ISBN: 978-93-82263-67-8
CMFRI Special Publication No.153

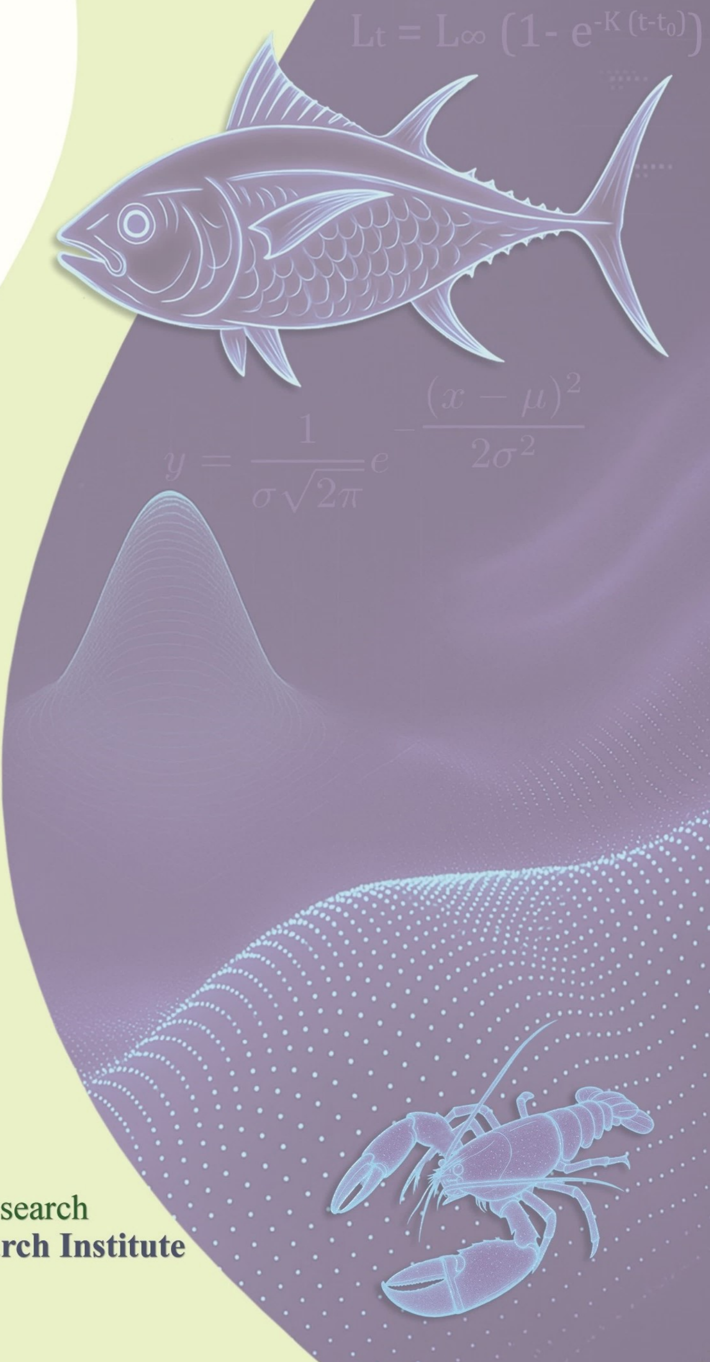
METHODOLOGICAL TOOLS FOR ASSESSING TROPICAL FISH STOCKS: A PRACTICAL GUIDE USING R (Vol.1)

$L_t = L_\infty (1- e^{-K(t-t_0)})$
$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Indian Council of Agricultural Research
Central Marine Fisheries Research Institute

ISBN: 978-93-82263-67-8
CMFRI Special Publication No.153

METHODOLOGICAL TOOLS FOR ASSESSING TROPICAL FISH STOCKS: A PRACTICAL GUIDE USING R (Vol.1)


$L_t = L_\infty \left(1 - e^{-K(t-t_0)}\right)$
$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Indian Council of Agricultural Research
Central Marine Fisheries Research Institute

CMFRI Special Publication No.153

METHODOLOGICAL TOOLS FOR ASSESSING TROPICAL FISH STOCKS: A PRACTICAL GUIDE USING R (Vol.1)

Gyanaranjan Dash, Rajan Kumar, Muktha Menon, Swatipriyanka Sen, Shikha Rahangdale, Santosh Bhendekar, A.P. Dineshbabu, Geetha Sasikumar, Sreenath K.R. and A. Gopalakrishnan

Published by:
Dr. A. Gopalakrishnan
Director
ICAR-Central Marine Fisheries Research Institute
P.B. 1603, Ernakulam North P.O.
Kochi- 682 018, Kerala, India
www.cmfri.org.in

E-mail: director.cmfri@icar.org.in
Telephone: +91 484 2394867
Fax: +91 484 2394909

Publication, Production & Co-ordination:
Arun Surendran P.S., Library and Documentation Centre,
ICAR-CMFRI, Kochi, India

Citation: Dash G., Kumar R., Menon M., Sen S., Rahangdale S., Bhendekar S., Dineshbabu A.P., Sasikumar G., Sreenath K.R. and Gopalakrishnan A., 2024. Methodological tools for assessing tropical fish stocks: a practical guide using R (Vol.1). CMFRI Special Publication No.153. ICAR-Central Marine Fisheries Research Institute, Kochi, India, pp.236.

ISBN 978-93-82263-67-8

Foreword

Tropical fisheries play a crucial role in sustaining national well-being and human livelihoods. Their vast biodiversity and ecological importance, combined with their socio-economic value, demand precise and effective management strategies. Stock assessment, a cornerstone of fisheries management, provides essential insights into the health, dynamics, and sustainability of fish populations. With the increasing global emphasis on sustainable fisheries, the need for comprehensive stock assessments has grown, especially in tropical regions, which have traditionally been constrained by limited data availability.

To address this need, the ICAR-Central Marine Fisheries Research Institute (ICAR-CMFRI) has developed this book-A practical and comprehensive guide for stock assessment of tropical fisheries using R. This resource underscores ICAR-CMFRI's commitment to advancing fisheries science and promoting sustainable practices in the region. It is designed for fisheries scientists, managers, and research scholars, equipping them with modern tools and techniques to conduct rigorous and reliable stock assessments, even in data-limited contexts. The book introduces users to R, a powerful statistical software widely used for data analysis, modeling, and visualization. Detailed instructions on the installation and configuration of R and RStudio provide the foundation for implementing advanced stock assessment methods. By leveraging the capabilities of R, the book enables users to gain deeper insights into fish population dynamics, thereby empowering informed decision-making and sustainable management strategies.

Focusing on tropical fisheries, the manual covers a range of methods tailored to their unique challenges. It delves into length-based methods, such as ELEFAN approaches, catch curve analysis, cohort analysis, stock simulation, and length-based Bayesian techniques. These methods are valuable in regions where traditional age-based assessments are not workable. Practical guidance and real-world examples show the application of these techniques, supported by clear and reproducible R code. Besides length-based approaches, the book explores catch-based surplus production methods, like effort standardization, CMSY++, and stock reduction analysis (sraplus). These approaches are especially useful in data-poor situations, where only catch and/or catch rate data are available. By providing robust tools and practical examples, the manual equips users to assess the impact of fishing and derive meaningful conclusions to support effective management decisions.

This manual aligns with ICAR-CMFRI's vision of sustainable marine resource management through innovative research, capacity building, and stakeholder engagement.

By fostering a deeper understanding of fish population dynamics and fishing effects, it aims to strengthen the development of strategies that balance ecosystem health, fisheries sustainability, and the livelihoods of coastal communities.

The preparation of this book reflects the dedication and expertise of a team of scientists and contributors committed to excellence in fisheries research. Their efforts have helped in producing this resource. Practical and user-friendly, it empowers fisheries professionals to navigate the complexities of tropical fisheries management, addressing challenges posed by limited data availability and the urgent need for sustainable practices.

As we face the emerging challenges of fisheries management in a changing world, this book serves as an essential tool for fisheries scientists and practitioners. It provides the knowledge and technical foundation needed to implement advanced stock assessment methods, contributing to the resilience and sustainability of tropical fisheries. By fostering a deeper understanding of fish populations and their dynamics, we can secure the resilience of these vital resources, safeguarding livelihoods and ecosystems for generations to come.

Dr. A. Gopalakrishnan
Director
ICAR-CMFRI

Overview

Tropical fisheries represent some of the most biologically diverse and economically significant marine resources globally. The abundance and variety of tropical fish species support millions of livelihoods, providing food, employment, and cultural value to many coastal communities. Effective management of these resources is essential for ensuring their sustainability and resilience in the face of increasing environmental and anthropogenic pressures. Stock assessment, the scientific evaluation of the status of fish populations, forms the backbone of sustainable fisheries management, offering critical insights into fish stock health and guiding management decisions.

Tropical fish stocks face unique challenges that distinguish them from their temperate counterparts. The high diversity of species, variability in life history traits, and often limited data availability pose significant hurdles to effective stock assessment. Besides this, tropical fisheries are highly dynamic and complex, characterized by the use of crafts and gear targeting multitude of species. This diversity makes stock assessment particularly challenging. Traditional stock assessment methods, developed primarily for temperate fisheries, may not fully capture the dynamics of tropical ecosystems. Therefore, it is crucial to adapt and develop methodologies that account for the specificities of tropical fisheries, enabling accurate assessments that inform sustainable management practices. R, a powerful statistical programming language, has emerged as a crucial tool in fisheries science, providing a versatile and comprehensive environment for data analysis, visualization, and modeling. R's extensive package ecosystem, particularly those tailored to fisheries science, such as TropFishR, offers a robust framework for implementing advanced stock assessment techniques.

This book is designed to guide fisheries scientists and practitioners through the process of using R for stock assessment of tropical fishes, equipping them with the skills and knowledge needed to conduct detailed and reliable assessments. It provides a step-by-step approach to stock assessment using R, covering installation and setup of R and RStudio, application of length-based methods, and catch-based methods. Each section builds on the previous, offering a comprehensive guide from basic setup to advanced analytical techniques.

Installing R and RStudio

The foundation of this manual is the installation and configuration of the necessary software tools. This section provides a detailed guide to installing R, RStudio and Rtools, ensuring that users have a functional environment for implementing the stock assessment methods described in subsequent sections. RStudio, an integrated development environment for R, enhances usability with its user-friendly interface, making it easier to write and manage code, visualize data, and debug analyzes. Rtools, a collection of resources for building R packages on Windows, is essential for compiling and installing packages that extend the capabilities of R.

Length-based methods

Length-based methods are crucial for assessing fish stocks, particularly in data-moderate contexts where detailed age data may be scarce. This section delves into various length-based techniques, providing practical examples and R code to implement each method.

Estimation of VBGF growth parameters using ELEFAN

The Electronic Length Frequency Analysis (ELEFAN) approach allows for the estimation of growth parameters from length frequency data. TropFishR includes both a traditional and updated versions of ELEFAN with new optimization techniques. The K-scan technique, part of the ELEFAN framework, involves scanning for different values of the growth parameters, i.e., L_∞, and K that best fit the observed length frequency data, providing a foundation for further analysis of population dynamics. The section also deals with the application of advanced versions of ELEFAN that use simulated annealing and genetic algorithm for deriving the growth parameters and their confidence intervals.

Estimation of hatching time/gestation period (t_o)

Estimating the hatching time or gestation period (t_o) is vital for understanding the breeding/ spawning time, spawning periodicity, and recruitment dynamics, which enable more precise assessments of stock status. The section provides codes for empirical and precise estimation of t_o and its confidence interval using generic R codes not available in TropFishR.

Estimation of maximum age/longevity (t_{max})

Determining the maximum age or longevity t_{max} of fish species is essential for understanding their life history strategies and population dynamics. This parameter helps in modeling the natural mortality (M) experience by fish populations and assessing their vulnerability to fishing pressures. The section provides codes for empirical and precise estimation of t_{max} and its confidence interval using generic R codes, besides the default estimation by the TropFishR.

Estimation of natural mortality (M)

Natural mortality (M) is a critical parameter influencing fish population dynamics. Accurate estimation of M provides insights into the survival rates of fish and their resilience to exploitation, forming a key component of stock assessment models. The section provides codes for empirical and precise estimation of M and its confidence interval using generic R codes, besides the default estimation method by the TropFishR.

Catch curve analysis

The catch curve analysis estimates the total mortality rate (Z) by analyzing the exponential decay in the age structured catch as a proxy of the real fish population. This section covers the implementation of catch curve analysis in R, including the preparation of age composition data, fitting the catch curve, and interpreting the results. The analysis also provides crucial information regarding the probability of capture for different length (age) groups, which is used to assess gear selectivity and length at capture (LC50). Understanding the total mortality rate is crucial for assessing the fishing mortality rate (F), exploitation rate (E) endured by the fish stocks, which helps in identifying potential overfishing. This section provides detailed guidance on performing length converted catch curve analysis in R, including data preparation, model fitting, and interpretation of results.

Virtual population analysis (VPA)/cohort analysis (CA)

The virtual population analysis (VPA)/cohort analysis (CA) involves tracking the dynamics of a cohort or group of fish born in the same period. This method helps in understanding the growth, survival, and exploitation rates of different length groups (cohorts), providing valuable insights into population dynamics. This section provides detailed guidance on performing cohort analysis in R, including data preparation, model fitting, and interpretation of results.

Stock simulation

Stock simulation models are essential for evaluating the potential outcomes of different management scenarios. The analysis simulates the effect of a change in fishing mortality and gear selectivity on the yield and biomass of the stock. This section introduces the Thompson and Bell's prediction model and Beverton and Holt's yield per recruit model, providing practical examples of how to implement these models in R.

Length-Weight relationship (LWR)

Establishing the relationship between length and weight (LWR) of the fish is essential to convert the easily available length-based information to biomass during biomass modeling and stock simulation. This section provides a detailed guide to modeling the LWR, testing the growth of the fish (isometric vs. allometric) and assessing the significant difference in the body weight of fish between sexes through implementing ANCOVA using the R interface. The section also addresses the advance procedures for increasing the accuracy of LWR modeling, such as information theory-based multi-model comparison criterion such as Akaike Information Criterion (AICc) to assess the best modeling approach for the LWR.

Length at maturity (LM50)

Estimating the length at maturity is crucial for understanding the reproductive biology of fish species. This section provides a detailed guide to calculating the length at maturity in R, including data preparation, model fitting, and interpretation of results. Accurate estimation of the length at maturity informs management decisions aimed at protecting spawning stock biomass (SSB), which is critically required for ensuring the sustainability of fish stocks.

Length-based spawning potential ratio (LBSPR)

The length-based spawning potential ratio (LBSPR) is a metric used to assess the reproductive potential of a fish stock. LBSPR is a valuable tool for assessing the sustainability of fish stocks and identifying potential risks to reproductive capacity. This section provides a comprehensive guide to calculating LBSPR in R, including data preparation, model fitting, and interpretation of results.

Length-based Bayesian biomass estimator (LBB)

The length-based Bayesian biomass estimator (LBB) is an advanced method for estimating fish biomass using length frequency data. This section introduces the LBB method, providing practical examples of how to implement it in R, including parameter estimation, model fitting, and interpretation of results.

Catch-based methods

The catch-based methods are essential for assessing fish stocks under data poor conditions when length or age data are limited. This section covers various catch-based techniques, providing practical examples and R code to implement each method.

Effort standardization

Tropical fisheries are characterized by its multi-gear and multi-species nature, leading to mixed species fisheries. Therefore, it is necessary to standardize the effort in terms of the major gear used for exploiting any species, especially when multiple gears are involved in its exploitation. Effort standardization is a method used to account for differences in fishing effort when analyzing catch data. This section provides a comprehensive guide to performing effort standardization in R.

Catch-based MSY (CMSY++)

The CMSY++ is a catch-based method for estimating maximum sustainable yield (MSY) and related reference points using time series of catch and effort data. This section provides detailed instructions on implementing catch-based MSY in R when only the catch data is available and the Bayesian Schaefer model (BSM) when both the catch and effort data are available.

Stock reduction analysis (sraplus)

The sraplus is an advanced version of stock reduction analysis (SRA) for assessing fish stocks using catch data, but also has the option to incorporate effort data. It uses SRA when only catch data is available but has the flexibility to fit the SRA estimates with observed CPUE when effort data is supplied. This section introduces the sraplus method, providing practical examples of how to implement it in R. SRA Plus is a valuable tool for assessing the effects of historical fishing on fish stocks.

This volume of the book aims to empower fisheries scientists and practitioners with the tools and knowledge to conduct effective stock assessments of tropical fish stocks using R. The authors are also committed to publishing Volume II of the book, which will cover advanced topics such as LIME, Stock Synthesis, SPiCT, JABBA, CatDyn, and Bayesian growth modelling. By providing detailed guidance, this book equips users with the skills needed to implement advanced stock assessment techniques and make informed management decisions. The combination of R's powerful analytical capabilities and the specific methodologies described in this book offers a robust framework for addressing the challenges of tropical fisheries management and ensuring the sustainability of these vital resources for future generations. Through the practical examples and comprehensive instructions provided in this book, users will gain a deep understanding of the principles and practices of stock assessment, enabling them to contribute to the sustainable management of tropical fish stocks. As the challenges in tropical fisheries continue to evolve, the ability to apply rigorous and innovative stock assessment methods will be essential for maintaining the health and resilience of these ecosystems.

(Authors)

CONTENTS

Sl. No.	Subject	Page No.
1.	**Installing R, RStudio and Rtools**	**01-03**
1.1.	Installing R for the first time	01
1.2.	Installing RStudio for the first time	02
1.3.	Installing Rtools for the first time	03
1.4.	Updating R	04-05
2.	**Length-based Methods**	**06-170**
2.1.	**Estimation of VBGF growth parameters using ELEFAN**	**06-23**
2.1.1.	Requirements for ELEFAN • *R package (TropFishR)* • *Length-based data*	07
2.1.2.	Creating a length frequency file (LFQ) on R using Data format 1 (Raw data)	07
2.1.3.	Controlling the length class interval (Bin size)	08
2.1.4.	Raising the length frequency data	09
2.1.5.	Creating a length frequency file (LFQ) on R using Data Format 2 (Raised data)	10
2.1.6.	Controlling length class interval (Bin size)	11
2.1.7.	ELEFAN: Response surface analysis (RSA) • *Preparing L∞ and K ranges for RSA* • *RSA Cross' method (default method of FISAT II)* • *RSA optimize method (new method of TropFishR)* • *Understanding the RSA output*	11-14
2.1.8.	ELEFAN: K-scan • *Preparing L∞ and K ranges for K-scan* • *K-scan cross' method (Default method of FiSAT II)* • *K-scan optimize method (New method of TropFishR)* • *Understanding the K-scan output*	14-16
2.1.9.	ELEFAN_SA: ELEFAN simulated annealing • *Preparing L∞ and K ranges for ELEFAN_GA* • *Understanding the ELEFAN_SA diagnostic plot*	16-18
2.1.10.	ELEFAN_GA: ELEFAN Genetic Algorithm • *Preparing L∞ and K ranges for ELEFAN_SA*	18-21

	- *Understanding the ELEFAN_GA diagnostic plot*	
2.1.11.	ELEFAN comparison plot	21
2.1.12.	Estimation of mean and confidence intervals of VBGF growth parameters - *Through Simulated Annealing (Using ELEFAN_SA)* - *Through Genetic Algorithm (Using ELEFAN_GA)*	22
2.2.	**Estimation of gestation period or hatching time (t_o)**	**23-25**
2.2.1.	Requirements for t_o calculation	24
2.2.2.	Empirical equation for t_o	24
2.2.3.	Rearranged VBGF equation for t_o	24
2.2.4.	Estimation of mean and confidence intervals of t_o	25
2.3.	**Estimation of longevity/ maximum age (t_{max})**	**25-27**
2.3.1.	Requirements for t_{max} calculation	26
2.3.2.	Empirical equation for t_{max}	26
2.3.3.	Rearranged VBGF equation for t_{max}	27
2.3.4.	Natural mortality rate (M)-based equation for t_{max}	27
2.3.5.	Estimation of mean and confidence intervals of t_{max}	27
2.4.	**Compilation, plotting and saving of growth parameters**	**28-30**
2.4.1.	Compilation and exporting the mean and confidence intervals of growth parameters as CSV	28
2.4.2.	Plotting confidence intervals of the growth parameters	29
2.4.3.	Saving essential growth parameters to the LFQ file	29
2.5.	**Estimation of natural mortality rate (M)**	**30-38**
2.5.1.	Requirements for the estimation of M	32
2.5.2.	Estimation of mean and confidence intervals of M - *Pauly's Method for M* - *Then_growth Method for M* - *Alagaraja's Method for M* - *Then_t_{max} Method for M* - *Hoenig's Method for M*	33-35
2.6.	**Compilation, plotting and saving of natural mortality rates (M) derived from different methods**	**36-37**
2.6.1.	Compilation and exporting the mean and confidence intervals of M as CSV	36
2.6.2.	Plotting confidence intervals of M	36

2.6.3.	Saving the selected M to the LFQ file	37
2.7.	**Estimation of exploitation parameters using the length converted catch curve analysis**	**38-52**
2.7.1.	Requirements for the length converted catch curve (CC) analysis	42
2.7.2.	Catch curve analysis for a combination of years using multiyear mean catch data • *Preparing a catch vector* • *Preparing a mean catch vector from the catch vector*	42
2.7.3.	Catch curve analysis for a specific year • *Preparing a catch vector*	43
2.7.4.	Default graphical outputs from the catch curve analysis	45
2.7.5.	Enhanced visualization of the catch curve analysis	45
2.7.6.	Enhanced visualization of the probability of capture • *Catch curve model prediction* • *Deriving mean and confidence intervals of age at capture* • *Deriving mean and confidence intervals of length at capture* • *Plotting age vs. probability of capture* • *Plotting length vs. probability of capture*	46
2.7.7.	Summarizing derived exploitation parameters from the catch curve analysis	50
2.7.8.	Plotting the exploitation parameters derived from the catch curve analysis	51
2.8.	**Estimation of length-class wise fishing mortality rate, yield and biomass using virtual population analysis (VPA)/cohort analysis (CA)**	**53-65**
2.8.1.	Requirement for cohort analysis (CA)	56
2.8.2.	Cohort analysis for a combination of years using multiyear mean catch data	57
2.8.3.	Cohort analysis for a specific year	58
2.8.4.	Summarizing the population estimates from the length-based cohort analysis	58
2.8.5.	Default graphical outputs from the cohort analysis results • Population number plot • Population biomass plot	60
2.8.6.	Enhanced visualization of the cohort analysis results • Population number plot (2D) • Population biomass plot (2D)	61

	• Population number plot (3D) • Population biomass plot (3D)	
2.9.	**Stock simulation by Thompson and Bell's model (TB)**	**66-87**
2.9.1.	Requirement for Thompson and Bell's model (TB)	67
2.9.2.	Simulating the effect of changes in FM on the stock status (TB1)	68
2.9.3.	Biological reference points from TB1	71
2.9.4.	Current stock status estimates from TB1	72
2.9.5.	Default graphical output from TB1	73
2.9.6.	Simulating the combined effect of changes in FM and LC$_{50}$ on stock status (TB2)	73
2.9.7.	Biological reference points from TB2	77
2.9.8.	Current stock status estimates from TB2	77
2.9.9.	Default graphical output from TB2	78
2.9.10.	Enhanced visualization of Thompson and Bell prediction model • *Plotting the effect of a change in FM on stock status (TB1 graph)* • *Including the required SPR threshold as precautionary management reference point in TB1 graph* • *Plotting the effect of a change in FM and LC on stock status (TB2 graph)* • *Enhanced Yield Isopleth visualization from TB2 (2D plot)* • *Enhanced Yield Isopleth visualization from TB2 (3D plot)* • *Enhanced Biomass Isopleth visualization from TB2 (2D plot)* • *Enhanced Biomass Isopleth visualization from TB2 (3D plot)*	79
2.10.	**Stock simulation by Beverton and Holt's yield per recuit model (BHYPR)**	**88-110**
2.10.1.	Requirement for Beverton and Holt's yield per recruit model (BHYPR)	92
2.10.2.	Simulating the effect of changes in FM on the stock status (BHYPR1)	93
2.10.3.	Biological reference points from BHYPR1	95
2.10.4.	Current stock status estimates from BHYPR1	95
2.10.5.	Default graphical output from BHYPR1	96
2.10.6.	Simulating the combined effect of changes in FM and LC50 on stock status (BHYPR2)	96
2.10.7.	Biological reference points from BHYPR2	98
2.10.8.	Current stock status estimates from BHYPR2	99
2.10.9.	Default graphical output from BHYPR2	99
2.10.10	Enhanced visualization of Beverton and Holt's yield per recuit model	100

