)          // Observed adbundance
at age
 init_number C_var                                                  // Var. in catch
 init_number I_var                                                  // Var. in index

 init_int ESS                                                       // survey ESS
 init_vector log_R_true(first_data_yr,years)                // True log(recruitments)
 init_vector log_F_true(first_data_yr,years)                // True log(recruitments)
 init_vector S_true(first_data_yr,years)
 init_vector log_Ninit_true(a_r+1,amax)                     // True log(initial
population size)
 init_number log_q_true                                             // true log(catchability)
 init_number log_sf50_true                                  // age @ 50% selectivity
in fishery
 init_number log_ss50_true                  // age @ 50% selectivity in the survey
 init_number log_sf_slope                                   // age @ 50% selectivity
in fishery
 init_number log_ss_slope                                   // age @ 50% selectivity
in fishery
 //init_vector sf_true(1,amax)                              // true sele
 //init_vector ss_true(1,amax)
 init_number SPR_lim
 init_int EOF_flag

 int t
                // year index
 int tt
 int a
 int p_int
                // age index
 int SA_interval
 !! SA_interval = OFL_years;

 //!! cout << "2"<< endl;

// Change the phases to 1 for all.
// feed in F values and start pop at true values
// take out variable C_obs and I_obs


PARAMETER_SECTION
 init_bounded_number log_mean_R(0.,20.,1)           //log mean recruitment
 init_bounded_dev_vector log_R_dev(first_data_yr,years,-20.,20.,1)// log deviations
for mean recruitment
   init_bounded_vector log_Ninit(a_r+1,amax,0.,20.,1)          // mean initial pop
size (ages
```

```
 //init_bounded_dev_vector log_Ninit_dev(2,amax,-20.,20.,1)
   init_bounded_number log_mean_F(-10.,5.,1)                //log of mean F
 init_bounded_dev_vector log_F_dev(first_data_yr,years,-10,10,1)//log deviations
from mean F
 init_bounded_number log_qest(-15.,-5.,1)          // log(estimated q)
   init_bounded_number log_sf_50(-2,3,1)                  // log(fishery age @ 50%
selectivity)
 init_bounded_number log_sf_slope(-5.,5.,1)        // log(fishery selectivity slope)
 init_bounded_number log_ss_50(-2.,3,1)            // log(survey age @ 50%
selectivity)
 init_bounded_number log_ss_slope(-5.,5.,1)        // log(fishery selectivity slope)
   matrix N_est(first_data_yr,years,a_r,amax)      // Est. abundance at age
 vector log_Rest(first_data_yr,years)
 matrix C_obs(first_data_yr,years,a_r,amax)                    // Est. catch at age
 matrix C_est(first_data_yr,years,a_r,amax)                    // Est. catch at age
 matrix I_obs(first_data_yr,years,a_r,amax)                    // Est. abundance
index
 matrix I_est(first_data_yr,years,a_r,amax)                    // Est. abundance
index
 matrix pI_est(first_data_yr,years,a_r,amax)           // Est. proportion of Index
abundance
 matrix pC_est(first_data_yr,years,a_r,amax)           // Est. proportion of catch
 matrix pI_obs(first_data_yr,years,a_r,amax)           // Proportion of Index
abundance
 matrix pC_obs(first_data_yr,years,a_r,amax)           // Proportion of catch

 vector Ninit(a_r+1,amax)                              // initial pop size
(ages 2-
 vector F_est(first_data_yr,years)                     // Estimated F
 vector ESS_C(first_data_yr,years)                 // Effective sample size for catch
at age
 vector ESS_I(first_data_yr,years)                 // Effective sample size for index
at age
 vector log_R(first_data_yr,years)
  vector I_est_tot(first_data_yr,years)            // Est. abundance index
 vector C_est_tot(first_data_yr,years)                                 // Est. total
catch
 vector C_obs_tot(first_data_yr,years)                                 // Est. total
catch
 vector S_est(first_data_yr,years)                 // Est. spawning biomass
 vector ss(a_r,amax)                               // selex at age in survey
 vector sf(a_r,amax)                               // selex at age in fishery

 vector OFL(1,OFL_years)
 vector ctarg(1,OFL_years)
```

```
vector N_spr(a_r,amax)
number SPR
number Slim
number FL
number FU
number F1
number F2
number Est_F
number SPR_1
number SPR_2
number DF1
number DF2
number D_dev
number D_dev_targ
number SPR_0
number Flim
number Slim_temp
number Flim_temp
number OFL_est
number Ftarg
number mu_Rec

sdreport_number S_term     // terminal spawning biomass

objective_function_value NLL


        LOCAL_CALCS

                if(EOF_flag != 12345)
                {
                        cout << "----------------------------------------"<<endl;
                        cout <<" Data not read in properly to em_cf.tpl !"<< endl;
                        cout << "----------------------------------------"<<endl;
        cout << "# iteration" << endl;
                        cout << iter << endl;
                        cout << "# stock assessment number" << endl;
                        cout << sa_num << endl;
                        cout <<"# end yr "<< endl;
                        cout << years << endl;
                        cout <<"# first year of data  "<< endl;
                        cout << first_data_yr << endl;
                        cout <<" # max age "<< endl;
                        cout << amax << endl;
                        cout <<" # age at recruitment  "<< endl;
                        cout << a_r << endl;
```

```
                cout <<" # Nat. mort  "<< endl;
                cout << M << endl;
                cout <<" # Weight "<< endl;
                cout << W << endl;
                cout <<" #  maturity"<< endl;
                cout << m << endl;
                cout <<" # Catch "<< endl;
                cout << Catch << endl;
                cout <<" # N index "<< endl;
                cout << N_Index << endl;
                cout <<" #  Var in C"<< endl;
                cout << C_var << endl;
                cout <<" # Var in I "<< endl;
                cout << I_var << endl;
                cout <<" # ESS "<< endl;
                cout << ESS << endl;
                cout <<" #  log (R)"<< endl;
                cout << log_R_true << endl;
                cout <<" #  log (F)"<< endl;
                cout << log_F_true << endl;
                cout <<" #  log (N init)"<< endl;
                cout << log_Ninit_true << endl;
                cout <<" #  log(q)"<< endl;
                cout << log_q_true << endl;
                cout <<" #  age @ 50% selex in fishery"<< endl;
                cout << log_sf50_true << endl;
                cout <<" #  age at 50% selex in survey"<< endl;
                cout << log_ss50_true << endl;
                cout <<" #  fishery selex"<< endl;
                cout << log_sf_slope << endl;
                cout <<" #  survey selex"<< endl;
                cout << log_ss_slope << endl;
                cout <<" #  EOF flag"<< endl;
                cout << EOF_flag << endl;
                exit(1);
        }

//cout << log_R_true << endl;
//cout << mfexp(log_R_true(years-1,years)) << endl;

ESS_C = ESS;
ESS_I = ESS;

        log_mean_R=sum(log_R_true)/double(years-first_data_yr+1.);
        log_R_dev = log_R_true-log_mean_R;
        log_qest = log_q_true;
```

```
                //log_mean_F = -0.35;
        log_mean_F = sum(log_F_true)/double(years-first_data_yr+1.);
                log_F_dev = log_F_true-log_mean_F;
        log_Ninit = log_Ninit_true;

                //log_sf_50 = log(sf50_true);
                //log_ss_50 = log(ss50_true);

                log_sf_50 = log_sf50_true;
                log_ss_50 = log_ss50_true;



                get_data();



        END_CALCS

PROCEDURE_SECTION

        get_selectivities();
        get_estimates();
        evaluate_objective_function();
        //cout << Flim << " " << OFL_est << " " << mfexp(log_F_est(years)) << " "
<< sum(Catch(years)) << endl;


FUNCTION get_selectivities

  for(a=a_r; a<=amax; a++)
   {
      ss(a) = 1. / (1.+exp(-(double(a)-mfexp(log_ss_50))/mfexp(log_ss_slope)));
      sf(a) = 1. / (1.+exp(-(double(a)-mfexp(log_sf_50))/mfexp(log_sf_slope)));

        }


FUNCTION get_data
        for(t=first_data_yr;t<=years;t++)
   {
                C_obs(t) = Catch(t);
                I_obs(t) = N_Index(t);
                //log_R(t) = log_R_true(t);
      C_obs_tot(t) = sum(elem_prod(C_obs(t),W));
```

```
      pI_obs(t) = I_obs(t) / sum(I_obs(t));
                  pC_obs(t) = C_obs(t) / sum(C_obs(t));
         }

FUNCTION get_estimates


         F_est=exp(log_mean_F+log_F_dev);
         log_Rest=(log_mean_R+log_R_dev);

   N_est(first_data_yr,a_r) = exp(log_Rest(first_data_yr));
   //cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   N_est(first_data_yr)((a_r+1),amax) = exp(log_Ninit);
     //  cout << N_est(first_data_yr)((a_r+1),amax) << endl;
   //exit(1);
   //cout << exp(log_Ninit) << endl;