	(BHYPR) • *Plotting the effect of a change in FM on stock status (BHYPR1 graph)* • *Controlling the BHYPR1 plotting parameters for primary and secondary Y-axis* • *Plotting the effect of a change in FM and LC on stock status (BHYPR2 graph)* • *Controlling the BHYPR2 plotting parameters for primary and secondary Y-axis* • *Plotting and controlling yield and biomass isopleths from BHYPR2* • *Enhanced yield isopleth visualization from BHYPR2 (2D plot)* • *Enhanced yield isopleth visualization from BHYPR2 (3D plot)* • *Enhanced biomass isopleth visualization from BHYPR2 (2D plot)* • *Enhanced biomass isopleth visualization from BHYPR2 (3D plot)*	
2.11	**Understanding the stock simulation outputs (fisheries management reference points)**	**111-112**
2.12.	**Length-Weight Relationship (LWR)**	**115-132**
2.12.1.	Requirement for Length-Weight Relationship (LWR) • *Installing and loading dependent R-packages* • *Length-Weight data*	116
2.12.2.	Importing length-weight data to R Interface	117
2.12.3.	Modeling the relationship between length and weight (Linear Model) • *Linear modeling* • *Understanding the output of linear modeling*	117
2.12.4.	Modeling the relationship between length and weight (Nonlinear Model) • *Nonlinear modeling* • *Understanding the output of nonlinear modeling*	118
2.12.5.	Selecting the best modeling approach • *Criteria for model selection* • *Calculate AICc and their difference (delta AICc)* • *AICc comparison*	119
2.12.6.	Model Diagnosis • *Residual diagnostic plots for linear model* • *Residual diagnostic plot for nonlinear model* • *Understanding the output of residual diagnostic plot* • *Comparative residual density plot for normality of residuals* • *Shapiro-Wilk test of normality of residuals*	123

- *Understanding the output of the Shapiro-Wilk test*

2.12.7. Plotting the length-weight relationship 125

- *Model prediction and error propagation for the linear model*
- *Plotting linear model output*
- *Model prediction and error propagation for nonlinear model*
- *Plotting nonlinear model output*

2.12.8. Test for isometric growth 128

- *t-test for isometric growth*
- *Understanding the output of t-test*
- *Wald test for isometric growth*
- *Understanding the output of Wald test*

2.12.9. Test for difference in body weights between the sexes 129

- *Analysis of Covariance (ANCOVA)*
- *Building the model for testing the difference in slopes (b) of the regressions*
- *Understanding the output of the slope test*
- *Building the model for testing the difference in intercepts (a) of the regressions*
- *Understanding the output of the intercept test*

2.13. Length at Maturity 132-138

2.13.1. Requirement for Length at Maturity

- *Installing and loading dependent R-packages*
- *Maturity-at-length data*

2.13.2. Importing length-at-maturity data to R Interface 135

2.13.3. Modeling the relationship between length and maturity stage 135

Generalized linear modeling (GLM)

- *Model prediction and error propagation*
- *Understanding the output of generalized linear modeling*

2.13.4. Plotting the length at maturity 136

- *Model prediction and error propagation for logistic regression*
- *Deriving means and confidence intervals for length at maturity*
- *Deriving means and confidence intervals of age at maturity*
- *Plotting the length at maturity*

2.14. Length-Based Spawning Potential Ratio (LBSPR) 139-153

2.14.1. Requirements for LBSPR 143

2.14.2. Running the LBSPR analysis 146

2.14.3. Plotting LBSPR analysis outputs 147

2.14.4. Plotting observed LFQ vs. target LFQ 150

2.14.5. Implementing R Shiny package for LBSPR analysis 152

2.15. Length-Based Bayesian Biomass Estimator (LBB) 153-170

2.15.1. Requirements for LBB 157

- *Installing dependent R-packages*
- *Install the correct JAGS (Just Another Gibbs Sampler)*
- *Downloading the LBB files*
- *Supplying essential length frequency data and input parameters*

2.15.2. Running the LBB analysis 162

- *Open the R-script file in RStudio and set the working directory*
- *Define the stock to be analyzed in the opened R-script*
- *Define the ID file to be used in the opened R-script*
- *Run LBB codes*
- *LBB text output*
- *LBB Graphical output*

2.15.3. Plotting crucial outputs of LBB analysis (B/B_0 and F/M**)** 168

3. Catch-based Methods 171-236

3.1. Effort Standardization 171-181

3.1.1. Effort standardization using FESta R package 173

3.1.2. Requirements for FESta 175

- *Installing and loading FESta R package*
- *Importing the catch and effort data*
- *Formatting of imported catch and effort data*

3.1.3. Running FESta R package 176

- *FESta tabular output*
- *FESta graphical output*

3.1.4. Effort standardization using tweaked implementation of FESta 177

3.1.5. Requirements for tweaked implementation of FESta 178

- *Tweaked R codes following the concept of FESta*
- *Importing the catch and effort data*
- *Formatting of imported catch and effort data*

3.1.6. Running tweaked FESta R codes 178

3.1.7. Exporting the standardized catch, effort and CPUH as CSV 179

3.1.8. Plotting the effort and CPUE standardization results 180

3.2. Surplus production model (SPM) 181-209

3.2.1. Equilibrium SPM 184

3.2.2.	Non-equilibrium SPM	186
3.2.3.	Incorporating uncertainty (stochasticity) in the non-equilibrium SPM	187
3.2.4.	ASPIC (A Stock-Production model Incorporating Covariates)	188
3.2.5.	SPiCT (Surplus Production model in Continuous Time)	189
3.2.6.	JABBA (Just another Bayesian Biomass Assessment)	189
3.2.7	CMSY++	190
3.2.8	Requirements for CMSY++ • *Installing dependent R-packages* • *Installing the JAGS (Just Another Gibbs Sampler)* • *Installing 'snpar' R package* • *Downloading the CMSY++ files* • *Supplying essential catch (and effort) data and input parameters* • *Preparing own input data file* • *Preparing own input prior parameter file* • *Controlling the analysis*	191
3.2.9.	Running the CMSY++ analysis • *Open the R-script file in RStudio and set the working directory* • *Define the stock to be analyzed in the opened R-script* • *Define the output settings for the analysis* • *Run CMSY++ codes*	197
3.2.10.	CMSY++ text output	201
3.2.11.	CMSY++ graphical outputs	202
3.3.	**Stock Reduction Analysis (sraplus)**	**209-236**
3.3.1.	Depletion corrected average catch (DCAC)	210
3.3.2.	Depletion based stock reduction analysis (DB-SRA)	212
3.3.3.	Stock Reduction Analysis Plus (sraplus)	214
3.3.4.	Requirements for sraplus • *Updating R* • *Updating Rtools* • *Installing sraplus* • *Updating dependent R packages* • *Troubleshooting sraplus installation failure* • *Installing TAF* • *Installing SOFIA* • *Installing Git* • *Cloning a demo data from the Git repository*	216

- *Preparing own input catch (and effort) data file*
- *Preparing own input prior parameter file*

3.3.5. Running the sraplus analysis 224

- *Erase traces of previous analysis*
- *Preparation of data*
- *sraplus modeling*
- *Model engine customization*

3.3.6. sraplus tabular outputs 227

3.3.7. sraplus graphical outputs 229

Example data file download link **236**

https://drive.google.com/drive/folders/1cNNiT5t3vr1MRpK69vygENsQdqwn-OJD?usp=sharing

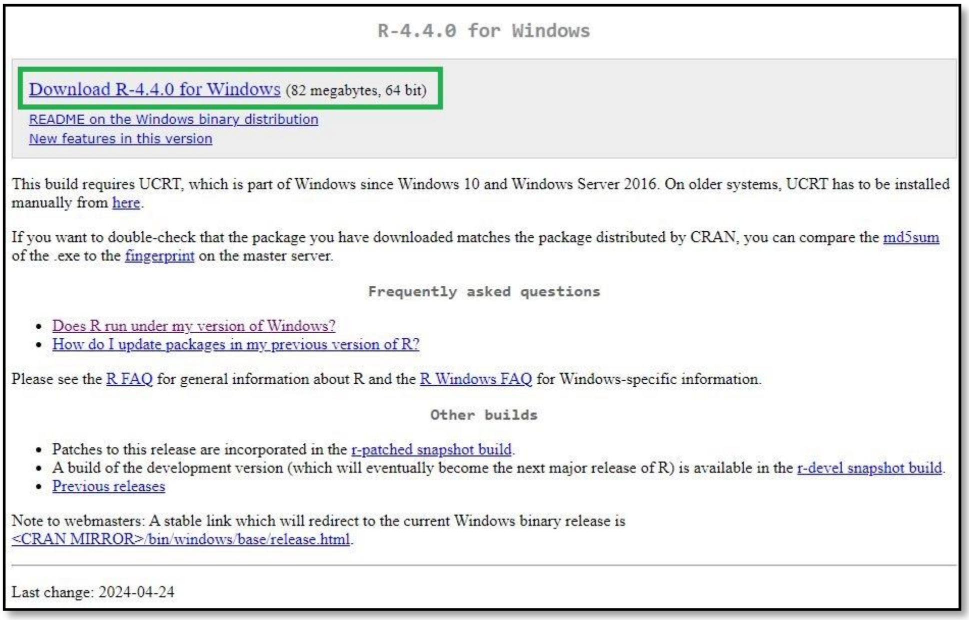
1. Installing R, RStudio and Rtools

1.1. Installing R for the first time

The R is a free open-source programming language widely used for organizing, analyzing, and visualizing data. It is highly versatile and user-friendly, offering a vast array of user-created packages and easy-to-follow instructions and guides for implementation (https://cran.r-project.org/web/packages/available_packages_by_name.html). R also includes an environment or console that can execute code without RStudio. However, the basic R console is quite limited and does not offer as many accessible tools as RStudio. Therefore, many users prefer RStudio for data analysis and visualization.

To install the R for different operating systems (Windows OS/ Mac OS/ Linux OS), visit the following site: https://www.rstudio.com/products/rstudio/download/#download or https://cran.r-project.org/ or https://cloud.r-project.org/, select the correct R setup file depending on the operating system (OS) and then click the **'Download R'**.

For example, to install the latest version of R for the commonly used Windows OS, visit the following site: https://cloud.r-project.org/bin/windows/base/ and click the **Download R** for Windows OS.



R-4.4.0 for Windows

Download R-4.4.0 for Windows (82 megabytes, 64 bit)
README on the Windows binary distribution
New features in this version

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

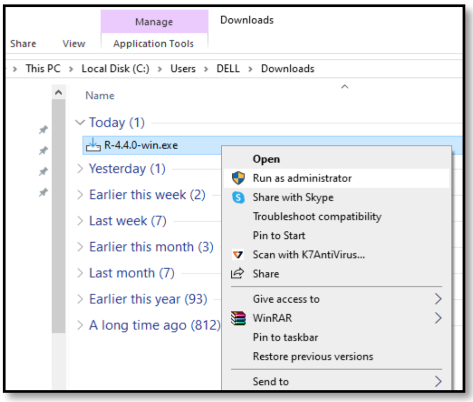
Other builds

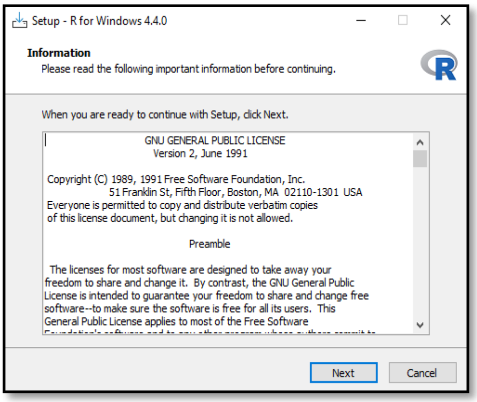
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2024-04-24

Clicking the **Download R** link will download the executable file for the latest version of R (ex: R-4.4.0-win.exe) usually to the download folder of the computer. Once the executable file for the latest version of R (e.g., R-4.4.0-win.exe) has finished downloading, proceed with the installation. It's recommended to install R as an administrator by right-clicking on the executable file and selecting **'Run as administrator'**. During the installation, simply click **'Next'** in each dialog box that appears to complete the process.





Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

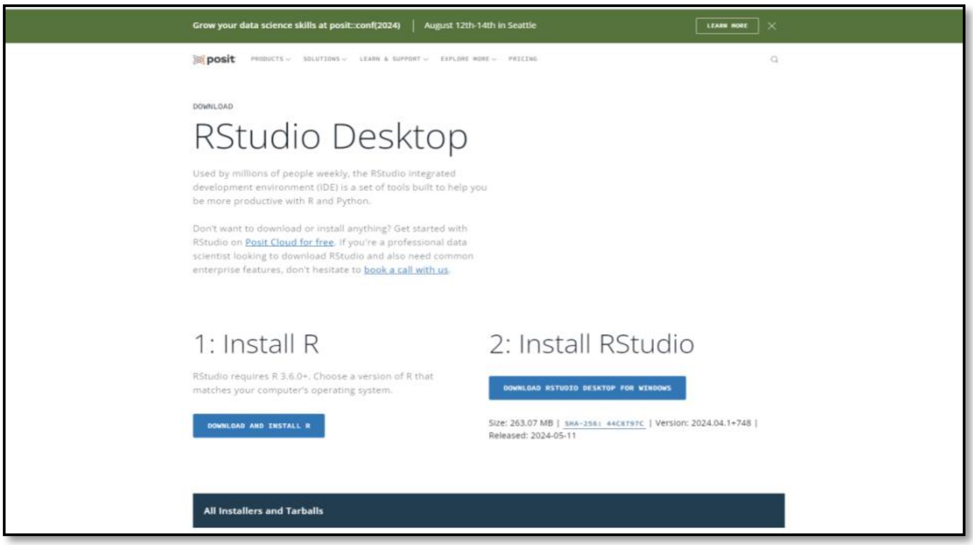
Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software

1.2. Installing RStudio for the first time

RStudio is an essential tool for working with the installed version of R. Although the basic R console can be used for data analysis and visualization, RStudio offers a clean, point-and-click interface with a variety of accessible tools. It allows for easy coding, organized management of data and files, convenient viewing of figures, and quick access to the help window. Therefore, many users prefer RStudio for data analysis and visualization.

To install the RStudio for different operating systems (Windows OS/ Mac OS/ Linux OS), visit the following site: https://www.rstudio.com/products/rstudio/download/#download, scroll down and click the 'Download RStudio' under 'All Installers and Tarballs' for the required OS. For example, to install the latest version of RStudio for the commonly used Windows OS, visit the following site: https://www.rstudio.com/products/rstudio/download/#download and click the 'Download RStudio Desktop for Windows'.


Grow your data science skills at posit::conf(2024) | August 12th-14th in Seattle
LEARN MORE
posit
PRODUCTS
SOLUTIONS
LEARN & SUPPORT
EXPLORE MORE
PRICING
DOWNLOAD

RStudio Desktop

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.

Don't want to download or install anything? Get started with RStudio on Posit Cloud for free. If you're a professional data scientist looking to download RStudio and also need common enterprise features, don't hesitate to book a call with us.

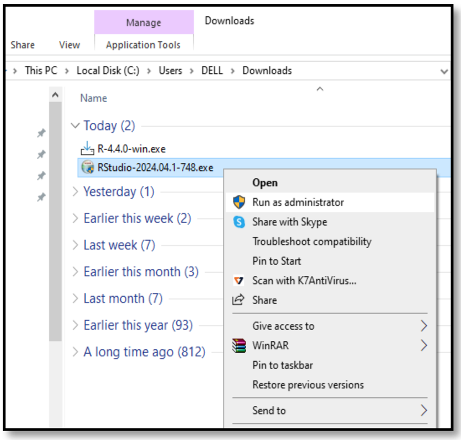
1: Install R

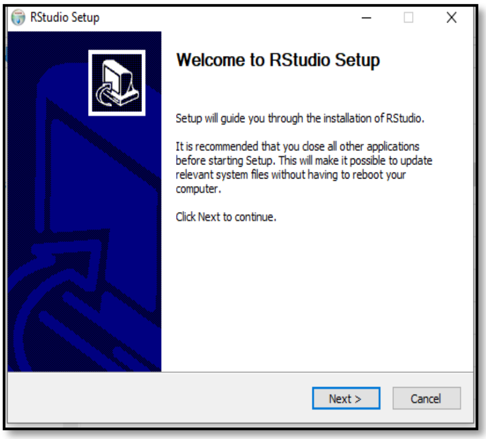
RStudio requires R 3.6.0+. Choose a version of R that matches your computer's operating system.

2: Install RStudio

Size: 263.07 MB | SHA-256: 44C8797C | Version: 2024.04.1+748 |
Released: 2024-05-11

Clicking ‘Download’ will download and save the executable (.exe) file for the latest version of RStudio (e.g., RStudio-2024.04.1-748.exe) to the download folder of the computer. Once the download is complete, proceed with the installation. It's recommended to install RStudio as an administrator by right-clicking on the executable file and selecting ‘Run as administrator’. To complete the installation, simply click ‘Next’ in each dialog box that appears.





Welcome to RStudio Setup

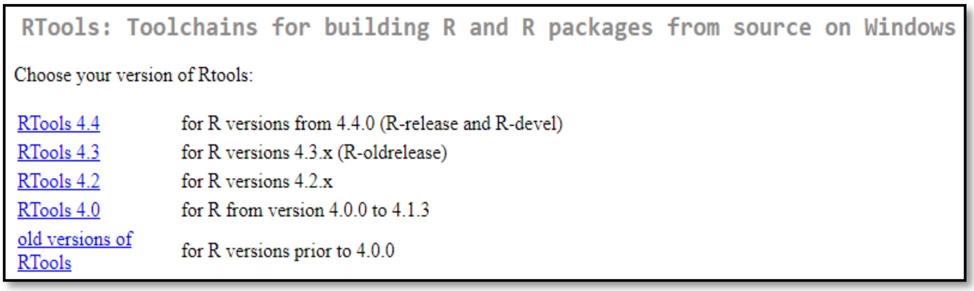
Setup will guide you through the installation of RStudio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

1.3. Installing Rtools for the first time

Rtools are required only when installing R packages from the source, particularly those that require the compilation of C/C++ or Fortran code, or when building R from the source. Ensure to install the appropriate version of Rtools based on the version of R installed on the system. The Windows installation packages for different versions of Rtools can be found at the following link: https://cran.r-project.org/bin/windows/Rtools/



RTools: Toolchains for building R and R packages from source on Windows

Choose your version of Rtools:

RTools 4.4	for R versions from 4.4.0 (R-release and R-devel)
RTools 4.3	for R versions 4.3.x (R-oldrelease)
RTools 4.2	for R versions 4.2.x
RTools 4.0	for R from version 4.0.0 to 4.1.3
old versions of RTools	for R versions prior to 4.0.0

Note: *Use the following code to manually put the Rtools on the PATH (for* ***RTools 4.0****).*

```
write('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"', file = "~/.Renviron", append = TRUE)
```

This will create a .Renviron text file in the Documents folder, which if opened using the Notepad, will show the following line:

```
PATH="${RTOOLS40_HOME}\usr\bin;${PATH}"
```

1.4. Updating R

Do not frequently update the R if it is not absolutely necessary. Updating R necessitates updating the dependent R packages, which is a time taking process.

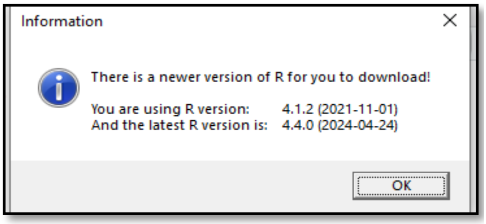
Check the version of **R** installed in the system, using the following code

```
R.version.string
```

Update R automatically

R can be easily updated with **'installr'** package on Windows OS. To update R on MacOS, the user needs to use **'updateR'** package.

```
install.packages ("installr")
library(installr)
updateR()
```

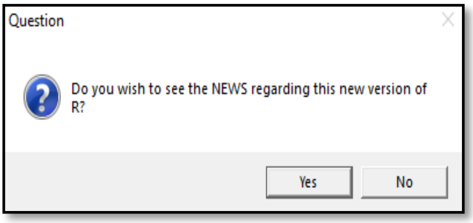


Information

There is a newer version of R for you to download!

You are using R version: 4.1.2 (2021-11-01)
And the latest R version is: 4.4.0 (2024-04-24)

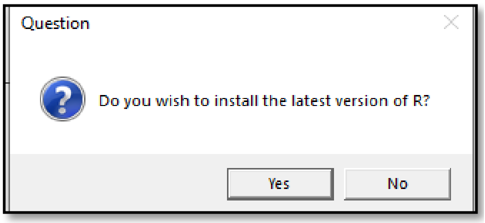
Click OK



Question

Do you wish to see the NEWS regarding this new version of R?

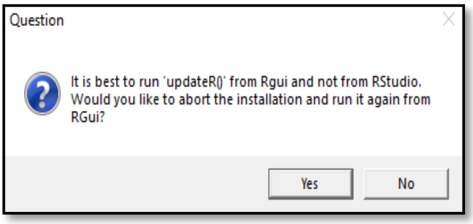
Click No



Question

Do you wish to install the latest version of R?

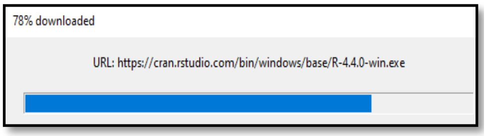
Click Yes



Question

It is best to run 'updateR()' from Rgui and not from RStudio. Would you like to abort the installation and run it again from RGui?

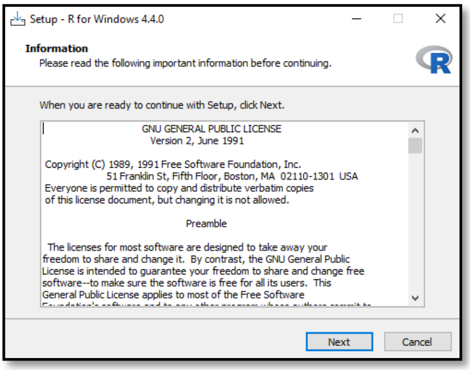
Click No



78% downloaded

URL: https://cran.rstudio.com/bin/windows/base/R-4.4.0-win.exe

It will download and prompt for the installation of a new version of R



Setup - R for Windows 4.4.0

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

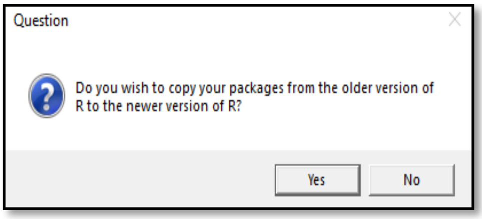
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

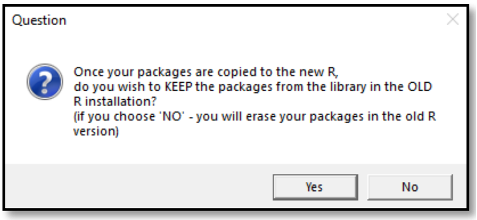
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software

Give permission (Yes/OK/Next) during the installation of a new version of R



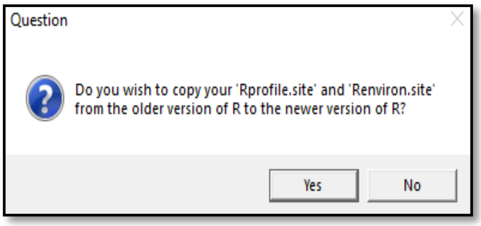
Do you wish to copy your packages from the older version of R to the newer version of R?

Click Yes



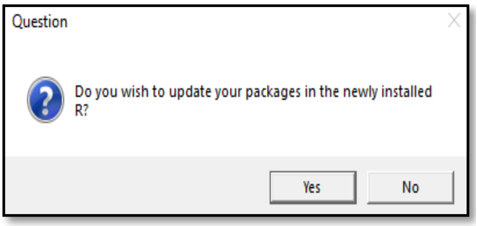
Once your packages are copied to the new R,
do you wish to KEEP the packages from the library in the OLD R installation?
(if you choose 'NO' - you will erase your packages in the old R version)

Click Yes



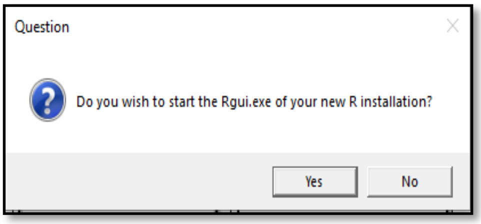
Do you wish to copy your 'Rprofile.site' and 'Renviron.site' from the older version of R to the newer version of R?

Click Yes



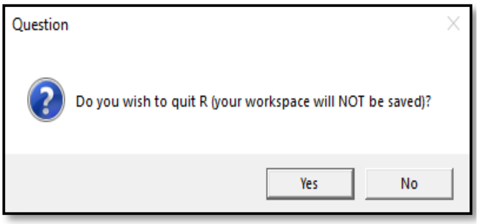
Do you wish to update your packages in the newly installed R?

Click Yes



Do you wish to start the Rgui.exe of your new R installation?

Click Yes



Do you wish to quit R (your workspace will NOT be saved)?

Click Yes

The installation process will check for a newer version of R. If an update is available, the process will download the latest R version and run the installer. After installation, the process will prompt the user to copy (or move) all packages from the old R library to the new one. It will then offer to update the transferred packages, open the new R GUI, and finally, close the old R.

References

Online resources

https://jennhuck.github.io/workshops/install_update_R.html#Installing_R

https://www.r-bloggers.com/2022/01/how-to-install-and-update-r-and-rstudio/

https://www.wikihow.com/Update-R

2. Length-based Methods

2.1. Estimation of VBGF growth parameters using ELEFAN

Introduction

The Electronic Length Frequency Analysis (ELEFAN) is a computer program developed by ICLARM that employs robust methods for analyzing length-frequency data of finfish and shellfish. It comprises five programs (ELEFAN 0, 1, 2, 3, and 4), with ELEFAN 1 specifically used for estimating von Bertalanffy Growth Function (VBGF) parameters (Pauly, 1987). ELEFAN is particularly suited for analyzing length-frequency data (LFQ) collected from fish stocks, especially in tropical regions where age-based data may be unavailable. The fundamental concept behind ELEFAN is that fish populations are made up of cohorts, which are groups of fish born around the same time. Due to their high numbers, cohorts form peaks (distinct modal size classes) in length-frequency distributions. When these cohorts grow over time, their growth is reflected in length-frequency distributions. The process begins by collecting fish length data at regular intervals (often monthly) and converting it into length-frequency distributions, which involves grouping the fish into different size classes. These periodic length-frequency distributions are then fitted with a series of growth curves generated using various combinations of L_∞ (asymptotic length) and K (growth coefficient). The goal is to align as many modal (peak) lengths observed at different times as possible to produce the best-fitting growth curve. ELEFAN uses a scoring algorithm to calculate an Rn-score, which measures how well the growth curves fit the peaks in the length-frequency data. The Rn-score is determined by comparing the ESP (Explained Sum of Peaks—the total number of 'peaks' and 'troughs' hit by a growth curve) with the ASP (Available Sum of Peaks—the sum of the maximum available 'peaks' accumulated by that curve). A higher Rn-score, calculated as $10^{(ESP/ASP)}/10$, indicates a better fit. The algorithm tests different growth parameter combinations to find the growth curve that maximizes the Rn-score. ELEFAN 1 applies a seasonally oscillating version of the von Bertalanffy Growth Function (Somer, 1988) to estimate growth parameters such as L_∞ and K.

Generalized VBGF

$$L_t = L_\infty \times \left(1 - exp^{-K(t-t_0)}\right)$$

Seasonally oscillating version of VBGF

$$L_t = L_\infty \times \left(1 - exp^{-K(t-t_0) + \left(\frac{CK}{2\pi} \times \sin(2\pi(t-t_s))\right) + \left(\frac{CK}{2\pi} \times \sin(2\pi(t_0-t_s))\right)}\right)$$

Where L_t is the length at age t, L_∞ is the asymptotic length, K is the growth coefficient, and t_0 is the theoretical age at which the fish's length would be zero. The parameter t_s (Summer Point = WP - 0.5) represents the start of the sinusoidal growth oscillation relative to t=0. The term "Winter Point" (WP = 0.5 + t_s) is often used instead of t_s, and it indicates the time of year (expressed as a fraction of the year) when growth is at its slowest (Pauly, 1987; Pauly et al., 1992). 'C' represents the amplitude of growth oscillation. When C=0, the seasonally oscillating version of the von Bertalanffy Growth Function (VBGF) reduces to the generalized VBGF without seasonal variation. When 0<C<1, growth oscillates seasonally but never halts entirely. For instance, when C=0.5, growth increases by 50% in summer and decreases by 50% in winter, but is never completely stopped. When

C=1, growth doubles in the summer (t_s=WP−0.5) and ceases entirely in winter (WP=0.5+t_s). If C>1, growth oscillates so strongly that it predicts a reduction in length during winter, which is highly unlikely in nature. In most cases, C>1 does not imply that fish shrink in winter, but rather suggests a prolonged period of no growth, which may occur in cold habitats (Pauly, 1987; Pauly et al., 1992). The value of C generally correlates with the difference between mean monthly summer and winter temperatures (ΔT). A ΔT of 10°C can cause a C value near 1. It has been observed that even a ΔT of 2°C is sufficient to induce statistically significant seasonal growth oscillations in tropical fish (Longhurst and Pauly, 1987). Unlike temperate fish, most tropical fish spawn and recruit continuously throughout the year. While this can make it difficult to identify and fit a growth curve to a distinct cohort, it is still workable. This is largely because, even though a portion of the population spawns year-round, the major spawning and recruitment events occur seasonally, which creates the peaks and troughs observed in length-frequency data (Pauly, 1987).

ELEFAN: R Implementation

The ELEFAN can be implemented on R interface using 'TropFishR' R package. The 'TropFishR' R package has been developed by Mildenberger et al. (2017). It provides a set of tools and functions for assessing the life history characteristics, stock dynamics, exploitation characteristics and health status of tropical and sub-tropical fish. The TropFishR package follows the FAO manual for tropical fish stock assessment (Sparre and Venema, 1998) and enhances data-limited fishery analysis by incorporating traditional and updated versions of the ELEFAN method, new optimization techniques, Millar's nonlinear selectivity models, and comprehensive methods for analyzing length-frequency data (LFQ). It supports stock assessment routines and management reference level derivation using yield per recruit modeling. Unlike the FiSAT II, the TropFishR offers greater data handling flexibility and automated analysis capabilities.

2.1.1. Requirements for ELEFAN

R package (TropFishR)

Install the 'TropFishR' R package using the following code (Do not install again if already installed): `install.packages("TropFishR", repos = "https://cran.rstudio.com/")`

Load the 'TropFishR' package using the following code: `library(TropFishR)`

Length-based data

Length-based data is also required for the analysis which is typically available in the following two formats, viz. (1) Data format 1 (Raw data), and (2) Data format 2 (Raised data) which need to be imported into R for analysis. Refer ***'Example data file download link'*** in the last page to download and used the example data.

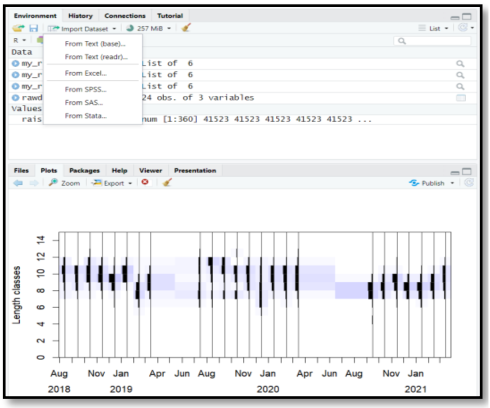
2.1.2. Creating a length frequency file (LFQ) on R using Data format 1 (Raw data)

Raw data (dates and lengths) that have not been aggregated needs to be formatted to create length frequencies (LFQ) for different months and subsequently raised to reflect the proportion of monthly landings.

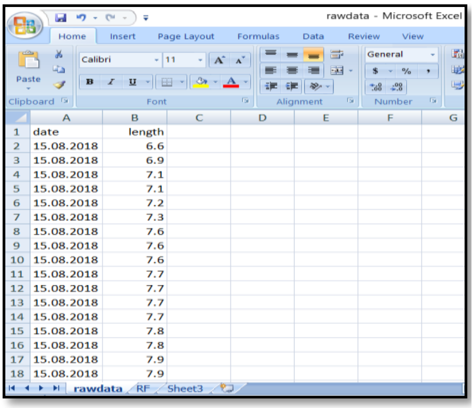
Click on the 'Import Dataset' option in the ‘Environment’ tab (top right panel). Then select 'From Excel'. Browse for the Excel file (e.g., the 'rawdata' sheet in the 'rawdata' Excel file) and import it.

Environment> Import Dataset> From Excel and then browse the file on disk and import.

Excel import window



Unraised dataset (Raw data)



	A	B
1	date	length
2	15.08.2018	6.6
3	15.08.2018	6.9
4	15.08.2018	7.1
5	15.08.2018	7.1
6	15.08.2018	7.2
7	15.08.2018	7.3
8	15.08.2018	7.6
9	15.08.2018	7.6
10	15.08.2018	7.6
11	15.08.2018	7.7
12	15.08.2018	7.7
13	15.08.2018	7.7
14	15.08.2018	7.7
15	15.08.2018	7.8
16	15.08.2018	7.8
17	15.08.2018	7.9
18	15.08.2018	7.9

> ***Note:*** *Use length measurements in centimeters (cm) to ensure uniformity in subsequent calculations.*

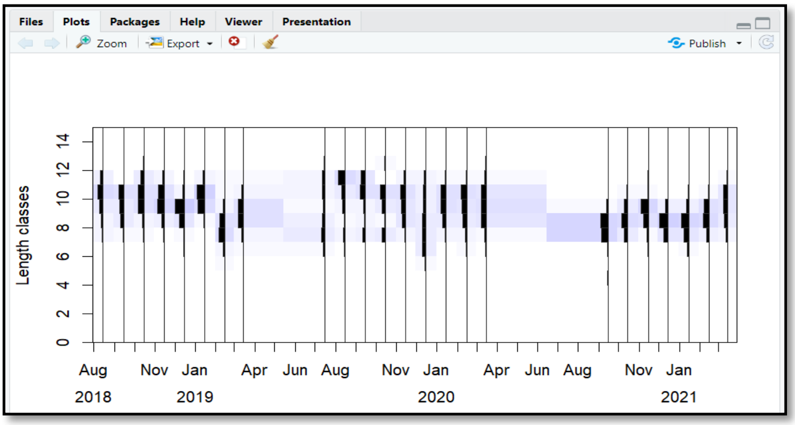
Load the library TropFishR and create a length frequency (LFQ) file using the following codes:

```
library(TropFishR)
rawdata$date<- as.Date(rawdata$date, format = "%d.%m.%Y")
my_data<- lfqCreate(data = rawdata, Lname = "length", Dname = "date")
```

2.1.3. Controlling the length class interval (Bin size)

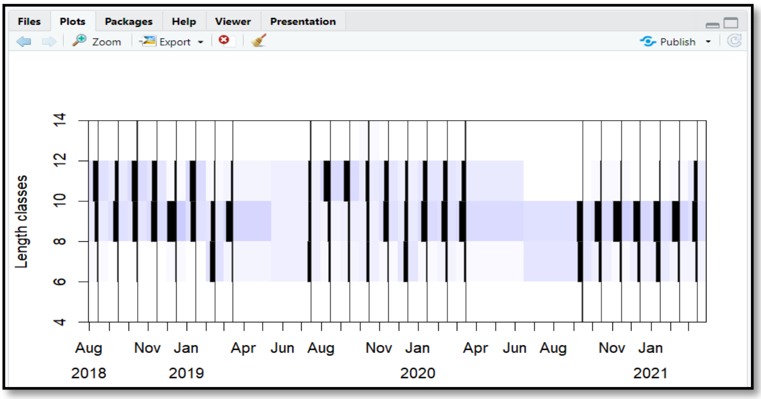
Use the following code to set the bin size (length class interval) to 1 cm and plot the output:

```
my_data1 <- lfqModify(my_data, bin_size = 1)
plot(my_data1, Fname = "catch")
```



Use the following code to reset the bin size (length class interval) to 2 cm and plot the output.

```
my_data2 <- lfqModify(my_data, bin_size = 2)
plot(my_data2, Fname = "catch")
```



> ***Note:*** *The length class interval can be adjusted by changing the* ***'bin_size'*** *parameter. The optimal length class interval should be determined by evaluating the* ***'Rn score'*** *and ensuring that the derived growth parameters align with the biological rationale for the species. According to Wang et al. (2020), the optimal bin size (OBS) can be selected using the following rule of thumb:* $OBS = 0.23 \times L_{max}{}^{0.6}$

Once the best length class interval for the data has been identified, save the dataset as my_data for use in subsequent analysis. For example, to save the length frequencies with 1 cm class intervals (e.g., my_data1), use the following code: `my_data<- my_data1`

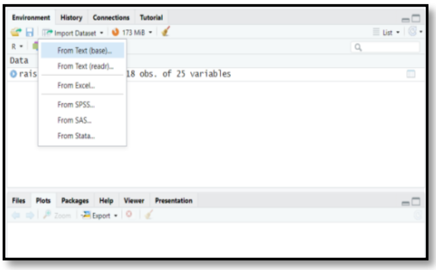
2.1.4. Raising the length frequency data

The unraised length frequency data can be converted to monthly raised frequencies by multiplying it with the raising factors (i.e., monthly landings/sample weight) for each month.

To import the monthly raising factors, click on the 'Import Dataset' option in the 'Environment' tab (top right panel), and then select 'From Excel'. Browse for the Excel file (e.g., the RF sheet in the rawdata Excel file) and import it.

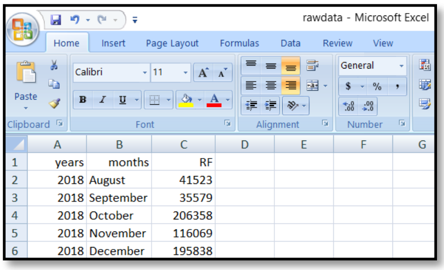
Environment> Import Dataset> from Excel and then browse the file on disk and import.

Excel Import window



Data
rais 18 obs. of 25 variables

Raising factor data (RF sheet)



	A	B	C	D	E	F	G
1	years	months	RF				
2	2018	August	41523				
3	2018	September	35579				
4	2018	October	206358				
5	2018	November	116069				
6	2018	December	195838				

raising_factors<-rep(rawdata$RF, each=length(my_data$midLengths))

my_data$catch<- my_data$catch*raising_factors

Note: *Although growth and mortality parameters can be assessed using the unraised monthly length frequencies (LFQ file), it is generally preferable to raise the data for subsequent analysis, such as virtual population analysis, stock yield, and biomass estimation.*

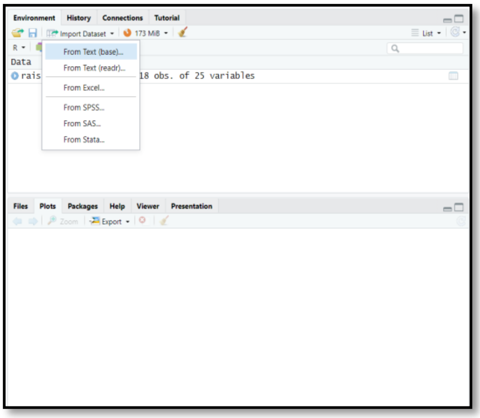
2.1.5. Creating a length frequency file (LFQ) on R using Data Format 2 (Raised data)

Raised data (Dates, length ranges and frequencies) needs to be formatted to create length frequencies (LFQ) for different months. For ease of splitting the dataset by month, use date formatting with a prefixed capital 'X' in the month headers.

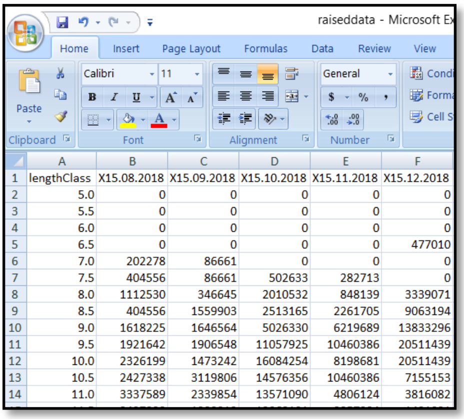
Click on the 'Import Dataset' option in the 'Environment' tab (top right panel) and select 'From Excel'. Browse for the Excel file (e.g., the 'lfq' data sheet in the 'raiseddata' Excel file) and import it.

Environment> Import Dataset> from Excel and then browse the file on disk and import.

Excel import window



Raised dataset (lfq)
Dates, length ranges, and frequencies



	A	B	C	D	E	F
1	lengthClass	X15.08.2018	X15.09.2018	X15.10.2018	X15.11.2018	X15.12.2018
2	5.0	0	0	0	0	0
3	5.5	0	0	0	0	0
4	6.0	0	0	0	0	0
5	6.5	0	0	0	0	477010
6	7.0	202278	86661	0	0	0
7	7.5	404556	86661	502633	282713	0
8	8.0	1112530	346645	2010532	848139	3339071
9	8.5	404556	1559903	2513165	2261705	9063194
10	9.0	1618225	1646564	5026330	6219689	13833296
11	9.5	1921642	1906548	11057925	10460386	20511439
12	10.0	2326199	1473242	16084254	8198681	20511439
13	10.5	2427338	3119806	14576356	10460386	7155153
14	11.0	3337589	2339854	13571090	4806124	3816082

Note: *Use length measurements in centimeters (cm) to ensure uniformity in subsequent calculations. Ensure that no column under any month header is entirely empty, NA, or zero. There should be at least one observation (catch number or frequency) for any length class in each month within the dataset. If a month has no observed frequency, it should be excluded from the dataset. Including months with no observed frequencies will cause errors.*

Load the library TropFishR and create a length frequency file (LFQ) using the following codes:

library(TropFishR)

dates <- colnames(raiseddata)[-1]

dates <- strsplit(dates, "X")

dates <- unlist(lapply(dates, function(x) x[2]))

dates <- as.Date(dates, "%d.%m.%Y")

Specify variables like midLengths, dates and month-wise catches using the following code:

```
my_data <- list(dates = dates, midLengths = raiseddata$lengthClass, catch = as.matrix(raiseddata[,-1]))
```

Convert the above file (e.g., my_data) to a length frequency file (LFQ) using the following code: `class(my_data) <- "lfq"`

Fill the empty cells (if any) in the newly created lfq file with zeros using the following code:

```
my_data$catch[is.na(my_data$catch)] <- 0
```

Plot the newly created length frequency file (e.g., my_data) using the following code:

```
plot(my_data, Fname = "catch")
```

2.1.6. Controlling length class interval (Bin size)

Use the following code to set the bin size (length class interval) to 1 cm and plot the output:

```
my_data1 <- lfqModify(my_data, bin_size = 1)
plot(my_data1, Fname = "catch")
```

Use the following code to set the bin size (length class interval) to 2 cm and plot the output:

```
my_data2 <- lfqModify(my_data, bin_size = 2)
plot(my_data2, Fname = "catch")
```

> ***Note:*** *The desired length class interval can be adjusted by changing the **'bin_size'** parameter. The optimal length class interval should be determined by evaluating the **'Rn score'** and ensuring that the derived growth parameters align with the biological rationale for the species. According to Wang et al. (2020), the optimal bin size (OBS) can be selected using the following rule of thumb:* $OBS = 0.23 \times {L_{max}}^{0.6}$

Once the best length class interval for the data has been identified, save the dataset as my_data for use in subsequent analysis. For example, to save the length frequencies with 1 cm class intervals (e.g., my_data1), use the following code: `my_data<- my_data1`

The LFQ file created from data format 1 or 2 can be further processed to estimate the Von Bertalanffy's growth parameters (1957), i.e., L∞ (asymptotic length) and K (growth coefficient) using the Electronic Length Frequency Analysis (ELEFAN) routine of TropFishR package which provides following four approaches: (1) ELEFAN: response surface analysis (RSA), (2) ELEFAN: K-scan, (3) ELEFAN_SA: ELEFAN simulated annealing, and (4) ELEFAN_GA: ELEFAN genetic algorithm.

2.1.7. ELEFAN: Response surface analysis (RSA)

In this analysis, various combinations of L∞ and K are explored to identify the growth parameters that best fit the restructured length-frequency data. The best combination is identified using the Rn-score. TropFishR provides two options for response surface analysis: (1) the Cross method, which is the default method used in the old FiSAT software, and (2) the Optimize method, a computationally expensive yet more sophisticated new method available in TropFishR.

Preparing L∞ and K ranges for RSA

Guess L∞ using the following empirical relationship between the asymptotic length (L∞) and the maximum observed length (L_{max}):

$$L_\infty = \frac{L_{max}}{0.95}$$

Provide information about the maximum observed length (Lmax) as follows:

```
Lmax<-13.15
linf_guess <- Lmax / 0.95
```

To create a 5% lower and upper range for the guessed L∞, use the following codes:

```
linf_guess_range<- linf_guess*(5/100)
linf_guess_low <- linf_guess- linf_guess_range
linf_guess_high <-linf_guess+linf_guess_range
```

Guess K based on the biological traits of the species, such as growth rate (fast vs. slow) or longevity (short-lived vs. long-lived). If the species' longevity (t_{max}) is known, the user can use the following empirical relationship between t_{max} and K to estimate K:

$$K = \frac{3}{t_{max}}$$

For example, if a fast-growing shrimp lives up to 1.5 years (t_{max}), then its K can be estimated as follows: `K_guess<-3/1.5`

To create a 10% lower and upper range for the guessed K, use the following codes:

```
K_guess_range <-K_guess*(10/100)
K_guess_low <-K_guess-K_guess_range
K_guess_high <-K_guess+K_guess_range
```

> ***Note:*** *The initial guess range for L∞ and K can be further fine tuned by multiplying with required range factor (ex: 5/100 for 5% range) after getting the results from the initial run. Initially, use a broad range of 10% and then squeeze the range based on output from analysis.*

RSA Cross' method (default method of FiSAT II)

Automatic approach:

The system automatically aligns the growth function to intersect (cross) the bin (length class) with the maximum positive score.