   S_est(first_data_yr) = sum(elem_prod(elem_prod(N_est(first_data_yr),m),W));
   C_est(first_data_yr) =
elem_prod(N_est(first_data_yr),elem_prod(elem_div(sf*F_est(first_data_yr),M+sf*F
_est(first_data_yr)),(1.-exp(-M-sf*F_est(first_data_yr)))));
   I_est(first_data_yr) = exp(log_qest) * elem_prod(ss, N_est(first_data_yr));
   C_est_tot(first_data_yr) = C_est(first_data_yr) * W;
   pI_est(first_data_yr) =  I_est(first_data_yr) / sum(I_est(first_data_yr));
   pC_est(first_data_yr) = C_est(first_data_yr) / sum(C_est(first_data_yr));


   for(t=first_data_yr+1; t<=years; t++){

      for(a=a_r; a<=amax; a++)
      {
         if(a==a_r) N_est(t,a) = exp(log_Rest(t));
         else
         {
            //if(t==first_data_yr) N_est(t,a) = Ninit(a);
            //else
            {
               if(a < amax) N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1));
               else  N_est(t,a) = N_est(t-1,a-1)*exp(-M - sf(a-1)*F_est(t-1)) + N_est(t-
1,a)*exp(-M - sf(a)*F_est(t-1));

            }

         }
```

```
      }

            S_est(t) = sum(elem_prod(elem_prod(N_est(t),m),W));
            C_est(t) =
elem_prod(N_est(t),elem_prod(elem_div(sf*F_est(t),M+sf*F_est(t)),(1.-exp(-M-
sf*F_est(t)))));
            I_est(t) = exp(log_qest) * elem_prod(ss, N_est(t));
            C_est_tot(t) = sum(elem_prod(C_est(t),W));

            pI_est(t) =  I_est(t) / sum(I_est(t));
            pC_est(t) = C_est(t) / sum(C_est(t));
          }
      //cout << N_est << endl;
      //exit(1);
         S_term = S_est(years);



FUNCTION evaluate_objective_function


         NLL = lognorm_nll(C_obs_tot, C_est_tot, C_var) +
lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) +
           multinom_nll(pC_obs, pC_est, ESS_C) + multinom_nll(pI_obs, pI_est,
ESS_I);
      //cout << "neg LL 1 " << lognorm_nll(C_obs_tot, C_est_tot, C_var) << endl <<
"neg LL 2 " << lognorm_nll(rowsum(I_obs), rowsum(I_est), I_var) << endl
       //    << "neg LL 3 " <<  multinom_nll(pC_obs, pC_est, ESS_C) << endl << "neg
LL 4 " <<  multinom_nll(pI_obs, pI_est, ESS_I) << endl;

         // lognormal NLL = n*ln(sigma)+0.5∑[(ln(obs)-ln(est))^2 / sigma^2]
         // where sigma^2 = ln(CV^2+1)
         // multionomial NLL = -ESS * ∑obs_p * ln(est_p)

FUNCTION get_spr_brps

         double lambda = (sqrt(5.) - 1.)/2.;
         int jmax=200;

         dvariable SPR_targ;
         SPR_targ = 0.35;
   //SPR_targ = SPR_lim;
         FL = 0.;
         FU = 2.;
         D_dev_targ = pow(0.0005,2);
         D_dev = 1;
```

```
        F2 = FU - (1.-lambda)*(FU-FL);
        F1 = FL + (1.-lambda)*(FU-FL);

        N_spr(a_r) = 1.;

        for(a = a_r+1; a <=amax; a++)
   {
                if(a<amax) N_spr(a) = N_spr(a-1) * exp(-M);
      else N_spr(a) = N_spr(a-1) * exp(-M) / (1.-exp(-M));
   }

   //N_spr(amax) += (1.-exp(-M));

        SPR_0 = sum(elem_prod(W,elem_prod(N_spr,m)));

        for(int j=1; j<=jmax; j++){

                for(a = a_r+1; a <=amax; a++)
      {
        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F1);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1) * F1) / (1.-exp(-M -
sf(amax)*F1));
      }


      SPR_1 = sum(elem_prod(W,elem_prod(N_spr,m)));

                //cout << "1" << " " << a <<" " <<  N_spr(a) << endl;


                for(a = a_r+1; a <=amax; a++)
      {
                        if(a < amax) N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2);
        else N_spr(a) = N_spr(a-1) * exp(-M -sf(a-1)*F2) / (1.-exp(-M -
sf(amax)*F2));
        //cout << "1" << " " << a <<" " <<  " " << sf(a) << " " << N_spr(a) << endl;

      }

      //cout <<F2 << " " <<  N_spr << endl;

      //N_spr(amax) /= (1.-exp(-M -sf(amax)*F2));

      //cout << "2" << " " << a <<" " <<  N_spr(a) << endl;
```

```
SPR_2 = sum(elem_prod(W,elem_prod(N_spr,m)));


                DF2 = pow(SPR_targ - SPR_2/SPR_0,2);
                DF1 = pow(SPR_targ - SPR_1/SPR_0,2);

                D_dev = DF1 + DF2;

                        if(DF2 > DF1) {
                                Est_F = F1;
                                SPR = SPR_1;
                                FU = F2;
                                FL = FL;
                                F2 = F1;
                                F1 = FL + (1.-lambda)*(FU-FL);
                                }

                        else if(DF1 > DF2) {
                                Est_F = F2;
                                SPR = SPR_2;
                                FU = FU;
                                FL = F1;
                                F1 = F2;
                                F2 = FU - (1.-lambda)*(FU - FL);
                                }

                        else{
                                if(D_dev > D_dev_targ){
                                        Est_F = F1;
                                        SPR = SPR_1;
                                        FU = 1.1*FU;
                                        FL = 0.9 * FL;
                                        F2 = FU - (1.-lambda)*(FU-FL);
                                        F1 = FL + (1.-lambda)*(FU-FL);
                                        }

                                else{
                                        Est_F = F1;
                                        break;
                                        }
                                }

                        if(D_dev <= D_dev_targ){
                                if(DF1 < DF2){
                                        Est_F = F1;
                                        SPR = SPR_1;
```

```
                                        }
                                else{
                                        Est_F = F2;
                                        SPR = SPR_2;
                                        }
                                break;
                                }

                        //cout <<j<<" " <<F1 << " " << SPR_1/SPR_0 <<" " << F2 <<
" " << SPR_2/SPR_0 << " " << Est_F << endl;

                        }//end for loop

                        Flim = Est_F;
                        Slim = SPR * sum(mfexp(log_Rest))/size(log_Rest);



FUNCTION dvariable ofl_calc(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Est_Flim)

        dvar_vector N_ofl(a_r,amax);
        dvariable OFL;

        N_ofl(a_r) = Rec;

        for(a=a_r+1; a<=amax; a++)
        {
                if(a < amax) N_ofl(a) =  N_term(a-1) * exp(-M - sel_f(a-1) * F_cur);
      else N_ofl(a) = N_term(a-1) * exp(-M - sel_f(a-1) * F_cur) + N_term(a) * exp(-
M - sel_f(a) * F_cur);
   }

        OFL =
sum(elem_prod(W,elem_prod(N_ofl,elem_prod(elem_div(sel_f*Est_Flim,M+sel_f*
Est_Flim),(1.-mfexp(-M-sel_f*Est_Flim))))));
        /*
        cout << F_cur << endl;
        cout << Est_Flim << endl;
        cout << M << endl;
        cout << W << endl;
        cout << sel_f << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << OFL << endl;
        */
```

```
        return(OFL);

      cout << "OFL"<<OFL<<endl;

//FUNCTION dvariable ofl_proj(dvar_vector N_term, dvar_vector sel_f, dvariable
Rec, dvariable F_cur, dvariable Frec_cur, dvariable Est_Flim, dvariable Est_Ftarg,
data_int pyrs, int proj_switch)
FUNCTION dvar_vector ofl_proj(dvar_vector N_term, named_dvar_vector sel_f,
prevariable Rec, prevariable F_cur, prevariable Est_Flim, data_int pyrs, data_int
proj_switch, data_int rp)
     //changed to dvar_vector ASylvia
   //cout << pyrs << " " << proj_years << endl;

        dvar_matrix N_ofl(1,proj_years,a_r,amax);
        dvar_vector OFL_vec(1,proj_years);
        dvar_vector ctarg_vec(1,proj_years);
   dvar_vector Zt(a_r,amax);
   dvar_vector Z_cur(a_r,amax);
        dvar_vector Z_lim(a_r,amax);
        dvar_vector Z_targ(a_r,amax);
   dvariable OFL_cur;

        Z_cur =  M + sel_f * F_cur;
        Z_lim =  M + sel_f * Est_Flim;
      //cout<<"1"<<endl;
        //Z_targ =  M + sel_f * Est_Ftarg;