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= TRUE, Linf_range = seq(linf_guess_low, linf_guess_high, 0.1), K_range = (seq(K_guess_low, K_guess_high, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

In the code provided above, the user can directly specify the range for an initial guess of L∞ and K. For example, the user might set the L∞ range from 13.5 to 14.5 cm and the K range from 1.5 to 2.0 yr^{-1} as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= TRUE, Linf_range = seq(13.5, 14.5, 0.1), K_range = (seq(1.5, 2, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

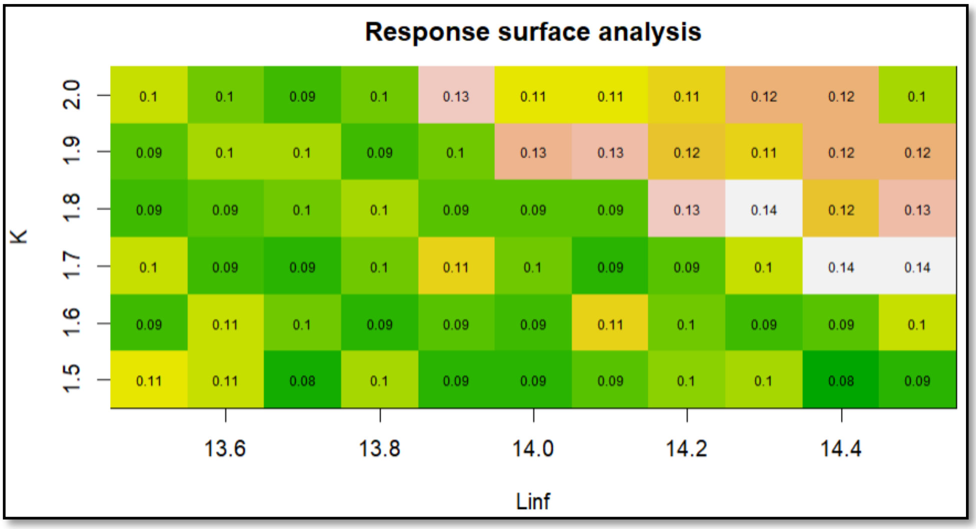
Manual approach:

The user can specify the growth function to intersect a required bin (length class) by setting the specific date (e.g., first month) and length (e.g., second mid-length) as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= FALSE, Linf_range = seq(linf_guess_low, linf_guess_high, 0.1), K_range = (seq(K_guess_low, K_guess_high, 0.1)), cross.date = my_data$dates[1], cross.midLength = my_data$midLengths[2], contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

In the code provided above, the user can directly specify the range for an initial guess of L_∞ and K. For example, the user might set the L_∞ range from 13.5 to 14.5 cm and the K range from 1.5 to 2.0 yr^{-1} as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= FALSE, Linf_range = seq(13.5, 14.5, 0.1), K_range = (seq(1.5, 2, 0.1)), cross.date = my_data$dates[1], cross.midLength = my_data$midLengths[2], contour = FALSE, MA = 5, addl.sqrt = TRUE)
```



RSA optimize method (new method of TropFishR)

It is a sophisticated but computationally expensive approach in TropFishR that solves for t_anchor through a maximization process of the reconstructed score.

```
fit1 <- ELEFAN(lfq = my_data, method = "optimise", Linf_range = seq(linf_guess_low, linf_guess_high, 0.1), K_range = (seq(K_guess_low, K_guess_high, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

In the code provided above, the user can directly specify the range for an initial guess of L_∞ and K. For example, the user might set the L_∞ range from 13.5 to 14.5 cm and the K range from 1.5 to 2.0 yr^{-1} as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "optimise", Linf_range = seq(13.5, 14.5, 0.1), K_range = (seq(1.5, 2, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

Understanding the RSA output

The result shows that L_∞ and K at a combination of 14.3-14.5 cm and 1.7-1.8 yr^{-1} yield the highest Rn-score. This output can be used to narrow down the search range for L_∞ and K, potentially improving the results further.

Note: *Biological characteristics such as L_{max}, t_{max}, and expected growth performance should be given due consideration along with the Rn-score when determining the best combination pair for L_∞ and K.*

2.1.8. ELEFAN: K-scan

The analysis is similar to the RSA described above, except that in K-Scan, L_∞ is fixed, and a range of K values is scanned in combination with the fixed L_∞ to identify the growth parameters that best fit the restructured length-frequency data. The best combination is identified using the Rn score. As with RSA, TropFishR provides two options for K-Scan analysis: (1) the Cross method, which is the default method used in the old FiSAT software, and (2) the Optimize method, a computationally expensive but more sophisticated approach available in TropFishR.

Preparing a fixed L_∞ and K range for K-scan

Guess L_∞ using the following empirical relationship between the asymptotic length (L_∞) and the maximum observed length (L_{max}):

$$L_\infty = \frac{L_{max}}{0.95}$$

Provide information about the maximum observed length (Lmax) as follows:

```
Lmax<-13.15
linf_guess <- Lmax / 0.95
```

To create a 5% lower and upper range for the guessed L_∞, use the following codes:

```
linf_guess_range<- linf_guess*(5/100)
linf_guess_low <- linf_guess- linf_guess_range
linf_guess_high <-linf_guess+linf_guess_range
```

Guess K based on the biological traits of the species, such as growth rate (fast vs. slow) or longevity (short-lived vs. long-lived). If the species' longevity (t_{max}) is known, the user can use the following empirical relationship between t_{max} and K to estimate K:

$$K = \frac{3}{t_{max}}$$

For example, if a fast-growing shrimp lives up to 1.5 years, then its K can be estimated as:

```
K_guess<-3/1.5
```

To create a 10% lower and upper range for the guessed K, use the following codes:

```
K_guess_range <-K_guess*(10/100)
K_guess_low <-K_guess-K_guess_range
K_guess_high <-K_guess+K_guess_range
```

> ***Note:*** *The initial guess range for L_{∞} and K can be further fine tuned by multiplying with required range factor (ex: 5/100 for 5% range) after getting the results from the initial run. Initially, use a broad range of 10% and then squeeze the range based on output from analysis.*

K-scan cross' method (Default method of FiSAT II)

Automatic approach:

The system automatically aligns the growth function to intersect (cross) the bin (length class) with the maximum positive score.

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= TRUE, Linf_fix = linf_guess, K_range = (seq(K_guess_low, K_guess_high, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

In the above code, the user can directly specify the L_{∞} and the range for an initial guess of K. For example, the user might set L_{∞} to 14 cm and the K range from 1.5 to 2.0 yr^{-1} as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= TRUE, Linf_fix = 14.0, K_range = (seq(1.5, 2, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

Manual approach:

The user can specify the growth function to intersect a required bin (length class) by setting the specific date (e.g., first month) and length (e.g., second mid-length) as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= FALSE, Linf_fix = linf_guess, K_range = (seq(K_guess_low, K_guess_high, 0.1)), cross.date = my_data $dates[1], cross.midLength = my_data $midLengths[2], contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

In the above code, the user can directly specify the L_{∞} and the range for an initial guess of K. For example, the user might set L_{∞} to 14 cm and the K range from 1.5 to 2.0 yr^{-1} as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "cross", cross.max= FALSE, Linf_fix = 14.0, K_range = (seq(1.5, 2, 0.1)), cross.date = my_data $dates[1], cross.midLength = my_data $midLengths[2], contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

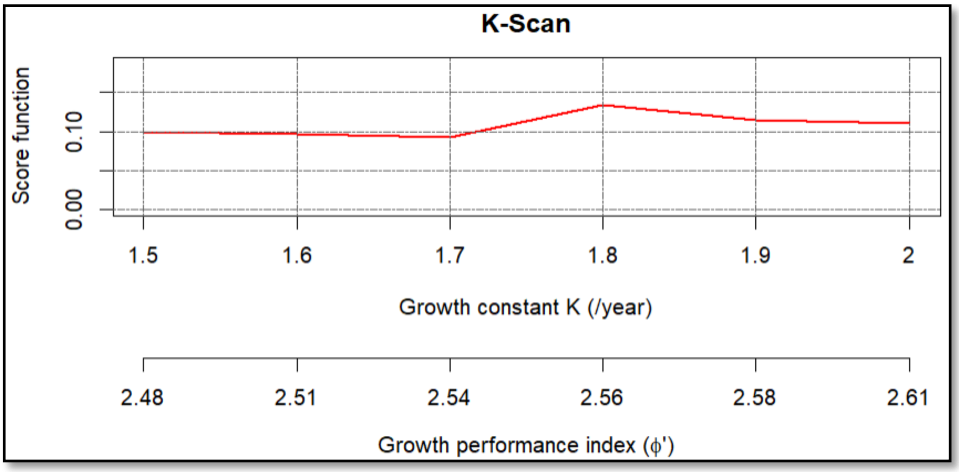
K-scan optimise method (New method of TropFishR)

It is a sophisticated but computationally expensive approach in TropFishR that solves for t_anchor through a maximization process of the reconstructed score.

```
fit1 <- ELEFAN(lfq = my_data, method = "optimise", Linf_fix = linf_guess, K_range = (seq(K_guess_low, K_guess_high, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```

In the code provided above, the user can directly specify the L_∞ and the range for an initial guess of K. For example, the user might set L_∞ to 14 cm and the K range from 1.5 to 2.0 yr^{-1} as follows:

```
fit1 <- ELEFAN(lfq = my_data, method = "optimise", Linf_fix = 14.0, K_range =
(seq(1.5, 2, 0.1)), contour = FALSE, MA = 5, addl.sqrt = TRUE)
```



Understanding the K-scan output

The output shows that for the given L_∞ and K range, the highest Rn-score is attained at a K value of 1.8 yr^{-1}. This output can be used as input to narrow down the search range for K to further improve the results.

> ***Note:*** *Ideally, the output from the response surface analysis (RSA) should be used to refine the K value through the K-Scan procedure. However, the K-Scan can also be conducted independently if one is confident about the L_∞ and only seeks to determine the K. Biological characteristics such as L_{max}, t_{max}, and expected growth performance should be carefully considered, along with the Rn-Score, when determining the best K value.*

2.1.9. ELEFAN_SA: ELEFAN Simulated Annealing

In addition to the traditional ELEFAN methods mentioned above (RSA and K-Scan), TropFishR also offers two new optimization procedures for simultaneously deriving life history parameters: (1) ELEFAN based on simulated annealing (ELEFAN_SA) and (2) ELEFAN based on genetic algorithms (ELEFAN_GA) (Taylor & Mildenberger, 2017). The ELEFAN_SA routine requires an educated guess for the initial, lower, and upper values of L_∞, K, and t-anchor.

Preparing L_∞ and K ranges for ELEFAN_SA

Guess L_∞ using the following empirical relationship between the asymptotic length (L_∞) and the maximum observed length (L_{max}):

$$L_\infty = \frac{L_{max}}{0.95}$$

Provide information about the maximum observed length (Lmax) as follows:

```
Lmax<-13.15
linf_guess <- Lmax / 0.95
```

To create a 5% lower and upper range for the guessed L_∞, use the following codes:

```
linf_guess_range<- linf_guess*(5/100)
linf_guess_low <- linf_guess- linf_guess_range
linf_guess_high <-linf_guess+linf_guess_range
```

Guess K based on the biological traits of the species, such as growth rate (fast vs. slow) or longevity (short-lived vs. long-lived). If the species' longevity (t_{max}) is known, the user can use the following empirical relationship between t_{max} and K to estimate K:

$$K = \frac{3}{t_{max}}$$

For example, if a fast-growing shrimp lives up to 1.5 years (t_{max}), then its K can be estimated as follows:

```
K_guess<-3/1.5
```

To create a 10% lower and upper range for the guessed K, use the following codes:

```
K_guess_range <-K_guess*(10/100)
K_guess_low <-K_guess-K_guess_range
K_guess_high <-K_guess+K_guess_range
```

Note: *The initial guess range for L_∞ and K can be further fine tuned by multiplying with required range factor (ex: 5/100 for 5% range) after getting the results from the initial run. Initially, use a broad range of 10% and then squeeze the range based on output from analysis.*

The code to implement ELEFAN_SA is given below:

```
fitSA <- ELEFAN_SA (my_data, seasonalised = FALSE, maxit = NULL, SA_time = 60*1,
init_par = list(Linf= linf_guess, K= K_guess, t_anchor=0.5, ts=0.5, C=0.5),
low_par = list(Linf= linf_guess_low, K=K_guess_low, t_anchor=0.0, ts=0.0,
C=0.0), up_par = list(Linf= linf_guess_high, K= K_guess_high, t_anchor=1.0,
ts=1.0, C=1.0), MA = 5, addl.sqrt = TRUE)
```

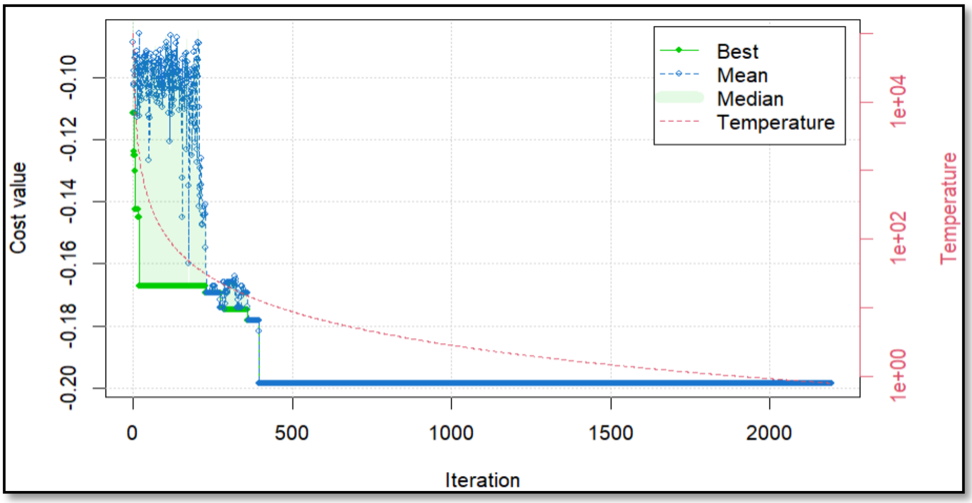
In the code provided above, the user can directly specify initial guess for L_∞ (e.g., 14 cm) and K (e.g., 1.8 yr^{-1}), along with their lower (e.g., L_∞ = 13.5 cm and K = 1.5 yr^{-1}) and upper (e.g., L_∞ = 14.5 cm and K = 2.0 yr^{-1}) bounds as follows:

```
fitSA <- ELEFAN_SA (my_data, seasonalised = FALSE, maxit = NULL, SA_time = 60*1,
init_par = list(Linf=14.0, K=1.8, t_anchor=0.5, ts=0.5, C=0.5), low_par =
list(Linf=13.5, K=1.5, t_anchor=0.0, ts=0.0, C=0.0), up_par = list(Linf=14.5,
K=2.0, t_anchor=1.0, ts=1.0, C=1.0), MA = 5, addl.sqrt = TRUE)
```

Note: *Considering the tropical condition, the seasonal oscillation in growth has been turned off (seasonalised = **FALSE**). If seasonalised = **FALSE,** then there is no need to provide initial, lower limit and upper limit values for the ts and C in the above-mentioned codes (ts=NA, C=NA). The t-*

anchor (represents the starting point of ELEFAN 1 in Fisat II) is the point that anchors the growth curve to the X-axis (time or age axis) and also represents the peak spawning period. Its value varies from 0 to 1, where 0 represents 1st January; 0.5 represents June; and 0.999 represents 31st December (i.e., Month = t-anchor ×12). Initially, use a broader range for L∞ and K, and then gradually narrow down the range. While determining the best L∞ and K values, consider biological characteristics such as L_{max}, t_{max}, and expected growth performance, in addition to the Rn-score.

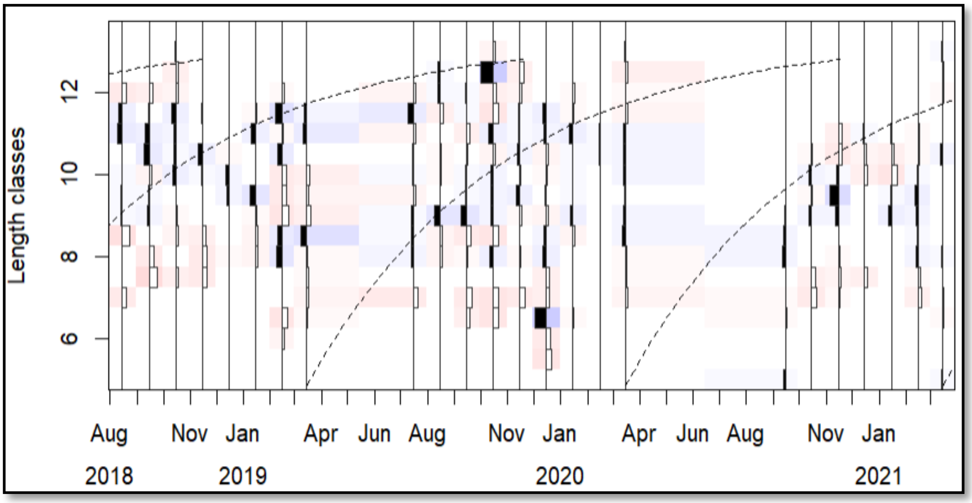
Iteration diagnostic plot



Understanding the ELEFAN_SA diagnostic plot

Better results will show a greater overlap between the best values (green points) and the mean values (blue points), leading to smaller median values (showed by the light green area) in the diagnostic plot.

To plot the VBGF growth parameters, use the following code: plot(fitSA)



To get the growth parameters such as L_{∞}, K, recruitment period (t_anchor) and growth performance index (phiL), use the following code: fitSA$par

To get the goodness-of-fit parameter (Rn-score), use the following code: fitSA$Rn_max

2.1.10. ELEFAN_GA: ELEFAN Genetic Algorithm

This new optimization procedure in TropFishR uses ELEFAN based on genetic algorithms (ELEFAN_GA) for deriving growth parameters (Taylor & Mildenberger, 2017). Unlike ELEFAN_SA, ELEFAN_GA does not require an educated guess for the initial values of L_{∞}, K, and t-anchor in the code. Instead, it derives the growth parameters based on an educated guess for the lower and upper values of L_{∞}, K, and t-anchor.

Preparing L_{∞} and K ranges for ELEFAN_GA

Guess L_{∞} using the following empirical relationship between the asymptotic length (L_{∞}) and the maximum observed length (L_{max}):

$$L_{\infty} = \frac{L_{max}}{0.95}$$

Provide information about the maximum observed length (Lmax) as follows:

```
Lmax<-13.15
linf_guess <- Lmax / 0.95
```

To create a 5% lower and upper range for the guessed L_{∞}, use the following codes:

```
linf_guess_range<- linf_guess*(5/100)
linf_guess_low <- linf_guess- linf_guess_range
linf_guess_high <-linf_guess+linf_guess_range
```

Guess K based on the biological traits of the species, such as growth rate (fast vs. slow) or longevity (short-lived vs. long-lived). If the species' longevity (t_{max}) is known, the user can use the following empirical relationship between t_{max} and K to estimate K:

$$K = \frac{3}{t_{max}}$$

For example, if a fast-growing shrimp lives up to 1.5 years (t_{max}), then its K can be estimated as follows: K_guess<-3/1.5

To create a 10% lower and upper range for the guessed K, use the following codes:

```
K_guess_range <-K_guess*(10/100)
K_guess_low <-K_guess-K_guess_range
K_guess_high <-K_guess+K_guess_range
```

Note: *The initial guess range for L_{∞} and K can be further fine tuned by multiplying with required range factor (ex: 5/100 for 5% range) after getting the results from the initial run. Initially, use a broad range of 10% and then squeeze the range based on output from analysis.*

The code to implement ELEFAN_GA is given below:

```
fitGA <- ELEFAN_GA (my_data, seasonalised = FALSE, maxiter = 100, low_par = list(Linf= linf_guess_low, K= K_guess_low, t_anchor=0.0, ts=0.0, C=0.0),
```

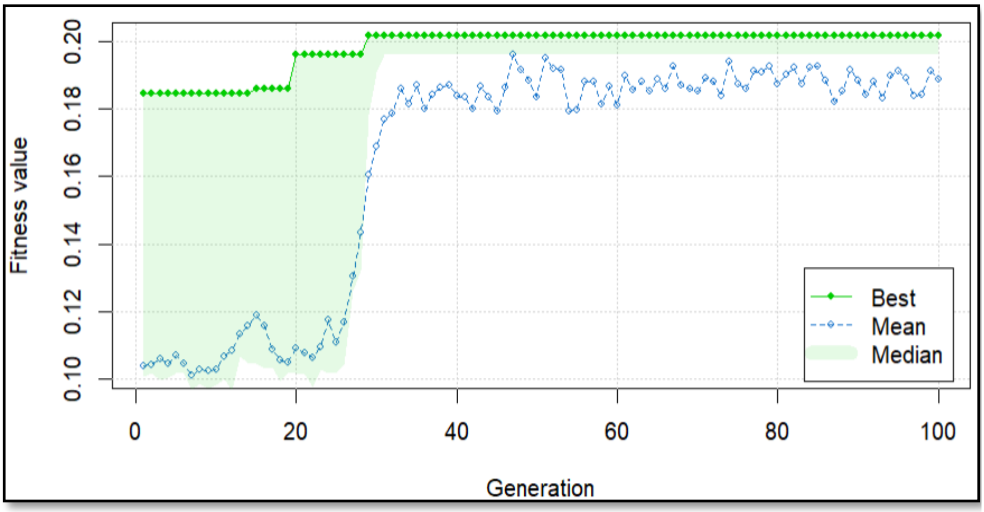
up_par = list(Linf= linf_guess_high, K= K_guess_high, t_anchor=1.0, ts=1.0, C=1.0), MA = 5, addl.sqrt = TRUE)

In the code provided above, the user can directly specify initial guess for the lower (e.g., L∞ = 13.5 cm and K = 1.5 yr^{-1}) and upper (e.g., L∞ = 14.5 cm and K = 2.0 yr^{-1}) bounds of L∞ and K as follows:

fitGA <- **ELEFAN_GA**(my_data, **seasonalised = FALSE**, maxiter = 100, **low_par = list(Linf=13.5, K=1.5, t_anchor=0.0,** ts=0.0, C=0.0), **up_par = list(Linf=14.5, K=2.0, t_anchor=1.0,** ts=1.0, C=1.0), MA = 5, addl.sqrt = TRUE)

> ***Note:*** *Considering the tropical condition, the seasonal oscillation in growth has been turned off (seasonalised =* ***FALSE****). If seasonalised =* ***FALSE,*** *then there is no need to provide initial, lower limit and upper limit values for the ts and C in the above codes (ts=NA, C=NA). The t-anchor (represents the starting point of ELEFAN 1 in Fisat II) is the point that anchors the growth curve to the X-axis (time or age axis) and also represents the peak spawning period. Its value varies from 0 to 1, where 0 represents 1st January; 0.5 represents June; and 0.999 represents 31st December (i.e., Month = t-anchor ×12). Initially, use a broader range for L∞ and K, and then gradually narrow down the range. While determining the best L∞ and K values, consider biological characteristics such as L_{max}, t_{max}, and expected growth performance, in addition to the Rn-score.*

Iteration diagnostic plot



Understanding the ELEFAN_GA diagnostic plot

Better results will show a greater overlap between the best values (green points) and the mean values (blue points), leading to lower median values (represented by the intermittent light green area) in the diagnostic plot.

To get the growth parameters such as L∞, K, recruitment period (t_anchor) and growth performance index (phiL), use the following code:

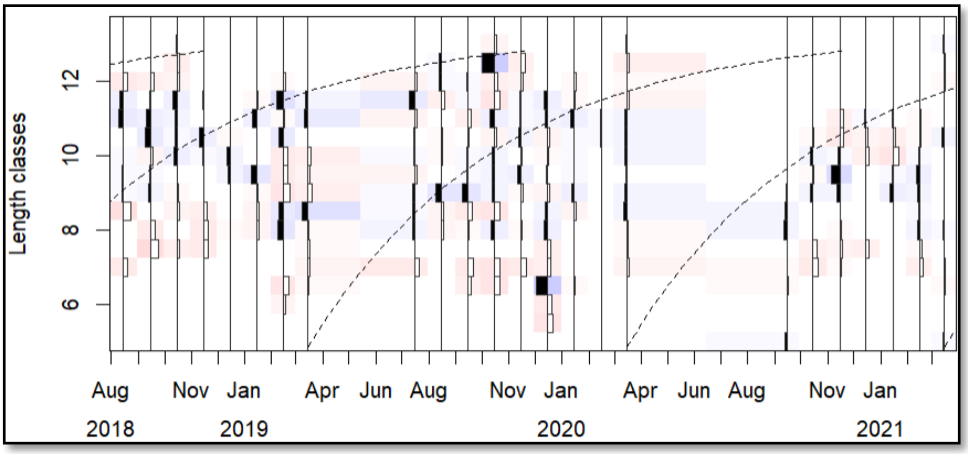
fitGA$par

To get the goodness-of-fit parameter (Rn score), use the following code:

```
fitGA$Rn_max
```

To simply plot the growth parameters, use the following code:

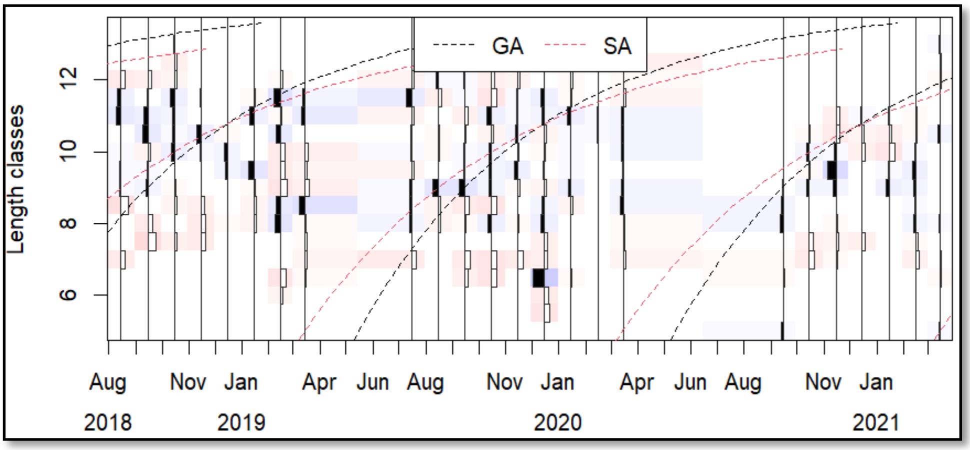
```
plot(fitGA)
```



2.1.11. ELEFAN comparison plot

To compare the genetic algorithm derived parameters with simulated annealing derived parameters

```
plot(fitGA, draw = FALSE)
#Plot genetic algorithm derived parameters
lfqFitCurves(fitGA, par= fitGA$par, draw = TRUE, col=1, flagging.out = FALSE)$fESP
#Plot simulated annealing derived parameters
lfqFitCurves(fitSA, par= fitSA$par, draw = TRUE, col=2, flagging.out = FALSE)$fESP
#Add legends
legend("top", legend=c("GA", "SA"), lty=2, col=1:2, ncol=2)
```



2.1.12. Estimation of mean and confidence intervals of VBGF growth parameters

The Jackknife (JK) resampling technique allows for the estimation of mean and confidence intervals for the VBGF parameters. In this statistical method, the growth parameters are recalculated repeatedly using either the ELEFAN_SA or ELEFAN_GA approach by resampling the LFQ data from the available months, excluding the length frequency data of one month each time. This resampling is repeated for every available month in the sample, generating a pool of VBGF parameters that is used to derive the confidence intervals. Any of the following two approaches can be used to derive the mean and confidence intervals of VBGF growth parameters:

Through Simulated Annealing (Using ELEFAN_SA)

The codes given below generate growth parameter values, their mean and confidence intervals using simulated annealing.

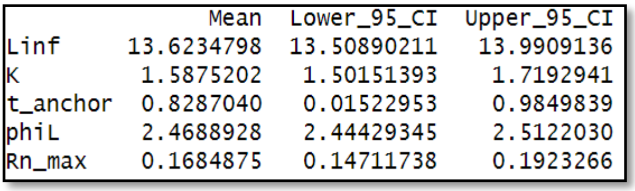
```
JK <- vector("list", length(my_data$dates))
for (i in 1:length(my_data$dates)){
loop_data <- list(dates = my_data$dates[-i],
midLengths = my_data$midLengths,
catch = my_data$catch[,-i])
JK_MODEL <- ELEFAN_SA(loop_data, seasonalised = FALSE, maxit = 100, SA_time = 60*1, init_par = list(Linf=14.0, K=1.8, t_anchor=0.5, ts=0.5, C=0.5), low_par = list(Linf=13.5, K=1.5, t_anchor=0.0, ts=0.0, C=0.0), up_par = list(Linf=14.5, K=2.0, t_anchor=1.0, ts=1.0, C=1.0), MA = 5, addl.sqrt = TRUE)
JK[[i]] <- unlist(c(JK_MODEL$par, list(Rn_max= JK_MODEL$Rn_max)))}
{JKres <- do.call(cbind, JK)
JKmeans <- apply(as.matrix(JKres), MARGIN = 1, FUN = mean)
JKconf <-apply(as.matrix(JKres), MARGIN = 1, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
JKconf <- t(JKconf)
JKvalues<-cbind(JKmeans, JKconf)
colnames(JKvalues) <- c("mean","lower","upper")}
JKvalues
```

Through Genetic Algorithm (Using ELEFAN_GA)

The codes given below generate growth parameter values, their mean and confidence intervals using genetic algorithm.

```
JK <- vector("list", length(my_data$dates))
for (i in 1:length(my_data$dates)){
loop_data <- list(dates = my_data$dates[-i],
midLengths = my_data$midLengths,
catch = my_data$catch[,-i])
JK_MODEL <- ELEFAN_GA(loop_data, seasonalised = FALSE, maxiter = 100, low_par = list(Linf=13.5, K=1.5, t_anchor=0.0, ts=0.0, C=0.0), up_par = list(Linf=14.5, K=2.0, t_anchor=1.0, ts=1.0, C=1.0), MA = 5, addl.sqrt = TRUE)
JK[[i]] <- unlist(c(JK_MODEL $par, list(Rn_max= JK_MODEL $Rn_max)))}
{JKres <- do.call(cbind, JK)
JKmeans <- apply(as.matrix(JKres), MARGIN = 1, FUN = mean)
JKconf <-apply(as.matrix(JKres), MARGIN = 1, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
JKconf <- t(JKconf)
growth_confidence<-cbind(JKmeans, JKconf)
colnames(growth_confidence) <- c("Mean"," Lower_95_CI"," Upper_95_CI")}
growth_confidence
```

Output



```
                Mean  Lower_95_CI  Upper_95_CI
Linf      13.6234798  13.50890211   13.9909136
K          1.5875202   1.50151393    1.7192941
t_anchor   0.8287040   0.01522953    0.9849839
phiL       2.4688928   2.44429345    2.5122030
Rn_max     0.1684875   0.14711738    0.1923266
```

Note: *The Jackknife resampling procedure takes a considerable amount of time to complete. The number of iterations (maxit) and annealing time (SA_time) in ELEFAN_SA, as well as the number of iterations (maxiter) in ELEFAN_GA, can be increased to improve the accuracy of the analysis; however, this will also significantly increase the analysis time. The initial, upper and lower range of Linf and K used in JK_MODEL above are just some examples for illustration purpose. Proceed with this procedure only after finalizing the VBGF parameters using the simulated annealing approach in ELEFAN (ELEFAN_SA) or the genetic algorithm approach in ELEFAN (ELEFAN_GA). Use your initial parameters (init_par), lower parameters (low_par), and upper parameters (up_par) that were used in ELEFAN_SA or ELEFAN_GA when finalizing the VBGF parameters.*

2.2. Estimation of gestation period or hatching time (t_o)

The time when the length of the fish is theoretically zero is denoted as t_o. Since fish is born with a length (size at birth, L_o) and age starts from the time of birth (t=0), the time required to reach the size at birth (L_o) from a theoretical zero length is usually expressed as a negative number (due to its position on the left side of the origin on the X-axis or the time axis). It is considered as the time taken by the fish to reach the size at birth (L_o) from a theoretically zero length and is also used as a proxy for the hatching or gestation period (t_o).

2.2.1. Requirements for t_0 calculation

Essential parameters

It is necessary to save the essential growth parameters (L_∞ and K) in the LFQ file (e.g., my_data) for the calculation of t_0. To assign the mean values directly from the Jack Knife approach, use the following codes:

```
my_data$Linf<-as.numeric (growth_confidence[1])
my_data$K<-as.numeric (growth_confidence[2])
```

User can directly assign the derived growth parameters (e.g., L_∞ = 13.95 cm and K = 1.71 yr^{-1}) in the LFQ file using the following codes:

```
my_data$Linf<-13.95
my_data$K<-1.71
```

The rearranged VBGF equation for t_0 also requires an additional parameter, i.e., the length at birth (L_0) in 'cm' (e.g., for a shrimp, assume the larval size on the day of hatching (0th day) is 0.025 cm), which can be assigned using the following code:

```
L0<-0.025
```

TropFishR does not provide an inbuilt routine for calculating the time when the length of the fish is theoretically zero (t_0). Nevertheless, t_0 can be calculated using the following two equations:

2.2.2. Empirical equation for t_0

It can be estimated using the empirical formula suggested by Pauly (1983), as shown below.

$$t_0 = 10^{(-0.3922-0.2752*log_{10}(L_\infty)-1.038*log_{10}(K))}$$

Use the following R code to calculate t_0, although the result may not be very reasonable:

```
tzero <- -10^(-0.3922-(0.2752*log10(my_data$Linf))-(1.038*log10(my_data$K)))
tzero
```

Alternatively, the user can directly assign the derived growth parameters (e.g., L_∞ = 13.95 cm and K = 1.71 yr^{-1}) in the code as follows:

```
tzero <- -10^(-0.3922-(0.2752*log10(13.95))-(1.038*log10(1.71)))
tzero
```

2.2.3. Rearranged VBGF equation for t_0

The t_0 can be precisely back-calculated using the length at birth (L_0) in the rearranged VBGF equation (von Bertalanffy, 1938), as outlined below:

$$t_0 = \frac{1}{K} \times ln\left[1 - \left(\frac{L_0}{L_\infty}\right)\right]$$

The length at birth (L_0) can be obtained from previously published literature. Define length at birth (L_0) in 'cm' (e.g., for a shrimp, assume the larval size on the day of hatching (0th day) is 0.025 cm) using the following code: `L0<-0.025`

Calculate t_0 using the following code:

```
tzero<-1/my_data$K*log((1-(L0/my_data$Linf)))
tzero
```

Alternatively, the user can directly assign the derived growth parameters (e.g., L_∞ = 13.95 cm and K = 1.71 yr^{-1}) and the L0 (0.025 cm) in the code as follows:

```
tzero<-1/1.71*log((1-(0.025/13.95)))
tzero
```

2.2.4. Estimation of mean and confidence intervals of t_0

To get the confidence intervals for t_0, a sample of t_0 values is estimated using the derived growth parameters (L_∞ and K) from the Jack Knife resampling procedure mentioned above. To save the individual outputs of each growth parameter derived from the Jack Knife resampling procedure as a data frame, first use the following code:

```
growthparamdata <-as.data.frame(t(JKres))
```

To generate t_0 values, along with their mean and confidence intervals, use the following codes:

```
t_zeros_function = function(x, growthparamdata){
Linf = x[1]
K = x[2]
return(1/K*log((1-(L0/Linf))))}
t_zeros<-apply(growthparamdata,1, t_zeros_function)
t_zeros_mean <- apply(matrix(t_zeros), MARGIN = 2, FUN = mean)
t_zeros_conf <- apply(matrix(t_zeros), MARGIN = 2, FUN = function(x) quantile(x,
probs=c(0.025,0.975)))
print(c(Mean=t_zeros_mean, Lower_95_CI=t_zeros_conf[1,1],
Upper_95_CI=t_zeros_conf[2,1]))
```

Add the t_0 values to 'growthparamdata' data frame using the following code:

```
growthparamdata$t_zero<-t_zeros
```

2.3. Estimation of longevity/ maximum age (t_{max})

The longevity is the maximum age (t_{max}) to which the fish survives in the absence of exploitation or fishing mortality (F = 0, so Z = M). In the temperate conditions, it is estimated directly from the hard part analysis, as it is easy to age the temperate species that exhibit distinct growth rings due to seasonal changes. However, in tropical conditions, it is very challenging to directly age the fish using hard part analysis, as tropical species often grow continuously due to relatively stable environmental conditions. In tropical species, the t_{max} is often estimated using life history theory as a function of growth coefficient (K) or natural mortality rate (M).

2.3.1. Requirements for t_{max} calculation

Essential parameters

It is necessary to save the essential growth parameters (L_∞ and K) in the LFQ file (e.g., my_data) for the calculation of t_o. To assign the mean values directly from the Jack Knife approach, use the following codes:

```
my_data$Linf<-as.numeric (growth_confidence[1])
my_data$K<-as.numeric (growth_confidence[2])
```

User can directly assign the derived growth parameters (e.g., L_∞ = 13.95 cm and K = 1.71 yr^{-1}) in the LFQ file using the following codes:

```
my_data$Linf<-13.95
my_data$K<-1.71
```

Apart from these above-mentioned parameters, the empirical calculation of t_{max} requires t_o. If t_o has already been estimated using methods mentioned above in '**2.2. Estimation of gestation period or hatching time (t_o)**', there is no need to save tzero again. However, user can always refer published literature and also independently specify the t_o value (e.g., -0.0011 yr) directly as follows:

```
tzero=-0.0011
```

Similarly, the rearranged VBGF equation-based estimation of t_{max} requires maximum length (L_{max}). The user can specify any published or observed value for t_{max} (e.g., 13.15 cm):

```
Lmax= 13.15
```

The natural mortality rate (M)-based equation for t_{max} requires M. The user can specify any published or observed value for M (e.g., 2.75 yr^{-1}) as follows:

```
M=2.75
```

TropFishR provides a default estimate on longevity as an inbuilt output of ELEFAN_SA and ELEFAN_GA. The following codes can be used to extract t_{max} after performing ELEFAN_SA or ELEFAN_GA:

```
tmax<-fitGA$agemax
tmax<-fitSA$agemax
```

> ***Note:*** *The above code will give the t_{max} value but round the value to the nearest integer. For the precise estimation of t_{max} use the empirical equation or the rearranged VBGF equation.*

2.3.2. Empirical equation for t_{max}

In the absence of direct observation data on the age of the largest individual, t_{max} can be indirectly estimated using the generalized equation suggested by Pauly (1983), as shown below:

$$t_{max} = \frac{3}{K} + t_0$$

Basically, it is the same as the rearranged VBGF equation for t_{max}, with an assumption that fish survive in the wild till attain a L_{max} which is 95% of the L_∞, i.e., L_{max}/L_∞ = 0.95.

Use the following code to calculate t_{max}, though the result may not be very precise.

```
tmax<- (3/my_data$K)+ tzero
```

Alternatively, the parameters (e.g., K = 1.71 yr^{-1}, and t_0 = -0.0011 yr) can also be specified directly in the code to obtain t_{max}, as given below:

```
tmax<- (3/1.71)-0.0011
```

2.3.3. Rearranged VBGF equation for t_{max}

The t_{max} can be precisely back-calculated using the exact L_{max}/L_∞ ratio in the rearranged VBGF equation (von Bertalanffy, 1938), as shown below:

$$t_{max} = \frac{-\ln\left(1 - \frac{L_{max}}{L_\infty}\right)}{K} + t_0$$

Calculate t_{max} using the following code:

```
tmax<- (-log(1-(Lmax/my_data$Linf))/my_data$K)+ tzero
```

Alternatively, these parameters (e.g., L_{max} = 13.15 cm, K = 1.71 yr^{-1}, and t_0 = -0.0011 yr) can also be specified directly in the code to obtain t_{max}, as given below:

```
tmax<- (-log(1-(13.15/13.95))/1.71)-0.0011
```

2.3.4. Natural mortality rate (M)-based equation for t_{max}

The method used inverse relationship between natural mortality rate (M) and longevity (t_{max}) by assuming a certain probability of fish living to maximum age (t_{max}) under a given level of total mortality. The t_{max} can be estimated from the survival (p) at t_{max} information and M using the generalized formula of Quinn and Deriso (1999):

$$t_{max} = \frac{-Ln\ (p)}{M}$$

The value of p is highly subjective and has typically been assumed to be between 1% and 5% (Hewitt and Hoenig, 2005). As a rule of thumb, a 1% survival rate (p=0.01) at t_{max} can be assumed, which will produce the following equation:

$$t_{max} = \frac{-Ln\ (p)}{M} = \frac{-Ln\ (0.01)}{M} = \frac{4.61}{M}$$

Calculate t_{max} using the following code: `tmax<- (4.61/M)`

Alternatively, the natural mortality parameter (e.g., M = 2.75 yr^{-1}) can also be specified directly in the code to get t_{max}, as given below: `tmax<- (4.61/2.75)`

2.3.5. Estimation of mean and confidence intervals of t_{max}

To generate t_{max} values, along with their mean and confidence intervals, use the following codes:

```
t_max_function = function(x, growthparamdata){
Linf = x[1]
K = x[2]
tzero = x[6]
return ((-log(1-(Lmax/Linf))/K)+ tzero)}
t_maxs<-apply(growthparamdata,1, t_max_function)
t_maxs_mean <- apply(matrix(t_maxs), MARGIN = 2, FUN = mean)
t_maxs_conf <- apply(matrix(t_maxs), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
print(c(Mean= t_maxs_mean, Lower_95_CI = t_maxs_conf[1,1], Upper_95_CI = t_maxs_conf[2,1]))
```

Add the t_{max} values as a column to 'growthparamdata' using the following code:

```
growthparamdata$t_max <- t_maxs
```

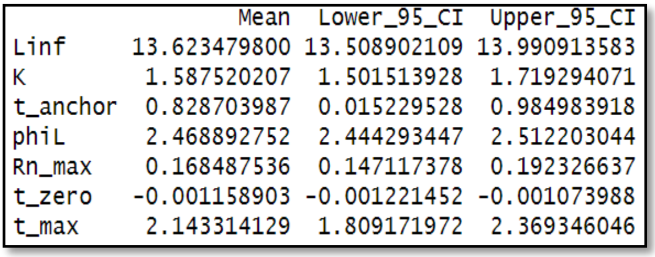
> ***Note:*** *The highlighted portion of the code in '**return (.....)**' is using the rearranged VBGF equation for t_{max}. The user can replace the portion of the code depending on the choice of method.*

2.4. Compilation, plotting and saving of growth parameters

2.4.1. Compilation and exporting the mean and confidence intervals of growth parameters as CSV

To summarize the growth parameter data derived so far, use the following code:

```
growthparam_means <- apply(as.matrix(growthparamdata), MARGIN = 2, FUN = mean)
growthparam_conf <- apply(as.matrix(growthparamdata), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
growthparam_confidence<-t(as.data.frame(rbind(Mean=growthparam_means,
Lower_95_CI =growthparam_conf[1,], Upper_95_CI =growthparam_conf[2,])))
growthparam_confidence
```



```
                 Mean  Lower_95_CI  Upper_95_CI
Linf      13.623479800 13.508902109 13.990913583
K          1.587520207  1.501513928  1.719294071
t_anchor   0.828703987  0.015229528  0.984983918
phiL       2.468892752  2.444293447  2.512203044
Rn_max     0.168487536  0.147117378  0.192326637
t_zero    -0.001158903 -0.001221452 -0.001073988
t_max      2.143314129  1.809171972  2.369346046
```

To export the mean and confidence intervals of growth parameters as CSV

```
write.csv(growthparam_confidence,
"C:\\Users\\Dell\\Desktop\\growth_parameters_confidence_intervals.csv",
row.names = TRUE)
```

> ***Note:*** *To determine the path where the CSV file will be saved, right-click on any file in the required location, select '**Properties**', and copy the file location under the '**General**' tab (e.g., C:\Users\Dell\Desktop). Replace the bold portion of the code with this copied file location, and then add the file name with its extension (e.g., \growth_parameters_confidence_intervals.csv). Make sure to use double backslashes (\\) or a single forward slash (/) between each segment of the*

path and enclose the entire path in quotation marks ("…/…./…./….") or ("…\\….\\….\\….").

2.4.2. Plotting confidence intervals of the growth parameters

To plot the growth parameters' confidence intervals, split the plotting screen into a single row with five columns. This setup will automatically plot all five parameters sequentially. To split the column use the following code: `par(mfrow = c(1,5))`

#To create a box plot for L_∞, use the following R code:

```
boxplot(growthparamdata$Linf, main= expression("L"[infinity]), col='magenta')
```

#To create a box plot for K, use the following R code:

```
boxplot(growthparamdata$K, main= "K", col='steelblue')
```

#To create a box plot for φ', use the following R code:

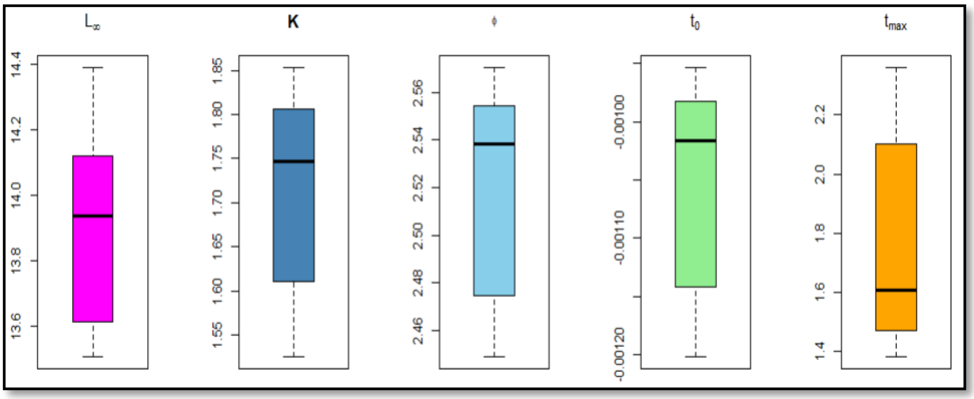
```
boxplot(growthparamdata$phiL, main= expression(""[phi]), col='skyblue')
```

#To create a box plot for t_0, use the following R code:

```
boxplot(growthparamdata$t_zero, main= expression("t"[0]), col='lightgreen')
```

#To create a box plot for t_{max}, use the following R code:

```
boxplot(growthparamdata$t_max, main= expression("t"[max]), col=' orange')
```



Note: *Switch off the plot screen split using the following code:* **`par(mfrow = c(1,1))`**

2.4.3. Saving essential growth parameters to the LFQ file

It is necessary to assign (save) the important growth parameters derived so far (L_∞ and K) in the LFQ file (e.g., my_data) for subsequent analysis, such as the length-converted catch curve analysis, length-based cohort analysis, and yield per recruit analysis.

To assign the growth parameters (L_∞ and K) derived from the ELEFAN_SA method, use the following codes:

```
my_data$Linf<-as.numeric (fitSA$par$Linf)
my_data$K<-as.numeric (fitSA$par$K)
```

To assign the growth parameters (L_∞ and K) derived from the ELEFAN_GA method, use the following codes:

```
my_data$Linf<-as.numeric (fitGA$par$Linf)
my_data$K<-as.numeric (fitGA$par$K)
```

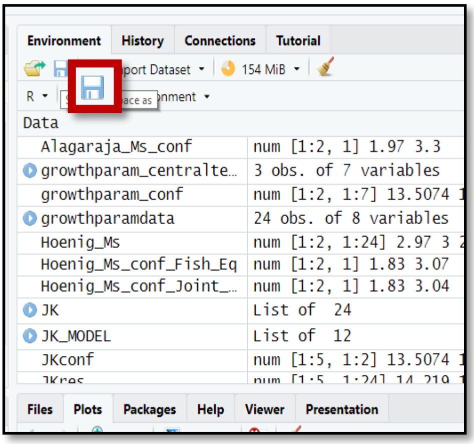
To assign the growth parameters (L_∞ and K) from the Jack Knife resampling after ELEFAN_SA or ELEFAN_GA, use the following codes:

```
my_data$Linf<- as.numeric(growth_param_confidence[1,1])
my_data$K<- as.numeric(growth_param_confidence[2,1])
```

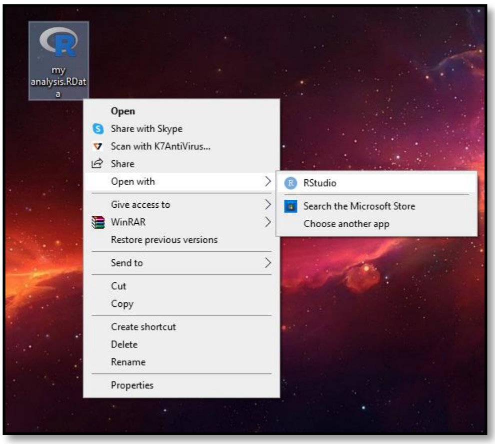
Alternatively, the growth parameters (e.g., L_∞ = 13.95 cm, and K = 1.71 yr^{-1}) can also be assigned directly using the following codes:

```
my_data$Linf<-13.95
my_data$K<-1.71
```

Optional step: If the user needs to close the ongoing session, the work progress can be saved by clicking the ' **save workspace as'** tab in the **'Environment'** panel (located on the top right side). Give the workspace a name (e.g., **'my_analysis'**). In a new session, the user can open the **'my_analysis'** file in RStudio by right-clicking the file and selecting the **'Open with'** RStudio option. Reload the TropFishR library each time a new session starts, using the code: library(TropFishR)



```
Data
  Alagaraja_Ms_conf          num [1:2, 1] 1.97 3.3
  growthparam_centralte…     3 obs. of 7 variables
  growthparam_conf           num [1:2, 1:7] 13.5074 1
  growthparamdata            24 obs. of 8 variables
  Hoenig_Ms                  num [1:2, 1:24] 2.97 3 2
  Hoenig_Ms_conf_Fish_Eq     num [1:2, 1] 1.83 3.07
  Hoenig_Ms_conf_Joint_…     num [1:2, 1] 1.83 3.04
  JK                         List of  24
  JK_MODEL                   List of  12
  JKconf                     num [1:5, 1:2] 13.5074 1
```



2.5. Estimation of natural mortality rate (M)

Introduction

Natural Mortality represents all forms of mortality unrelated to fishing activities such as predation, starvation, disease, senescence. The natural mortality rate (M) is one of the most influential parameters in fisheries stock assessment and management because it directly affects estimates of stock productivity and biological reference points (Hilborn and Ovando, 2014). In tropical fisheries, natural mortality rate (M) is one of the most challenging parameters to estimate due to the lack of unbiased tagging data or age-composition data in the absence of fishing (Maunder et al., 2023). Several methods for estimating natural mortality have been well reviewed by Kenchington (2014), Then et al. (2015) and Maunder et al. (2023). Most of these methods are based on life history theory and rely on empirical relationships between L_∞, K, and t_{max} in various combinations, and few also use additional parameters such as temperature (e.g., Pauly's method) or age at maturity tm_{50} (e.g., Rikhter-Efanov method) to derive M. A list of commonly used methods is given below.

Methods	Equation	Reference
Hoenig Fish equation	$M = 4.31 \times {t_{max}}^{-1.01}$	Hoenig, 1983
Hoenig Joint equation	$M = 4.22 \times {t_{max}}^{-0.982}$	Hoenig, 1983
t_{max}-based method	$M = \frac{-Ln\ (p)}{t_{max}}$	Quinn and Deriso, 1999
Alverson and Carney Method Zhang and Megrey Method	$M = \frac{3K}{(exp^{K \times 0.38 \times t_{max}}) - 1}$ $M = \frac{3K}{(exp^{K \times 0.393 \times t_{max}}) - 1}$	Alverson and Carney Method, 1975 Zhang and Megrey, 2006
Zhang and Megrey Demersal fish method	$M = \frac{3K}{(exp^{K \times 0.440 \times t_{max}}) - 1}$	Zhang and Megrey, 2006
Zhang and Megrey pelagic fish method	$M = \frac{3K}{(exp^{K \times 0.302 \times t_{max}}) - 1}$	Zhang and Megrey, 2006
Then t_{max}-based method	$M = 4.899 \times {t_{max}}^{-0.916}$	Then et al., 2015
Then growth-based method	$M = 4.118 \times K^{0.73} \times {L_{\infty}}^{-0.33}$	Then et al., 2015
Then K-based method	$M = 1.692 \times K$	Then et al., 2015
Jensen K-based	$M = 1.5 \times K\ or\ 1.6 \times K$	Jensen, 1996
Jensen tm_{50}-based	$M = \frac{1.65}{tm_{50}}$	Jensen, 1996
Hamel K-based method	$M = 1.753 \times K$	Hamel, 2015
Gunderson revised method	$M = 1.817 \times GSI$	Hamel, 2015
Gunderson and Dygert method	$M = 0.03 + 1.68 \times GSI$	Gunderson and Dygert, 1988
Gunderson method	$M = 1.79 \times GSI$	Gunderson, 1997
Rikhter-Efanov method	$M = \frac{1.521}{{tm_{50}}^{0.720}} - 0.155$	Rikhter and Efanov, 1976
Roff method	$M = \frac{3K}{(exp^{tm_{50} \times K}) - 1}$	Roff, 1984

Pauly's method	$log_{10}(M) = -0.0066 - 0.279 \times log_{10}(L_\infty) + 0.6543 \times log_{10}(K) + 0.4634 \times log_{10}(T)$	Pauly, 1980
Gislason's First Estimator	$M = 1.73 \times L^{-1.61} \times {L_\infty}^{1.44} \times K$	Gislason et al. (2010)
Gislason's Second Estimator	$M = K \times \frac{L_\infty}{L}$	Charnov et al. (2012)
Peterson - Wroblewski method	$M = 1.92 \times {W_{dry}}^{-0.25}$	Peterson and Wroblewski, 1984
Lorenzen method	$M = 3\ {W_{wet}}^{-0.288}$	Lorenzen, 1996

Natural mortality rate (M): R Implementation

2.5.1. Requirements for the estimation of M

LFQ file

A length-frequency data file (LFQ) is required for the analysis (e.g., my_data). Refer to the previously mentioned steps in ***'2.1.2 and 2.1.5. Creating a length frequency file (LFQ) on R'*** section to newly create a LFQ file if not created earlier.

Essential parameters

To derive M using any particular method or a combination of methods, it is necessary to have all the essential input parameters saved in the LFQ file (e.g., my_data). For example, to estimate M using some popular methods, such as Pauly's method, Hoenig's method, Then's methods, and Rikhter-Efanov method, input parameters such as L∞, K, t_{max}, tm_{50} and habitat temperature are required which must be supplied to the LFQ file (e.g., my_data) as follows:

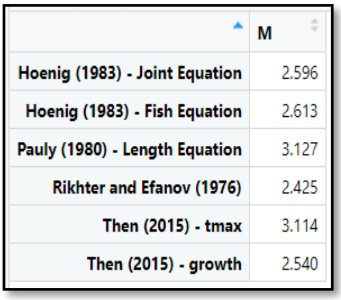
```
my_data$Linf<-13.95
my_data$K<-1.71
my_data$tmax<-1.67
my_data$tm50<-0.5
```

To estimate M, the user can use the following inbuilt code in TropFishR.

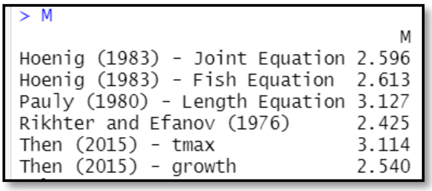
```
M <- M_empirical(Linf = my_data$Linf, K_l = my_data$K, temp = 28, tmax =
my_data$tmax, tm50= my_data$tm50, method = c("Pauly_Linf","Hoenig",
"Then_growth", "Then_tmax", "RikhterEfanov"))
M
```

Alternatively, if the requisite input parameters for the analysis such as L∞, K, t_{max}, tm_{50} are not saved in the LFQ file (e.g., my_data), they can be directly supplied in the code:

```
M <- M_empirical(Linf = 13.95, K_l = 1.71, temp = 28, tmax = 1.67, tm50= 0.50, method
= c("Pauly_Linf","Hoenig", "Then_growth", "Then_tmax", "RikhterEfanov"))
M
```



	M
Hoenig (1983) - Joint Equation	2.596
Hoenig (1983) - Fish Equation	2.613
Pauly (1980) - Length Equation	3.127
Rikhter and Efanov (1976)	2.425
Then (2015) - tmax	3.114
Then (2015) - growth	2.540



```
> M
                                   M
Hoenig (1983) - Joint Equation 2.596
Hoenig (1983) - Fish Equation  2.613
Pauly (1980) - Length Equation 3.127
Rikhter and Efanov (1976)      2.425
Then (2015) - tmax             3.114
Then (2015) - growth           2.540
```

Note: *To perform the analysis using a specific method or a combination of methods, ensure that the required input parameters are assigned or saved in the LFQ file* ***(my_data)*** *or supplied directly. For example, Pauly's method requires 'temperature,' and the Rikhter-Efanov method requires tm_{50} as additional input parameters. If these parameters are not provided, errors will occur during the analysis. Remove any unwanted methods or those lacking the necessary input parameters by deleting them from* ***the methods = c("........")*** *section of the code. The 12 different methods that can be employed using TropFishR are: "AlversonCarney", "Gislason" (size dependent mortality estimates), "GundersonDygert", "Hoenig", "Lorenzen", "Pauly_Linf", "Pauly_Winf", "PetersonWroblewski", "RikhterEfanov", "Roff", "Then_growth", or "Then_tmax".*

2.5.2. Estimation of mean and confidence intervals of M

Since M is calculated as a dependent parameter using the growth parameters L_∞, K, and t_{max}, multiple M values can be generated by applying Jack Knife samples of these growth parameters. These M values can then be used to calculate the mean and confidence intervals for M. The procedures for some of the commonly used methods are given below:

The mean and confidence intervals of M can be derived from some popular M estimation methods, as follows:

Pauly's method for M

To generate M values using the growth parameter–based equation recommended by Pauly (1980), along with their mean and confidence intervals, use the following codes:

First, add temperature data (e.g., 28°C) to the previously created ***growthparamdata*** data frame using the following code:

```
growthparamdata$temp<-28
Pauly_M_function = function(x, growthparamdata) {
Linf = x[1]
K_l = x[2]
temp= x[8]
return (M_empirical (Linf = Linf, K_l = K_l, temp = temp, method = c("Pauly_Linf")))}
Pauly_Ms <-apply(growthparamdata,1, Pauly_M_function)
Pauly_Ms_mean <- apply(matrix(Pauly_Ms), MARGIN = 2, FUN = mean)
Pauly_Ms_conf <- apply(matrix(Pauly_Ms), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
```

```
print(c(Mean=Pauly_Ms_mean, Lower_95_CI =Pauly_Ms_conf[1], Upper_95_CI = Pauly_Ms_conf[2]))
```

Then_growth method for M

To generate M values using the growth parameter–based equation recommended by Then et al. (2015), along with their mean and confidence intervals, use the below codes:

```
Then_growth_M_function = function(x, growthparamdata) {
Linf = x[1]
K_l = x[2]
return (M_empirical (Linf = Linf, K_l = K_l, method = c("Then_growth")))}
Then_growth_Ms <-apply(growthparamdata,1, Then_growth_M_function)
Then_growth_Ms_mean <- apply(matrix(Then_growth_Ms), MARGIN = 2, FUN = mean)
Then_growth_Ms_conf <- apply(matrix(Then_growth_Ms), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
print(c(Mean=Then_growth_Ms_mean, Lower_95_CI = Then_growth_Ms_conf[1], Upper_95_CI = Then_growth_Ms_conf[2]))
```

Natural mortality rate (M) can be estimated from the survival at maximum age (t_{max}) information using the generalized formula of Quinn and Deriso (1999):

$$M = \frac{-Ln\,(p)}{t_{max}}$$

Where p represents the proportion of fish in the stock that survive until they reach the maximum age (t_{max}). The value of p is highly subjective and has typically been assumed to be between 1% and 5% (Hewitt and Hoenig, 2005). As a rule of thumb, a 5% survival rate (p=0.05) at t_{max} is commonly used, but it has been found to overestimate M.

$$M = \frac{-Ln\,(p)}{t_{max}} = \frac{-Ln\,(0.05)}{t_{max}} = \frac{3}{t_{max}} \text{ (May not be very appropriate)}$$

A survival rate of 1.5% (p=0.015) at t_{max} has been recommended by Hoenig (1983) and Hewitt and Hoenig (2005) as a more appropriate value for estimating M.

$$M = \frac{-Ln\,(p)}{t_{max}} = \frac{-Ln\,(0.015)}{t_{max}} = \frac{4.2}{t_{max}} \text{ (More appropriate)}$$

For simplicity, a survival rate of 1% (p=0.01) at t_{max} has been recommended by Alagaraja (1984) for estimating M.

$$M = \frac{-Ln\,(p)}{t_{max}} = \frac{-Ln\,(0.01)}{t_{max}} = \frac{4.61}{t_{max}} \text{ (Appropriate)}$$

Alagaraja's method for M

To generate M values using the t_{max}–based equation recommended by Alagaraja (1984), along with their mean and confidence intervals, use the following codes:

```
M<-log(0.01)/-tmax
Alagaraja_M_function = function(x, growthparamdata) {
```

```
tmax = x[7]
return (log(0.01)/-tmax)}
Alagaraja_Ms <-apply(growthparamdata,1, Alagaraja_M_function)
Alagaraja_Ms_mean <- apply(matrix(Alagaraja_Ms), MARGIN = 2, FUN = mean)
Alagaraja_Ms_conf <- apply(matrix(Alagaraja_Ms), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
print(c(Mean=Alagaraja_Ms_mean, Lower_95_CI = Alagaraja_Ms_conf [1], Upper_95_CI = Alagaraja_Ms_conf [2]))
```

Then_t_{max} method for M

To generate M values using the t_{max}–based equation recommended by Then et al. (2015), along with their mean and confidence intervals, use the following codes:

```
Then_tmax_M_function = function(x, growthparamdata) {
tmax= x[7]
return (M_empirical (tmax = tmax, method = c("Then_tmax")))}
Then_tmax_Ms <-apply(growthparamdata,1, Then_tmax_M_function)
Then_tmax_Ms_mean <- apply(matrix(Then_tmax_Ms), MARGIN = 2, FUN = mean)
Then_tmax_Ms_conf <- apply(matrix(Then_tmax_Ms), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
print(c(Mean= Then_tmax_Ms_mean, Lower_95_CI = Then_tmax_Ms_conf[1], Upper_95_CI = Then_tmax_Ms_conf[2]))
```

Hoenig's method for M

To generate M values using the t_{max}–based equation recommended by Hoenig et al. (1983), along with their mean and confidence intervals, use the following codes:

```
Hoenig_M_function = function(x, growthparamdata) {
tmax = x[7]
return (M_empirical (tmax = tmax, method = c("Hoenig")))}
Hoenig_Ms <-apply(growthparamdata,1, Hoenig_M_function)
Hoenig_Ms_mean_Joint_Eq <- apply(matrix(Hoenig_Ms[1,]), MARGIN = 2, FUN = mean)
Hoenig_Ms_conf_Joint_Eq <- apply(matrix(Hoenig_Ms[1,]), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
Hoenig_Ms_mean_Fish_Eq <- apply(matrix(Hoenig_Ms[2,]), MARGIN = 2, FUN = mean)
Hoenig_Ms_conf_Fish_Eq <- apply(matrix(Hoenig_Ms[2,]), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
print(c(Joint_Eq_Mean=Hoenig_Ms_mean_Joint_Eq, Joint_Eq_Lower_95_CI= Hoenig_Ms_conf_Joint_Eq [1], Joint_Eq_Lower_95_CI = Hoenig_Ms_conf_Joint_Eq [2]))
print(c(Fish_Eq_Mean=Hoenig_Ms_mean_Fish_Eq, Fish_Eq_Lower_95_CI = Hoenig_Ms_conf_Fish_Eq [1], Fish_Eq_Upper_95_CI = Hoenig_Ms_conf_Fish_Eq [2]))
```

2.6. Compilation, plotting and saving of natural mortality rate

2.6.1. Compilation and exporting the mean and confidence intervals of M as CSV

To save the individual M values from different methods as a data frame, use the following code:

```
mortalityparamdata <- as.data.frame (cbind(Pauly=Pauly_Ms, Then_growth= Then_growth_Ms, Then_tmax=Then_tmax_Ms, Alagaraja=Alagaraja_Ms, Hoenig_Joint=Hoenig_Ms[1, ], Hoenig_Fish=Hoenig_Ms[2, ]))
```

```
mortalityparamdata
```

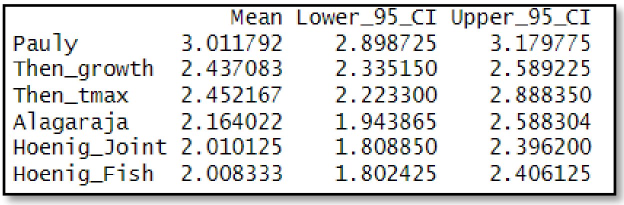
To summarize the mean and confidence intervals of M, use the following codes:

```
M_means <- apply(as.matrix(mortalityparamdata), MARGIN = 2, FUN = mean)
```

```
M_conf <- apply(as.matrix(mortalityparamdata), MARGIN = 2, FUN = function(x) quantile(x, probs=c(0.025,0.975)))
```

```
M_confidence <- t(as.data.frame (rbind (Mean=M_means, Lower_95_CI =M_conf[1,], Upper_95_CI =M_conf[2,])))
```

```
M_confidence
```



```
                 Mean Lower_95_CI Upper_95_CI
Pauly        3.011792    2.898725    3.179775
Then_growth  2.437083    2.335150    2.589225
Then_tmax    2.452167    2.223300    2.888350
Alagaraja    2.164022    1.943865    2.588304
Hoenig_Joint 2.010125    1.808850    2.396200
Hoenig_Fish  2.008333    1.802425    2.406125
```

To export the mean and confidence intervals of M as CSV, use the following code:

```
write.csv(M_confidence, "C:\\Users\\Dell\\Desktop\\M_confidence_intervals.csv", row.names = TRUE)
```

Note: *To determine the path where the CSV file will be saved, right-click on any file in the required location, select* ***'Properties'****, and copy the file location under the* ***'General'*** *tab (e.g., C:\Users\Dell\Desktop). Replace the bold portion of the code with this copied file location, and then add the file name with its extension (e.g., \\M_confidence_intervals.csv). Use double backslashes (\\) or a single forward slash (/) between each segment of the path and enclose the entire path in quotation marks ("…/…./…./….") or ("…\\….\\….\\….").*

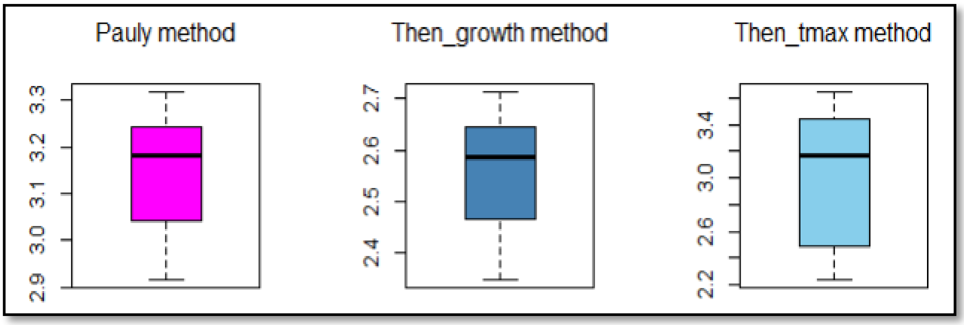
2.6.2. Plotting confidence intervals of M

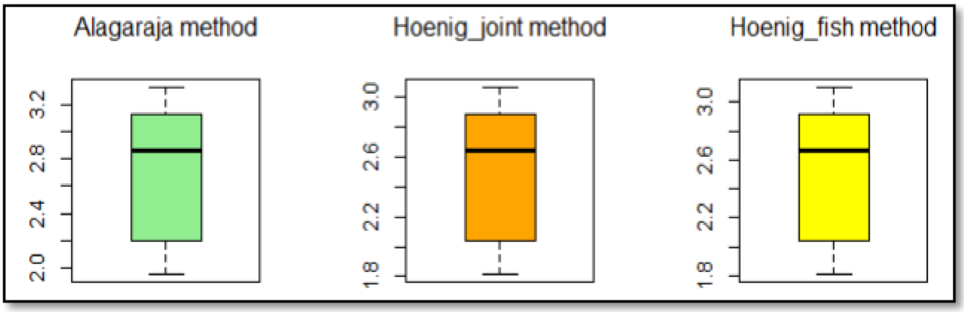
To plot the M confidence intervals, split the plotting screen into two rows, each having three split columns to automatically plot the M values derived from six different methods sequentially, as follows:

```
par(mfrow = c(2,3))
```

#To create a box plot for M from the Pauly's Method, use the following R code:

```
boxplot(mortalityparamdata $ Pauly, main= expression("Pauly method"), col='magenta')
#To create a box plot for M from the Then_growth's method, use the following R code:
boxplot(mortalityparamdata $ Then_growth, main= expression("Then_growth method"), col='steelblue')
#To create a box plot for M from the Then_tmax's method, use the following R code:
boxplot(mortalityparamdata $ Then_tmax, main= expression("Then_tmax method"), col='skyblue')
#To create a box plot for M from the Alagaraja's method, use the following R code:
boxplot(mortalityparamdata$Alagaraja, main= expression("Alagaraja method"), col='lightgreen')
#To create a box plot for M from the Hoenig_Joint method, use the following R code:
boxplot(mortalityparamdata $ Hoenig_Joint, main= expression("Hoenig_joint method"), col ='orange')
#To create a box plot for M from the Hoenig_Fish method, use the following R code:
boxplot(mortalityparamdata $ Hoenig_Fish, main= expression("Hoenig_fish method"), col ='yellow')
```





Note: *Switch off the plot screen split using the following code:* par(mfrow = c(**1**,**1**))

2.6.3. Saving the selected M to the LFQ file

It is necessary to assign (save) the M derived from the selected method in the LFQ file (e.g., **my_data**) for subsequent analysis, such as the length-converted catch curve analysis, length-based cohort analysis, and yield per recruit analysis.

To assign the M derived from the Pauly's Method, use the following code:

my_data$M<- as.numeric(M_confidence [1])

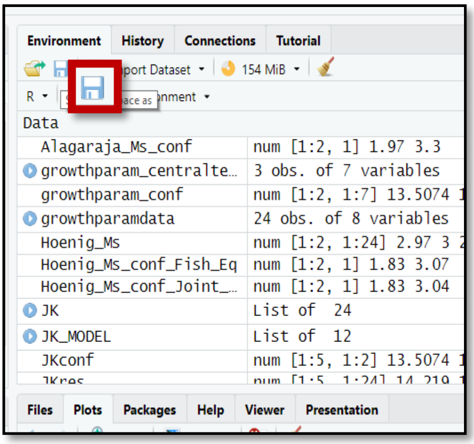
To assign the M derived from the Then t_{max}-based Method, use the following code:

my_data$M<- as.numeric(M_confidence [3])

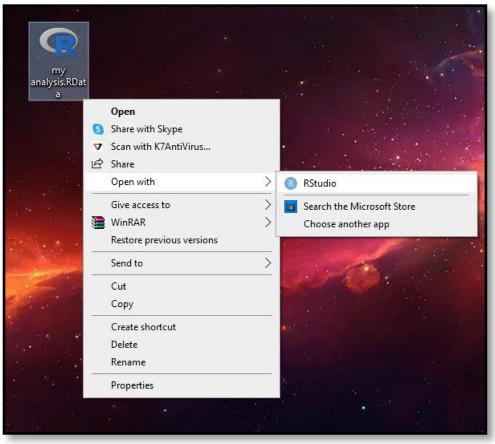
Alternatively, the selected M (e.g., 2.75 yr^{-1}) can also be assigned to the LFQ directly using the following code:

my_data$M<-**2.75**

Optional step: If the user needs to close the ongoing session, the work progress can be saved by clicking the '💾 **save workspace as'** tab in the **'Environment'** panel (located on the top right side). Give the workspace a name (e.g., **'my_analysis'**). In a new session, the user can open the **'my_analysis'** file in RStudio by right-clicking the file and selecting the **'Open with'** RStudio option. Reload the TropFishR library each time a new session starts, using the code: library(TropFishR)