   //cout << Z_cur << endl;
   //cout << Z_lim << endl;
   //cout << Z_targ << endl;
   //cout << 3 << endl;
   //OFL_cur =
sum(elem_prod(W,elem_prod(N_term,elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
   //cout << N_term << endl;
   // replace the terminal estimate of recruitment with the mean value to minimize the
effects of uncertainty

   if(rp>=2)
   {
      N_term(a_r) = Rec;
   }


   //cout << "pyrs: " << pyrs << endl;
   for(t = 1; t<=pyrs; t++)
```

```
   {
                N_ofl(t,a_r) = Rec;


        for(a=a_r+1; a<=amax; a++)
        {

                        if(t==1)
                        {
                                Zt = Z_cur;
                        if(a < amax) N_ofl(t,a) =  N_term(a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_term(a-1) * exp(-Zt(a-1)) +  N_term(a) *
exp(-Zt(a));
                        }
                        else
                        {
                        Zt = Z_lim;
                        if(a < amax) N_ofl(t,a) =  N_ofl(t-1,a-1) * exp(-Zt(a-1));
                        else N_ofl(t,a) = N_ofl(t-1,a-1) * exp(-Zt(a-1)) +  N_ofl(t-1,a)
* exp(-Zt(a));
                        }
        }


                OFL_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Flim,Z_lim),(
1.-mfexp(-Z_lim))))));
                //ctarg_vec(t) =
sum(elem_prod(W,elem_prod(N_ofl(t),elem_prod(elem_div(sel_f*Est_Ftarg,Z_targ),
(1.-mfexp(-Z_targ))))));
      //cout << t << N_ofl(t) << endl;
      //cout << Est_Flim << " " << OFL_vec(t) << endl;
   }
      //cout <<  N_ofl << endl;
      //cout << "est_flim: " << Est_Flim << " OFL_vec " << OFL_vec << endl;

    // cout<<"2"<<endl;
        p_int = OFL_years;
        //cout<<proj_years<<endl;
        for(t = 1; t<= OFL_years; t++)
        {
       //cout<<"p_int "<<p_int<<endl;
                if(proj_switch <=1)
                {
                        OFL(t) = OFL_vec(pyrs);
                        //ctarg(t) = ctarg_vec(pyrs);
```

```
                }
                else if(proj_switch==2)
                {
                        OFL(t) = OFL_vec(p_int);
                        //ctarg(t) = ctarg(p_int);
                }

                p_int--;
        }
  //    cout<<"3"<<endl;
        //cout << proj_switch << " " << proj_years << " " << OFL << endl;


        /*
        cout << Est_Flim << endl;
        cout << N_term << endl;
        cout << N_ofl << endl;
        cout << sum(elem_prod(elem_prod(N_ofl,m),W))<< " " << S_est(years) <<
endl;
        cout << sum(elem_prod(W,N_ofl)) <<  endl;
        cout << (1.-exp(-M-sel_f*Est_Flim)) << endl;
        cout << elem_div(sel_f*Est_Flim,M+sel_f*Est_Flim) << endl;
        cout << sf << endl;
        cout << sf_true << endl;
        cout << N_term << endl << N_ofl << endl;
        */
        //cout << OFL << endl;
        return(OFL);  //turned back into code ASylvia


FUNCTION write_term_output
   ofstream sae("term_est.txt");
   {
      for(tt = 0; tt <=(SA_interval-1);tt++)
      {
         sae << S_est(years-tt) << " " <<  exp(log_Rest(years-tt)) << " " <<
F_est(years-tt) << endl;


      }
      sae <<Slim<< endl;
      sae <<Flim<< endl;
      sae <<OFL<< endl;
      sae << mfexp(log_sf_50) << " " << mfexp(log_sf_slope) << endl;
      sae << objective_function_value::gmax << endl;
   }
```

```
FUNCTION write_final_estimates

            if(lh==1)
            {
                  ofstream ofs_fe("final_assessment_estimates_fast.txt",ios::app);
                  {
                  if(iter==1)
                  {
                ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                  }

                              for(t=first_data_yr;t<=years;t++)
                   {

                                    ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                   }
                  }

            }


          if(lh==2)
            {
                  ofstream ofs_fe("final_assessment_estimates_medium.txt",ios::app);
                  {
                  if(iter==1)
                  {
                ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                  }

                              for(t=first_data_yr;t<=years;t++)
                   {

                                    ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                   }
                  }
```

```
            }

          if(lh==3)
             {
                  ofstream ofs_fe("final_assessment_estimates_slow.txt",ios::app);
                  {
                  if(iter==1)
                  {
                ofs_fe<< "i" << " " << "year"<< " " << "true_S"<<"     "<< "estimated_S"
<< " " <<  "true_R"<<" "<< "estimated_R" << " " << "true_F"<<" "<< "estimated_F"
<< endl;
                  }

                           for(t=first_data_yr;t<=years;t++)
                   {

                                    ofs_fe  << iter <<" "<< t << " " << S_true(t) << " " <<
S_est(t) << " "<< exp(log_R_true(t)) << " " << exp(log_Rest(t)) << " "  << " " <<
exp(log_F_true(t)) << " " << F_est(t) <<endl;
                   }
                  }

             }


FUNCTION dvariable lognorm_nll(dvar_vector obs, dvar_vector est, data_number
var)    //Function to calculate a lognormal negative log likelihood
  dvariable L;
  L=norm2(log(obs+0.1)-log(est+0.1));
  L = 0.5*(size(obs))*log(var)+0.5*L/var;



  //cout << endl << L << endl<< endl;
  return(L);

FUNCTION dvariable multinom_nll(dvar_matrix obs_p, dvar_matrix est_p,
dvar_vector eff_size)            // multionomial NLL = -ESS * ∑obs_p * ln(est_p)
          dvariable ML;
          ML = 0.0;
    // make this sum and not rowsum
    // -eff_size change to a value and then change rowsum to sum
          ML = -eff_size * rowsum(elem_prod(obs_p,log(est_p+.001)));

    //cout << ML << endl;
```

```
   return(ML);

FUNCTION double size(dvar_vector obs)              // function used to return the
number of elements in a vector
  return(double(obs.indexmax()-obs.indexmin()+1));

FINAL_SECTION


   get_spr_brps();
  // cout<<"1.0"<<endl;
   if(cr <=7) Ftarg = Flim;
   else if(cr==8) Ftarg = 0.75*Flim;

   //cout << rp_switch << endl;
   //cout << mfexp(log_Rest) << endl;
   //cout << mfexp(log_Rest(years-5,years-1)) << endl;

   if(rp_switch<=2) mu_Rec = sum(mfexp(log_Rest))/size(log_Rest);
   else if(rp_switch==3) mu_Rec = sum(mfexp(log_Rest(years-5,years-1)))/5.;
  // cout<<"1.2"<<endl;


//ofl_calc(N_est(years),sf,sum(mfexp(log_Rest))/size(log_Rest),F_est(years),Flim);
   //
cout<<"N_est"<<N_est(years)<<endl<<"sf"<<sf<<endl<<"mu_Rec"<<mu_Rec<<en
dl<<"F_est"<<F_est(years)<<endl<<"Ftarg"<<Ftarg<<endl<<"proj_years"<<proj_ye
ars<<endl<<"proj_switch"<<proj_switch<<endl<<"rp_switch"<<rp_switch<<endl;
   // cout<<"proj_years " << proj_years<<endl;

OFL=ofl_proj(N_est(years),sf,mu_Rec,F_est(years),Ftarg,proj_years,proj_switch,rp_
switch);
    //added OFL= in front of ofl_proj function
   // cout<<"1.1"<<endl;
   write_term_output();
   //write_r_est();
   //if(sa_num==final_sa) write_final_estimates();

   // get rid of output here, and combine with other files
   /*
   ofstream brp("brp.txt");
   {
      brp << Flim << endl;
      brp << Slim << endl;
   }
   */
```

```
  //cout << "estimates" << endl;
  //cout << Flim << " " << Slim << endl;

  //cout << OFL_est << endl;
  //write_ofl();
        //write_sa_output();
        //write_r_est();
        //write_selex();

REPORT_SECTION


        //write_obs_est();
        //write_init_N();
        //write_SR_output();
        //cout << "est model done!" << endl;
        //write_ssb_output();