```
Alagaraja_Ms_conf          num [1:2, 1] 1.97 3.3
growthparam_centralte…     3 obs. of 7 variables
growthparam_conf           num [1:2, 1:7] 13.5074 1
growthparamdata            24 obs. of 8 variables
Hoenig_Ms                  num [1:2, 1:24] 2.97 3 2
Hoenig_Ms_conf_Fish_Eq     num [1:2, 1] 1.83 3.07
Hoenig_Ms_conf_Joint_…     num [1:2, 1] 1.83 3.04
JK                         List of  24
JK_MODEL                   List of  12
JKconf                     num [1:5, 1:2] 13.5074 1
JKres                      num [1:5, 1:24] 14.219 1
```



2.7. Estimation of exploitation parameters using the length converted catch curve analysis

Introduction

The length converted catch curve analysis is frequently employed in tropical fisheries, particularly in data-limited scenarios where age-structured data is unavailable, to estimate the total mortality rate (Z) experienced by a fish population over its life cycle. The von Bertalanffy Growth Function (VBGF), derived from the analysis of length-frequency data, is used to convert lengths to corresponding ages. Although these length-converted age groups may represent different cohorts sampled at a single point in time, they are assumed in the analysis to represent the entire lifespan of a single cohort, known as a pseudocohort. These converted age groups and their associated catches are then used to build an exponential decay model for the pseudocohort, which quantifies the total mortality rate (Z). It assumes that individuals in every size class (or age group) experience the same Z (Z = F + M), which may not be very realistic. Typically, smaller and younger fish in a population experience higher natural mortality (M) because of predation and other natural factors but lower fishing mortality (F) because they are not fully recruited to the fishing grounds and often escape capture due to their small size. As the fish grows, M decreases while F increases because of recruitment to the fishing grounds and gear selectivity. This inverse change in M and F at different life stages likely keeps Z constant across different length classes. Therefore, in the absence of length-specific Z values, it is often assumed to be constant across length classes (or age groups) (Sparre and Venema, 1998).

An unbiased catch from a non-selective gear can be considered a representative sample of the entire fish population, proportionally reflecting the number of individuals in each size class (or age group). The number of fish in the smallest size class (youngest age group) is higher, which gradually decreases because of mortality as the fish grow larger. The number of fish in any age group (N_{t+1}) can be calculated from the number of fish in the preceding age group (N_t) using the following formula:

$$N_{t+1} = N_t \times exp^{-Zt}$$

Since the catch is proportional to and representative of the total population, the rate of decrease in catch numbers by length classes (or age groups) can also reflect the actual population decline rate. Therefore, when the catch number is plotted against age (t), it produces a curvilinear plot similar to the population decrease curve, with a negative slope that declines exponentially at the rate of Z. To linearize the equation, the catch numbers are log-transformed, resulting in the following equation:

$$Ln(C) = a - \ Z \times t$$

In temperate fisheries, where fish are directly aged, the standard **age-based catch curve** of Ricker (1958 & 1975) is used in which the natural logarithm of catch numbers in each age group is directly plotted and regressed against age. However, when fish are aged indirectly (in tropical fisheries) by converting length to age, it causes a **"piling-up effect"**. As older fish grow more slowly than younger fish, the growth in size is not linear. Therefore, among larger fish, any length interval (e.g., a 5 cm length class) will encompass more age groups than among smaller fish, leading to the accumulation of more age groups as fish approach their asymptotic length (L_∞). This piling-up effect during the length-to-age conversion is compensated by dividing the catch by the time interval (Δt) the fish spends growing through the length class (Pauly, 1983). The length-based form of this approach is called the **length-converted catch curve** (Pauly, 1983), which is expressed as follows:

$$Ln\frac{C}{\Delta t} = a - Z \times t$$

Where Δt is the time required to grow through the length class (from L_1 to L_2). It is calculated using the following equation:

$$\Delta t = t(L_2) - t(L_1) = \frac{1}{K} \times ln\left[\frac{L_\infty - L_1}{L_\infty - L_2}\right]$$

The linearized catch curve equation is then solved to estimate the total mortality rate (Z) for the population. The fishing mortality (F) and exploitation rate (E) can be derived from Z using the following relationships:

$$F = Z - M \ and \ E = \frac{F}{Z}$$

However, since the catch only represents the exploited portion of the population, smaller fish in the lower age groups are not proportionately represented in the catch. This creates an illusion that these smaller individuals are absent from the population, when in fact they have not been caught by the gear because they are not fully recruited. Therefore, the catch numbers of smaller fish on the ascending arm (left-hand side of the peak catch number) are not used in the catch curve analysis, as they inaccurately represent the population. Similarly, the oldest age group, which contains the largest fish, accumulates survivors from previous cohorts. This mixed age group is referred to as the plus group and is assigned a minimum age, as it becomes difficult to distinguish the fish's age by size once they have reached their growth limit. Moreover, the oldest and largest fish may escape gear because of their scarce presence. Therefore, the catch numbers in the last length class (oldest age group) are also ignored due to their improper representation of the population. The peak catch number usually corresponds to the length class (or age group) at and above which fish are fully recruited and selected by the gear. The catch numbers of larger fish (or older fish) on the descending arm of the catch curve (right-hand side of the peak catch number) are used to represent the population for the catch curve analysis. Every point on the descending arm of the catch curve experiences a constant Z, and this Z is used to reconstruct (extrapolate) the theoretical catch numbers for smaller and younger fish on the ascending arm of the catch curve, assuming they had been fully selected by the fishing gear. The ratio of the actual recorded catch number (C_t) to the reconstructed catch number from the linearized catch equation ($\Delta t \times \exp\,(a\text{-}Z\times t)$) is calculated to derive the probability of capture (S_t) for the size classes (age groups).

$$S_t = \frac{C_t}{\Delta t \times exp^{(a-Z\times t)}}$$

The age (t) and the probability of capture (S_t) values are then regressed using a linearized logistic regression to derive the regression coefficients (a and b).

$$ln\left(\frac{1}{S_t} - 1\right) = a - b \times t$$

The above logistic regression equation can be rearranged as:

$$S_t = \frac{1}{1 + exp^{(a-b\times t)}} = \frac{exp^{-(a-b\times t)}}{1 + exp^{-(a-b\times t)}}$$

Where, t is the independent measurement variable (here, age of the fish), S_t is the dependent categorical variable (here, probability of capture, i.e., proportion of the fish caught compared to the total fish encountering the gear at a length); a & b are the intercept and slope of the equation, respectively. Since the probability of capture (S_t) falls in a narrow range from 0 to 1, it creates difficulty while fitting the regression. Therefore, to overcome the situation, the odds, i.e., S_t /(1- S_t), which is the ratio between probability of capture/probability of escape, are used for the regression. Finally, taking the natural log of the odds makes the variable more appropriate for regression analysis. The coefficients derived from the regression are subsequently used to estimate the age at which different levels of capture happen using the rearranged form of the above equation as follows:

$$t = \frac{\ln\left(\frac{1}{S_t} - 1\right) - a}{-\mathrm{b}}$$

For example, to derive the tc₂₅, i.e., the age (t) at which 25% capture (S_t = 0.25) happens, populate the values in the above equation as mentioned below:

$$\mathrm{tc}_{25} = \frac{\ln\left(\frac{1}{0.25} - 1\right) - a}{-\mathrm{b}} = \frac{\ln\left(\frac{0.75}{0.25}\right) - a}{-\mathrm{b}} = \frac{\ln(3) - a}{-\mathrm{b}}$$

To derive the tc_{50}, i.e., the age (t) at which 50% capture (S_t=0.50) happens, populate the values in the above equation as mentioned below:

$$\mathrm{tc}_{50} = \frac{\ln\left(\frac{1}{0.50} - 1\right) - a}{-\mathrm{b}} = \frac{\ln\left(\frac{0.50}{0.50}\right) - a}{-\mathrm{b}} = \frac{\ln(1) - a}{-\mathrm{b}} = \frac{-\mathrm{a}}{-\mathrm{b}}$$

To derive the tc_{75}, i.e., the age (t) at which 75% capture (S_t=0.75) happens, populate the values in the above equation as mentioned below:

$$\mathrm{tc}_{75} = \frac{\ln\left(\frac{1}{0.75} - 1\right) - a}{-\mathrm{b}} = \frac{\ln\left(\frac{0.25}{0.75}\right) - a}{-\mathrm{b}} = \frac{\ln(1/3) - a}{-\mathrm{b}}$$

To derive the tc_{95}, i.e., the age (t) at which 95% capture (S_t=0.95) happens, populate the values in the above equation as mentioned below:

$$\mathrm{tc}_{95} = \frac{\ln\left(\frac{1}{0.95} - 1\right) - a}{-\mathrm{b}} = \frac{\ln\left(\frac{0.05}{0.95}\right) - a}{-\mathrm{b}} = \frac{\ln(1/19) - a}{-\mathrm{b}}$$

The age and length at which different levels of capture happen can be summarized as follows:

Description	Age (tc)	Length (LC)
The age (tc_{25}) and length (LC_{25}) at which 25% of the fish encountering the gear are caught	$tc_{25} = \frac{\ln(3) - a}{-b}$	$LC_{25} = L_\infty \times \left(1 - exp^{-K(tc_{25} - t_0)}\right)$
The age (tc_{50}) and length (LC_{50}) at which 50% of the fish encountering the gear are caught	$tc_{50} = \frac{-a}{-b}$	$LC_{50} = L_\infty \times \left(1 - exp^{-K(tc_{50} - t_0)}\right)$
The age (tc_{75}) and length (LC_{75}) at which 75% of the fish encountering the gear are caught	$tc_{75} = \frac{\ln\left(\frac{1}{3}\right) - a}{-b}$	$LC_{75} = L_\infty \times \left(1 - exp^{-K(tc_{75} - t_0)}\right)$
The age (tc_{95}) and length (LC_{95}) at which 95% of the fish encountering the gear are caught	$tc_{95} = \frac{\ln\left(\frac{1}{19}\right) - a}{-b}$	$LC_{95} = L_\infty \times \left(1 - exp^{-K(tc_{95} - t_0)}\right)$

> ***Note:*** *The regression coefficient 'b' obtained from the length converted catch curve method output (i.e., linear_mod_sel) in TropFishR is usually expressed as negative b (i.e., -b) and therefore, the same can be directly used for the calculation of age at different level of captures.*

Catch curve analysis: R Implementation

2.7.1. Requirements for the length converted catch curve (CC) analysis

LFQ file

A length-frequency data file (LFQ) is required for the analysis (e.g., my_data). Refer to the previously mentioned steps in '2.1.2 and 2.1.5. Importing and creating a length frequency file (LFQ) on R' section to newly create a LFQ file if not created earlier.

Essential parameters

The length converted catch curve (CC) analysis requires growth parameters (L_∞, and K) and the natural mortality rate (M). If these essential parameters are not already assigned, use the following code to add them to the LFQ file (my_data):

```
my_data$Linf<-13.95
my_data $K<-1.71
my_data$M<-2.75
```

2.7.2. Catch curve analysis for a combination of years using multiyear mean catch data

If the aim of the analysis is to understand the average exploitation over multiple years, the user can perform a length-converted catch curve analysis by calculating the mean catch numbers by length-class for the desired combination of years. Use the following code to prepare first a catch vector and then a mean catch vector using the catch vector.

Preparing a catch vector

First, vectorize the catch per month to yearly catches using the following codes:

```
plus_group <- my_data$midLengths[max(which(my_data$midLengths < my_data
$Linf))]
catch_vec <- lfqModify(my_data, vectorise_catch = TRUE, plus_group = plus_group)
```

> ***Note:*** *The* ***vectorise_catch*** *function aggregates the monthly catches by length class for each year. Depending on the number of years in the LFQ data file, catch vectors are created (e.g., if the dataset contains information for 4 years, four yearly catch vectors are generated).*

Preparing a mean catch vector from the catch vector

To prepare a mean catch vector for **the first and second years,** use the following code:

```
mean_catch_vec<- catch_vec
mean_catch_vec$catch<-as.matrix(rowMeans(catch_vec$catch[,1:2]))
```

To prepare a mean catch vector for **the third and fourth years,** use the following code:

```
mean_catch_vec<- catch_vec
```

```
mean_catch_vec$catch<-as.matrix(rowMeans(catch_vec$catch[,3:4]))
```

To prepare a mean catch vector for **the second, third and fourth years,** use the following code:

```
mean_catch_vec<- catch_vec
mean_catch_vec$catch<-as.matrix(rowMeans(catch_vec$catch[,2:4]))
```

To prepare a mean catch vector for **the first, second, third and fourth years,** use the following code:

```
mean_catch_vec<- catch_vec
mean_catch_vec$catch<-as.matrix(rowMeans(catch_vec$catch[,1:4]))
```

To prepare a mean catch vector for all available years, use the following code:

```
mean_catch_vec<- catch_vec
mean_catch_vec$catch<-as.matrix(rowMeans(catch_vec$catch))
```

Finally, to perform the length-converted catch curve analysis for a combination of years, use the following code:

```
CC <- catchCurve(mean_catch_vec, catch_columns = 1,  calc_ogive = TRUE)
```

> ***Note:*** *The highlighted numbers inside the 'catch_vec$catch[,**1:4**]' portion of the code can be changed to include or exclude year(s) from the analysis. For example, to include only the last two years (3rd and 4th year) change the highlighted numbers inside the 'catch_vec$catch[,**3:4**]'. Similarly, if there are 5 years of data, the last three years (3rd, 4th and 5th year) of data can be included for the analysis by changing it to 'catch_vec$catch[,**3:5**]'. Creating **mean_catch_vec** is not a compulsory step for performing catch curve analysis and subsequent cohort analysis. These analyses can be conducted by specifying the required combination of years (serial numbers of the years) in the **catch_columns = c(.. , .. , ..)** section of the respective codes. When multiple combinations of years are used for catch curve analysis without averaging the length class-wise mean catch numbers over those years, the exploitation parameters (F, E, and LC_{50}) will be the same as those obtained using the length class-wise mean catch numbers over the multiple years. However, the population estimates (e.g., yield, biomass, spawning stock biomass, recruitment, and stock size) from subsequent cohort analysis will represent cumulative estimates for all those years due to the use of cumulative catch numbers. To address this issue, it is recommended to perform catch curve analysis using multiyear mean catch data by creating a **mean_catch_vec**. This ensures a seamless experience during subsequent cohort analysis. As all the available years' individual catch data are averaged in creating **mean_catch_vec**, it contains only one length class-wise mean catch data for the desired combination of years. Therefore, the catch_columns portion of the code should always have the value of 1 **(catch_columns = 1)** to represent this single length class-wise mean catch data for the required combination of years.*

2.7.3. Catch curve analysis for a specific year

If the aim of the analysis is to understand the actual exploitation for a specific year, then the length-converted catch curve analysis can be performed simply by using the catch_vec (containing annual aggregated catches of every available year) and specifying the required year (serial number of the year) in the catch_columns section of the code.

Preparing a catch vector

First, vectorize the catch per month to yearly catches using the following codes:

```
plus_group <- my_data$midLengths[max(which(my_data$midLengths < my_data
$Linf))]
catch_vec <- lfqModify(my_data, vectorise_catch = TRUE, plus_group = plus_group)
```

Note: *The* ***vectorise_catch*** *function aggregates the monthly catches by length class for each year. Depending on the number of years in the LFQ data file, catch vectors are created (e.g., if the dataset contains information for 4 years, four yearly catch vectors are generated).*

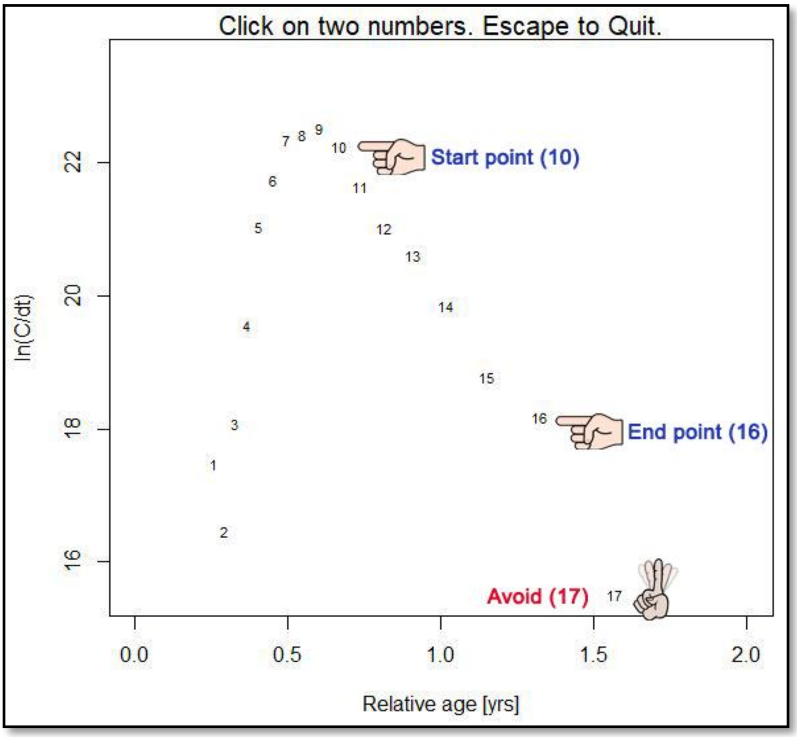
To perform the catch curve analysis for the first year, use the following code:

```
CC <- catchCurve(catch_vec, catch_columns = 1,  calc_ogive = TRUE)
```

To perform the catch curve analysis for the second year, use the following code:

```
CC <- catchCurve(catch_vec, catch_columns = 2,  calc_ogive = TRUE)
```

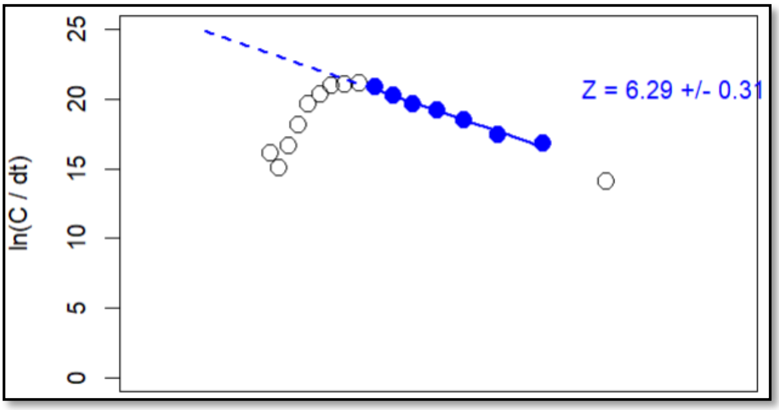
Note: *The highlighted numbers inside the '**catch_columns= 2**' portion of the code can be changed to change the year of the analysis. For example, to include 3rd year, change the highlighted numbers inside the '**catch_columns = 3**'. Similarly, the 4h year of data can be included for the analysis by changing it to '**catch_columns = 4**'.*

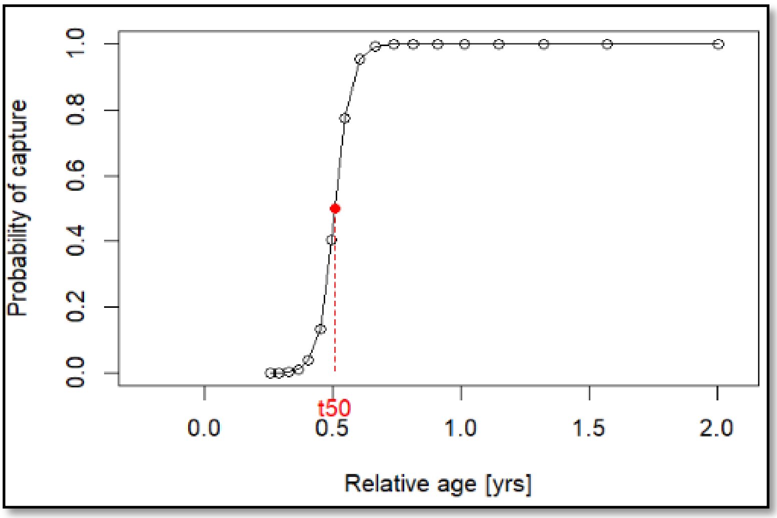


Note: *This will open a new interactive window where the user needs to connect two points to select a sample of catch data for fitting the regression model. Select the point immediately following the peak on the descending arm of the catch curve as the starting point and then choose a point before the last point on the descending arm as the endpoint. It is advisable to ignore the last point, which represents very large fish in the last age group, due to their improper representation of the population.*

2.7.4. Default graphical outputs from the catch curve analysis

After the selection of the end point, a probability of capture curve will be automatically generated with exploitation and selectivity information.



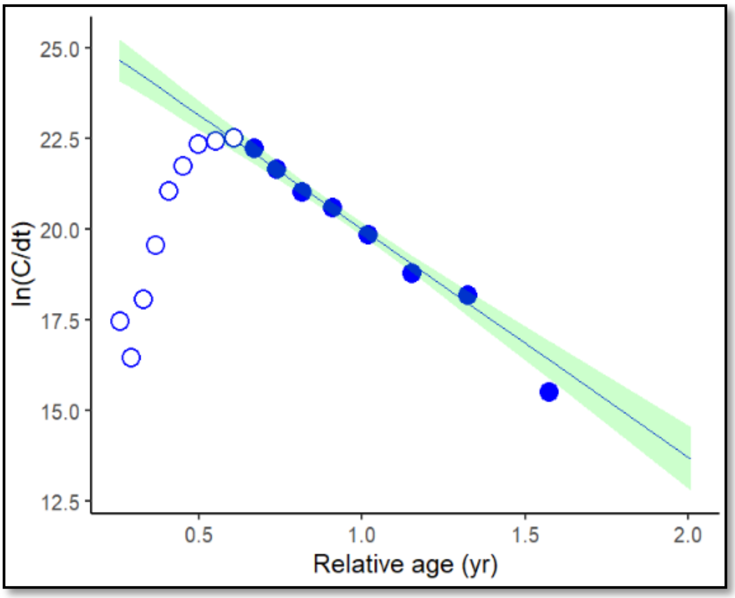


2.7.5. Enhanced visualization of the catch curve analysis

Install and use the **ggplot2** R package for better plotting and the **dplyr** R package for better data handling.