RUNTIME_SECTION
  maximum_function_evaluations 4000  // increase the # of evaluations
```

Bibliography

Alaska Fisheries Science Center. 2013. AFSC groundfish and crab surveys and their role in fisheries management. Executive Summary. Available from: http://www.afsc.noaa.gov/program_reviews/2013/background_materials/Surveys%20Role%20in%20Fisheries%20Management.pdf [accessed December 2014].

Alaska Fisheries Science Center. 2014. Alaska Fisheries Science Center-Alaska Department of Fish and Game -North Pacific Fishery Management Council Stock Assessment Review Process. Available from: www.npfmc.org/wp-content/PDFdocuments/resources/SAFE/AFSCsafeReviewProcess.pdf [accessed December 2014]

Bax, N. J., R. Tilzey, J. Lyle, S. E. Wayte, R. J. Kloser, and A. D. M. Smith. 2003. Providing management advice for deep-sea fisheries: lessons learned from Australia's orange roughy fisheries. Pages 259–272 in R. Shotton, editor. Deep Sea 2003: Conference on the Governance and Management of Deep-sea Fisheries. Food and Agriculture Organization of the United Nations, Queenstown, New Zealand.

Begg G. A., Campana S. E., Fowler A. J., and Suthers I. M. 2005. Otolith research and application: current directions in innovation and implementation. Marine and Freshwater Research, 56: 477.

Brown, C.J., Fulton, E.A., H. P. Possingham and A. J. Richardson. 2012. How long can fisheries management delay action in response to ecosystem and climate change? Ecol. Appl. 22:298-310

Butterworth, D. S. 2007. Why a management procedure approach? Some positives and negatives. ICES Journal of Marine Science. 64:613–617.

Caddy, J. F., and K. L. Cochrane. 2001. A review of fisheries management past and present and some future perspectives for the third millennium. Ocean and Coastal Management 44:653-682

Chen Y., L. Chen and K. I. Stergiou. 2002. Impacts of data quantity on fisheries stock assessment. Aquat. Sci. 65: 92-98.

Coggins L. G., D. C. Gwinn and M. S. Allen 2013. Evaluation of Age–Length Key Sample Sizes Required to Estimate Fish Total Mortality and Growth. Transactions of the American Fisheries Society. 142:3, 832-840

De Leeuw, J.J., W. Dekker, and A. D. Buijse. 2008. Aiming at a moving target, a slow hand fails! 75 years of fisheries management in Lake Ijsselmeer (the Netherlands). J. Sea. Res. 60:21-31

Fournier, D. and C.P. Archibald. 1982. A general theory for analyzing catch at age data. Canadian Journal of Fisheries and Aquatic Sciences 39: 941-949.

Fournier, D.A., Skaug, H.J., J. Ancheta, J. Ianelli, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert. 2012. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods and Software, 27(2): 233–249. doi:10.1080/10556788.2011.597854.

Haltuch, M. A., Punt, A. E., and Dorn, M. W. 2008. Evaluating fishery management reference points in a variable environment. Fisheries Research, 94: 290-303.

Hilborn, R., & C. V. Minte-Vera. 2008. Fisheries-induced changes in growth rates in marine fisheries: Are they significant?. Bulletin of Marine Science, 83.1:95.

Hilborn R. and C.J. Walters. 1992. Quantitative fisheries stock assessment: Choice, Dynamics and Uncertainty. Chapman and Hall, New York.

Holland, D.S. 2010. Management strategy evaluation and management procedures: tools for rebuilding and sustaining fisheries, OECD Food, Agriculture and Fisheries Working Papers, No. 25, OECD, Paris, France.

Honey K. T., J. H. Moxley, and R. M. Fujita. 2010. From Rags to Fishes: Data-poor Methods for Fisheries Managers. Managing Data-Poor Fisheries: Case Studies, Models and Solutions 1:159-184.

ICES. 2012. Report of the Workshop on Frequency of Assessments (WKFREQ), 6-8 March 2012, By Correspondence. ICES CM 2012/ACOM 34. International Council for the Exploration of the Sea, Copenhagen, Denmark. Available from: http://www.nwwac.org/_fileupload/Image/REPORT_ICES_WKFREQ_2012.pdf [accessed August 2014]

Kawasaki, T. 1980. Fundamental relations among the selections of life history in marine teleosts. Bull. Jpn. Soc. Sci. Fish. 46: 289–293.

Li Y., J. R. Bence and T. O. Brenden., in review. The Influence of Stock Assessment Frequency on Achievement of Fishery 1 Management Objectives. Canadian Journal of Fisheries and Aquatic Science

Mace P.M., N. W. Bartoo, A. B. Hollowed, P. Kleiber, R. D. Methot, S. A. Murawski, J. E. Powers, and G. P. Scott. 2001. Marine Fisheries Stock Assessment Improvement Plan. NOAA Technical Memorandum, NMFS-F/SPO-56. National Marine Fisheries Service, Washington D.C. Available from: http://www.st.nmfs.noaa.gov/StockAssessment/ [accessed September 2014].

McGoodwin, J. R., G. H. Kruse, V. F. Gallucci, D. E. Hay, R. I. Perry, R. M. Peterman, T. C. Shirley, P. D. Spencer, B. Wilson, and D. Woodby. 2007. Fisheries Assessment and Management in Data-Limited Situations. Arctic, Antarctic, and Alpine Research. 39:185.

Mid-Atlantic Fisheries Management Council**.** 2011. Omnibus Amendment. Available from: www.nero.noaa.gov/nero/regs/frdoc/11/11OmnibusAmendmentEA&CommentsFinal.pdf [accessed August 2014].

National Oceanic and Atmospheric Administration, Fisheries division. 2012. Fish stock assessments 101. Available from: http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB8QFjAA&url=http%3A%2F%2Fwww.omao.noaa.gov%2Fpublications%2F1to2grade.pdf&ei=sf8fVfybEsLnsATt64DQBA&usg=AFQjCNGnjBann

uXF-7Po1IGfJ7WY0vKM0w&bvm=bv.89947451,d.cWc&cad=rja. [accessed January 2012]

National Marine Fisheries Service. 2006 Magnuson-Stevens Fishery Conservation and Management Reauthorization Act of 2006. Available from: http://www.nmfs.noaa.gov/msa2007/ [accessed December 2014].

National Research Council. Review of Northeast Fishery Stock Assessments. Washington, DC: The National Academies Press, 1998. Pg. 35-38

NOAA. 2014. Prioritizing Fish Stock Assessments. NOAA Fisheries Draft Protocol for Prioritizing Fish Stock Assessments. Available from: www.st.nmfs.noaa.gov/Assets/stock/documents/Prioritizing%20Fish%20Stock%20Assessment_Feb2014_final%20draft.pdf [accessed February 2015]

Ono, K., R. Licandeo, M.L. Muradian, C.J. Cunningham, S.C. Anderson, F. Hurtado-Ferro, K.F. Johnson, C.R. McGilliard. C.C. Monnahan, C.S. Szuwalski, J.L. Valero, K.A.Vert-Pre, A.R. Whitten and A.E.Punt. 2014. The importance of length and age composition data in statistical age-structured models for marine species. ICES J. mar. Sci., Advance access doi:10.1093/icesjms/fsu007

Patterson, K., and Résimont, M. 2007. Catch and stability in landings: the response of fisheries to scientific advice and TACs. – ICES Journal of Marine Science, 64: 714–717.

Pilling GM, Apostolaki P, Failler P, Floros C, Large PA, Morales-Nin B, Reglero P, Stergiou KI & Tsikliras AC. 2008. Assessment and management of data-poor fisheries. In: A Payne, J Cotter, T Potter (eds) Advances in Fisheries science: 50 years on from Beverton and Holt. Blackwell Publishing, CEFAS. pp. 280-305

Pope, J.G. 1988. Collecting fisheries assessment data. Fish Population dynamics, 2nd edition. Wiley, Chichester, pp. 63-82.

Prager, M. H., and K. W. Shertzer. 2010. Deriving acceptable biological catch from the overfishing limit: implications for assessment models. North American Journal of Fisheries Management 30:289-294.

Punt, A. E., and G. P. Donovan. 2007. Developing management procedures that are robust to uncertainty: lessons from the International Whaling Commission. ICES Journal of Marine Science 64:603–612.

Quinn, T. J., II, and R. B. Deriso. 1999. Quantitative fish dynamics. Oxford University Press. New York.

Restrepo, V. R., G. G. Thompson, P. M. Mace, W. L. Gabriel, L. L. Low, A. D. MacCall, R. D. Methot, J. E. Powers, B. L. Taylor, P. R. Wade, and J. F. Witzig. 1998. Technical guidance on the use of precautionary approaches to implementing National Standard 1 of the Magnuson–Stevens Fishery Conservation and Management Act. NOAA Tech. Memo. NMFS-F/SPO-31, 54 p.

Samuels M. L., J. A. Witmer, and A. A. Schaffner. 2012. Statistics for the Life Sciences: Fourth Edition. Prentice Hall. Boston, MA.

Shertzer, K.W., and M. H. Prager. 2007. Delay in fishery management: diminished yield, longer rebuilding, and increased probability of stock collapse. ICES J. Mar. Sci. 64: 149–159.

Shertzer, K., M. Prager, and E. Williams. 2008. A probability-based approach to setting annual catch levels. Fishery Bulletin 106:225-232.

Smith, D., A. Punt, N. Dowling, A. Smith, G. Tuck, and I. Knuckey. 2009. Reconciling Approaches to the Assessment and Management of Data-Poor Species and Fisheries with Australia's Harvest Strategy Policy. Marine and Coastal Fisheries: Dynamics, Management, and Ecosystem Science: 244-254.

Smith A. D. M., K. J. Sainsbury and R. A. Stevens. 1999. Implementing effective fisheries-management systems – management strategy evaluation and the Australian partnership approach. ICES J. Mar. Sci. 56: 967-979.

Terceiro, M. 2002. The summer flounder chronicles: science, politics, and litigation, 1975–2000. Rev Fish Biol Fish 11: 125-168.

Terceiro, M. 2011. The summer flounder chronicles II: new science, new controversy, 2001–2010. Rev Fish Biol Fish 21: 681-712

Thorson J. T., O. P. Jensen and E. F. Zipkin. 2014. How variable is recruitment for exploited marine fishes? A hierarchical model for testing life history theory. Canadian Journal of Fisheries and Aquatic Sciences. 71:973-983

Thorson J. T., and C.V. Minte-Vera. 2014. Relative magnitude of cohort, age, and year effects on size at age of exploited marine fisheries. Fisheries Research: article in press.