```
install.packages("ggplot2") #Do not install again if already installed
install.packages("dplyr") #Do not install again if already installed
library(ggplot2)
library(dplyr)
#To prepare data for plotting from the catch curve analysis, use the following codes:
model_CC<-CC$linear_mod
CCdata<-as.data.frame(cbind(CC$t_midL, CC$lnC_dt))
```

```
new_predictor <-as.data.frame(CC$t_midL)
colnames(new_predictor)<-"xvar"
confid.int <- predict(model_CC, newdata= new_predictor, interval="confidence")
newCCdata <- cbind(CCdata, confid.int)
fig<-ggplot(newCCdata, aes(x=V1, y=V2)) + geom_point(color="blue", size = 5)+theme_classic(base_size = 15)+labs(y="ln(C/dt)", x="Relative age (yr)")+geom_line(aes(y = fit), color = "blue", linetype = "solid")+geom_line(aes(y = lwr), color = "darkgreen", linetype = "blank") +geom_line(aes(y = upr), color = "darkgreen", linetype = "blank") + geom_ribbon(aes(x = CC$t_midL, ymin = lwr, ymax = upr), fill = "green", alpha=0.2)
fig
#To remove the blue color from the points on the nonselected ascending limb
max_logcatch_values<- newCCdata %>% filter(V2== max(newCCdata$V2, na.rm = TRUE))
ascending_limb<- newCCdata %>% filter(V1<=as.numeric(max_logcatch_values[1]))
fig+ geom_point(data= ascending_limb, aes(x=V1, y=V2), color='white', size=4)
```



2.7.6. Enhanced visualization of the probability of capture

Catch curve model prediction

For better data handling, install and use the **dvmisc** R package using the following codes:

```
install.packages("dvmisc") #Do not install again if already installed
library(dvmisc)
```

#To prepare and plot a data frame from the catch curve analysis, use the following codes:

```
model_sel<-CC$linear_mod_sel
new_predictor <-as.data.frame(seq(min(CC$t_midL), max(CC$t_midL), 0.001))
colnames(new_predictor)<-"t_ogive"
confid.int <- predict(model_sel, newdata= new_predictor, interval="confidence")
prob_capture<-1-logit_prob(confid.int)
midLengths<- as.data.frame(as.numeric(my_data$Linf)*(1-exp(as.numeric(-my_data$K)*new_predictor)))
colnames(midLengths)<-"midLengths"
age<- new_predictor
colnames(age)<-"age"
newseldata <- as.data.frame(cbind(midLengths, age, prob_capture))
```

Deriving the mean and confidence intervals of age at capture

#Use the following codes to calculate the means of tLC_{25}, tLC_{50}, tLC_{75} and tLC_{95}

```
tL25<- as.numeric(log(3)-(coef(model_sel)[1]))/coef(model_sel)[2]
tL50<- as.numeric(-(coef(model_sel)[1]))/coef(model_sel)[2]
tL75<- as.numeric(log(1/3)-(coef(model_sel)[1]))/coef(model_sel)[2]
tL95<- as.numeric(log(1/19)-(coef(model_sel)[1]))/coef(model_sel)[2]
```

#Use the following codes to calculate the confidence intervals of tLC_{25}, tLC_{50}, tLC_{75} and tLC_{95}

```
lower_tLC25<-as.numeric((tail(newseldata %>% filter(lwr<= 0.25), n=1))[2])
upper_tLC25<-as.numeric((tail(newseldata %>% filter(upr<= 0.25), n=1))[2])
lower_tLC50<-as.numeric((tail(newseldata %>% filter(lwr<= 0.5), n=1))[2])
upper_tLC50<-as.numeric((tail(newseldata %>% filter(upr<= 0.5), n=1))[2])
lower_tLC75<-as.numeric((tail(newseldata %>% filter(lwr<= 0.75), n=1))[2])
upper_tLC75<-as.numeric((tail(newseldata %>% filter(upr<= 0.75), n=1))[2])
lower_tLC95<-as.numeric((tail(newseldata %>% filter(lwr<= 0.95), n=1))[2])
upper_tLC95<-as.numeric((tail(newseldata %>% filter(upr<= 0.95), n=1))[2])
```

#Prepare a data frame on mean and confidence intervals of age at capture

```
tLC_confidence <- data.frame (
  Parameter = c("tL25", "tL50", "tL75", "tL95"),
  Mean = c(tL25, tL50, tL75, tL95),
  Lower_95_CI = c(lower_tLC25, lower_tLC50, lower_tLC75, lower_tLC95),
  Upper_95_CI = c(upper_tLC25, upper_tLC50, upper_tLC75, upper_tLC95))
tLC_confidence
```

Deriving the mean and confidence intervals of length at capture

#Use the following codes to calculate the mean of LC_{25}, LC_{50}, LC_{75} and LC_{95}

```
L25<-as.numeric(my_data$Linf)*(1-exp(as.numeric(-my_data $K)*tL25))
L50<-as.numeric(my_data$Linf)*(1-exp(as.numeric(-my_data $K)*tL50))
L75<- as.numeric(my_data $Linf)*(1-exp(as.numeric(-my_data $K)*tL75))
L95<- as.numeric(my_data $Linf)*(1-exp(as.numeric(-my_data $K)*tL95))
```

#Use the following codes to calculate the confidence intervals of LC_{25}, LC_{50}, LC_{75} and LC_{95}

```
lower_LC25<-as.numeric((tail(newseldata %>% filter(lwr<= 0.25), n=1))[1])
upper_LC25<-as.numeric((tail(newseldata %>% filter(upr<= 0.25), n=1))[1])
lower_LC50<-as.numeric((tail(newseldata %>% filter(lwr<= 0.5), n=1))[1])
upper_LC50<-as.numeric((tail(newseldata %>% filter(upr<= 0.5), n=1))[1])
lower_LC75<-as.numeric((tail(newseldata %>% filter(lwr<= 0.75), n=1))[1])
upper_LC75<-as.numeric((tail(newseldata %>% filter(upr<= 0.75), n=1))[1])
lower_LC95<-as.numeric((tail(newseldata %>% filter(lwr<= 0.95), n=1))[1])
upper_LC95<-as.numeric((tail(newseldata %>% filter(upr<= 0.95), n=1))[1])
```

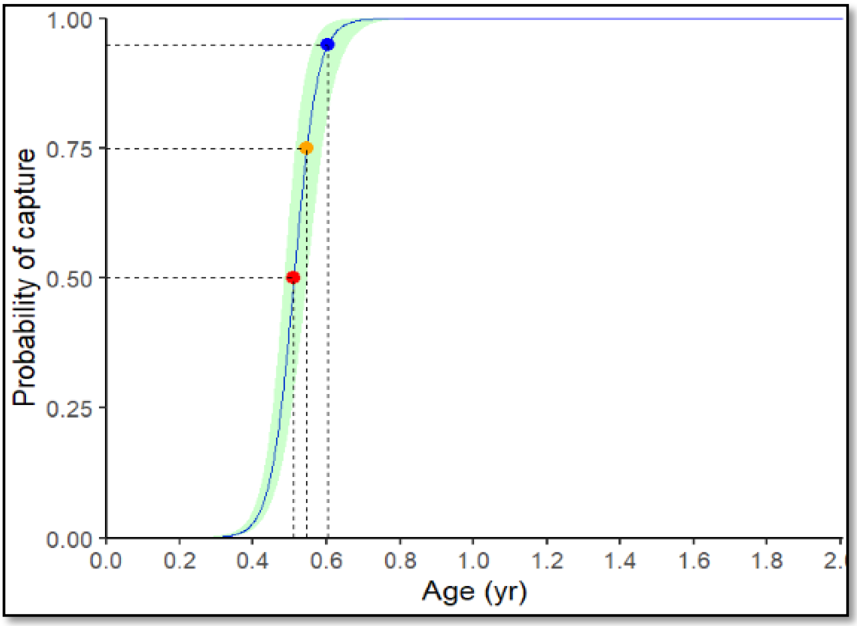
#Prepare a data frame on mean and confidence intervals of length at capture

```
LC_confidence <- data.frame (
  Parameter = c("L25", "L50", "L75", "L95"),
  Mean = c(L25, L50, L75, L95),
  Lower_95_CI = c(lower_LC25, lower_LC50, lower_LC75, lower_LC95),
  Upper_95_CI = c(upper_LC25, upper_LC50, upper_LC75, upper_LC95))
LC_confidence
```

Plotting age vs. probability of capture

Install and use the **ggplot2** R package for better plotting.

```
install.packages("ggplot2") #Do not install again if already installed
library(ggplot2)
fig<-ggplot(newseldata, aes(x=age, y=fit)) + theme_classic(base_size = 15) +
labs(y="Probability of capture", x="Age") + geom_line(aes(y = fit), color = "blue", linetype
= "solid") + geom_line(aes(y = lwr), color = "darkgreen", linetype = "blank") +
geom_line(aes(y = upr), color = "darkgreen", linetype = "blank") + geom_ribbon(aes(x =
age, ymin = lwr, ymax = upr), fill = "green", alpha=0.2) + annotate("point", x = tL50, y =
0.5, size = 3, colour = "red") + annotate("point", x = tL75, y = 0.75, size = 3, colour =
"orange") + annotate("point", x = tL95, y = 0.95, size = 3, colour = "blue") +
annotate("segment", x = 0, y = 0.5, xend = tL50, yend = 0.5, linetype = "dashed") +
annotate("segment", x = tL50, y = 0, xend = tL50, yend = 0.5, linetype = "dashed") +
annotate("segment", x = 0, y = 0.75, xend = tL75, yend = 0.75, linetype = "dashed") +
annotate("segment", x = tL75, y = 0, xend = tL75, yend = 0.75, linetype = "dashed") +
annotate("segment", x = 0, y = 0.95, xend = tL95, yend = 0.95, linetype = "dashed") + annotate("segment", x = tL95, y = 0, xend = tL95, yend = 0.95, linetype = "dashed") + scale_x_continuous(expand = c(0,0), limits = c(0,max(age)), breaks = seq(0,max(age),0.2))+ scale_y_continuous(expand = c(0,0),limits = c(0,1.0))
fig
```

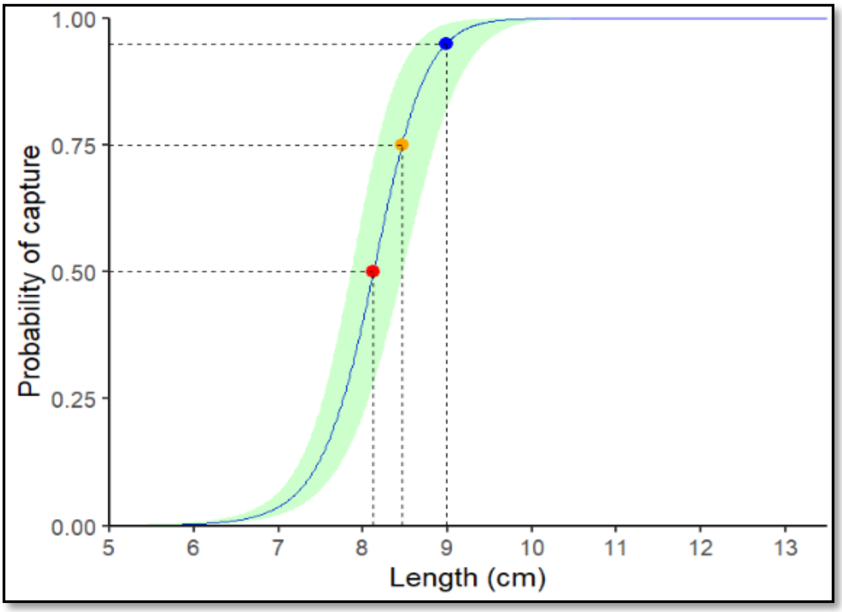


Note: *Control the bold highlighted* ***breaks*** *step size (e.g., 0.2) to decongest X-axis. If the X-axis gets over crowded with small breaks, then increase the break intervals. Ideally 10 nos. of breaks are enough. If the X-axis maximum value is 10 yr, then use 1 (10/10) as the ideal break interval.*

Plotting length vs. probability of capture

Install and use the '**ggplot2**' R package for better plotting.

```
install.packages("ggplot2") #Do not install again if already installed
library(ggplot2)
fig<-ggplot(newseldata, aes(x=midLengths, y=fit)) + theme_classic(base_size = 15) + labs(y="Probability of capture", x="Length (cm)") + geom_line(aes(y = fit), color = "blue", linetype = "solid") + geom_line(aes(y = lwr), color = "darkgreen", linetype = "blank") + geom_line(aes(y = upr), color = "darkgreen", linetype = "blank") + geom_ribbon(aes(x = midLengths, ymin = lwr, ymax = upr), fill = "green", alpha=0.2) + annotate("point", x = L50, y = 0.5, size = 3, colour = "red") + annotate("point", x = L75, y = 0.75, size = 3, colour = "orange") + annotate("point", x = L95, y = 0.95, size = 3, colour = "blue") + annotate("segment", x = min(midLengths), y = 0.5, xend = L50, yend = 0.5) + annotate("segment", x = L50, y = 0, xend = L50, yend = 0.5, linetype = "dashed") + annotate("segment", x = min(midLengths), y = 0.75, xend = L75, yend = 0.75) + annotate("segment", x = L75, y = 0, xend = L75, yend = 0.75, linetype = "dashed") +
annotate("segment", x = min(midLengths), y = 0.95, xend = L95, yend = 0.95) + annotate("segment", x = L95, y = 0, xend = L95, yend = 0.95, linetype = "dashed") + scale_x_continuous(expand = c(0,0), limits = c(min(midLengths),max(midLengths)), breaks = seq(min(midLengths), max(midLengths),1))+ scale_y_continuous(expand = c(0,0),limits = c(0,1.0))
fig
```



***Note:** Control the bold highlighted **breaks** step size (e.g., 1) to decongest the X-axis. If the X-axis gets over crowded with small breaks, then increase the break intervals. Ideally 10 nos. of breaks are enough. If X-axis maximum value is 100 cm, then use 10 (100/10) as the ideal break interval.*

2.7.7. Summarizing derived exploitation parameters from the catch curve analysis

To get all the information on exploitation parameters, use the following code:

```
CC
```

To get the current total mortality rate (Z), use the following code:

```
Z_cur <-CC$Z
Z_cur
```

To get the current fishing mortality rate (F), use the following code:

```
F_cur <-as.numeric(CC$FM)
F_cur
```

To get the current exploitation rate (Ecur), use the following code:

```
E_cur <- as.numeric(CC$FM)/CC$Z
```

```
E_cur
```

To get the current length at capture (LC_{50}), use the following code:

```
LC_cur <- CC$L50
LC_cur
```

2.7.8. Plotting the exploitation parameters derived from the catch curve analysis

Install and load **truncnorm** and **dplyr** R packages for better data handling.

```
install.packages("truncnorm") #Do not install again if already installed
install.packages("dplyr") #Do not install again if already installed
library(truncnorm)
library(dplyr)
```

Define standard deviation (SD) from the model standard error (SE)

```
SD<-sqrt(CC$reg_int[2]-CC$reg_int[1]+1)*CC$se
```

Create a sample of Zs from the model derived mean and SD

```
Zs<-rtruncnorm(n= ncol(my_data$catch), a= CC$confidenceInt[1], b=
CC$confidenceInt[2], mean= CC$Z, sd= SD)
Zs<-as.data.frame(Zs)
Zs<-arrange(Zs, Zs)
```

Create a sample of Fs from the sample of Zs

```
Fs<-Zs$Zs-CC$M
Fs<-as.data.frame(Fs)
```

Create a sample of Es from the sample of Fs and Zs

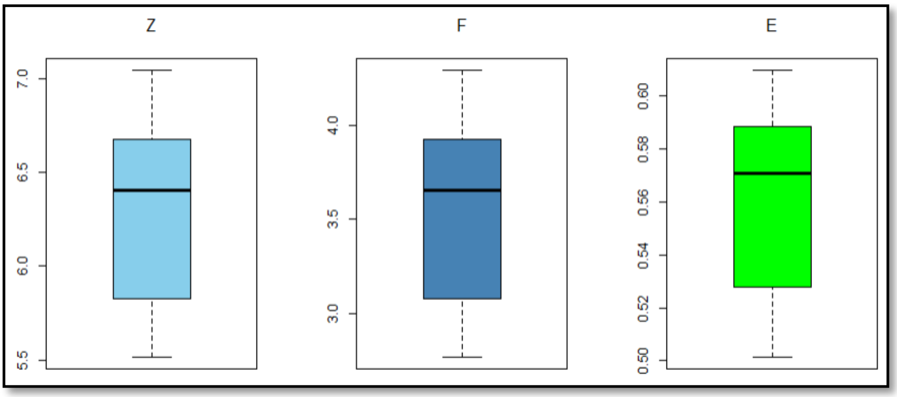
```
Es<-as.data.frame(Fs/Zs)
```

To plot the Z, F, and M confidence intervals, split the plotting screen into one row, having three split columns to plot the three different parameters sequentially.

```
par(mfrow = c(1,3))
```

Box plot for the exploitation parameters

```
boxplot(Zs, main= expression("Z"), col='skyblue')
boxplot(Fs, main= expression("F"), col='steelblue')
boxplot(Es, main= expression("E"), col='green')
```



Note: *Switch off the plot screen split using the following code:* ***par(mfrow = c(1,1))***

Z_data<-c(mean= CC$Z, low= CC$confidenceInt[1], high=CC$confidenceInt[2])

Z_data

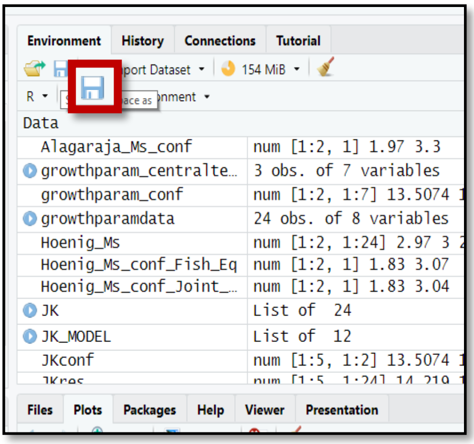
F_data<-c(Mean= CC$Z-CC$M, Lower_95_CI = CC$confidenceInt[1] -CC$M, Upper_95_CI =CC$confidenceInt[2] -CC$M)

F_data

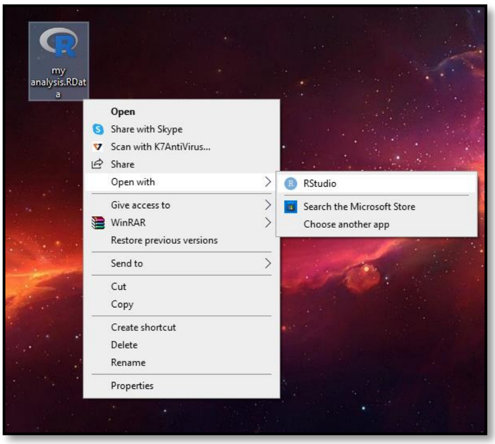
E_data<-F_data/Z_data

E_data

Optional step: If the user needs to close the ongoing session, the work progress can be saved by clicking the ' **save workspace as'** tab in the **'Environment'** panel (located on the top right side). Give the workspace a name (e.g., **'my_analysis'**). In a new session, the user can open the **'my_analysis'** file in RStudio by right-clicking the file and selecting the **'Open with'** RStudio option. Reload the TropFishR library each time a new session starts, using the code: library(TropFishR)



```
Data
  Alagaraja_Ms_conf           num [1:2, 1] 1.97 3.3
  growthparam_centralte…      3 obs. of 7 variables
  growthparam_conf            num [1:2, 1:7] 13.5074 1
  growthparamdata             24 obs. of 8 variables
  Hoenig_Ms                   num [1:2, 1:24] 2.97 3 2
  Hoenig_Ms_conf_Fish_Eq      num [1:2, 1] 1.83 3.07
  Hoenig_Ms_conf_Joint_…      num [1:2, 1] 1.83 3.04
  JK                          List of  24
  JK_MODEL                    List of  12
  JKconf                      num [1:5, 1:2] 13.5074 1
  JKres                       num [1:5, 1:24] 14.219 1
```



2.8. Estimation of length-class wise fishing mortality rate, yield and biomass using virtual population analysis (VPA)/cohort analysis (CA)

Introduction

The virtual population analysis (VPA) and cohort analysis (CA) are widely used modeling technique in fisheries science to estimate the size and structure of fish populations over time. Both the methods reconstruct historical population dynamics by analyzing age- or length-structured data. VPA works by iteratively back-calculating the number of individuals in a cohort (a group of fish born in the same year) from the catch data, accounting for natural and fishing mortality. This method helps to infer population abundance, exploitation rates, and biomass at different life stages, which are critical for sustainable fisheries management.

Virtual population analysis (VPA)

Virtual population analysis (VPA) is a retrospective population modeling technique that reconstructs historical population structures across multiple years by analyzing mortality due to fishing and natural causes. The concept was first introduced by Derzhavin in 1922 and later referred to as the virtual population model by Fry in 1949. This model was further refined by several researchers, including Gulland (1965), Pope (1972), and Jones (1984). Originally, VPA was developed as an age-structured model to use age-based catch data, which is commonly available in temperate fisheries. Later, VPA was adapted into a length-structured model to overcome data limitations in tropical fisheries, where age-based data is difficult to get and length-based information is more prevalent.

In its most basic form, VPA involves solving the Baranov catch equation in a backward direction in time, beginning with the oldest age of each cohort at a time close to the present. The Baranov catch equation in its simplest form can be described as follows:

$$C_t = N_t \times \frac{F_t}{Z_t} \times (1 - exp^{-Z_t \times t})$$

VPA uses a rearranged version of the Baranov catch equation to calculate the abundance (N_t) of the oldest age in a cohort based on catch data as follows:

$$N_t = \frac{Z_t \times C_t}{F_t \times (1 - exp^{-Z_t \times t})}$$

Using the abundance of the oldest cohort (N_t) and the catch of the next younger cohort (C_{t-1}), the mortality rate of the younger cohort is calculated as follows:

$$F_{t-1} = \frac{Z_{t-1} \times C_{t-1}}{N_t \times (exp^{-Z_{t-1} \times t} - 1)}$$

This procedure is repeated until reaching the youngest age for which catch data is available. This approach is often referred to as cohort analysis because each cohort within the stock is analyzed and reconstructed separately from the other cohorts present in the population at the same time. The key distinction between VPA and cohort analysis lies in the method used to calculate fishing mortality for each age class or length group. VPA assumes fish are caught continuously throughout the year and employs a more complex iterative process (Newton-Raphson iteration) to solve the Baranov equation. In contrast, cohort analysis simplifies this by assuming that all fish are caught on a single day, making the calculation less complex (Sparre and Venema, 1998).

Cohort analysis (CA)

Age-based cohort analysis

When age-structured catch data are available, true age classes (cohorts) can be easily tracked over time as they age. Similar to VPA, cohort analysis estimates the number of fish in each age class (cohort) in previous years by working backward from the most recent year. However, unlike VPA, which solves the Baranov equation through a complex iterative process, cohort analysis uses Pope's approximation (Pope, 1972), a simplified solution to the VPA. This simplified method assumes that fish in a cohort are harvested through 'pulse fishing' on a single day, ideally in the middle of the year. As a result, fish in a cohort experience only natural mortality (M) until halfway through the year (Δt/2 = ½ = 0.5 yr). The number of survivors at the end of the half-year period ($N_{y,\ t+0.5}$) can be calculated as follows:

$$N_{y,t+0.5} = N_{y,t} \times exp^{-M\times\frac{1}{2}}$$

In the middle of the year, all the catch for that year ($C_{y,\,t+1}$) is assumed to be harvested on a single day, further reducing the number of survivors ($N_{y,\,t+0.5}$) as follows:

$$N_{y,t+0.5} = N_{y,t} \times exp^{-M\times\frac{1}{2}} - C_{y,t+1}$$

The remaining survivors then experience only natural mortality (M) for the second half of the year (t = 0.5 yr), further reducing their numbers ($N_{y+1,\,t+1}$) as follows:

$$N_{y+1,t+1} = (N_{y,t} \times exp^{-M\times\frac{1}{2}} - C_{y,t+1}) \times exp^{-M\times\frac{1}{2}}$$

To enable backward projection, the equation above can be rearranged as follows:

$$N_{y,t} = (N_{y+1,t+1} \times exp^{M\times\frac{\Delta t}{2}} + C_{y,t+1}) \times exp^{M\times\frac{\Delta t}{2}}$$

From the number of survivors, the fishing mortality rates ($F_{y,\,t,\,t+1}$) can be calculated using the rearranged Z from exponential decay model equations and M as follows:

$$F_{y,t,t+1} = ln\left[\frac{N_{y,t}}{N_{y+1,t+1}}\right] - M$$

To summarize, **the general equation for estimating the number of survivors** in the previous age group (N_t) from the current age group (N_{t+1}) using age-based cohort analysis can be expressed as follows:

$$N_t = (N_{t+\Delta t} \times exp^{M\times\frac{\Delta t}{2}} + C_{t,t+\Delta t}) \times exp^{M\times\frac{\Delta t}{2}}$$

Similarly, **the general equation for calculating the fishing mortality rate** for age groups ($F_{t,\,t+\Delta t}$) using age-based cohort analysis can be expressed as follows:

$$F_{t,t+\Delta t} = \frac{1}{\Delta t} \times ln\left[\frac{N_{y,t}}{N_{t,t+\Delta t}}\right] - M$$

The mean number of survivors in the population is calculated using the following formula:

$$\bar{N}_{t,t+\Delta t} = \left[\frac{N_t - N_{t+\Delta t}}{Z}\right]$$

The formula above is used to estimate the mean or average number of individuals in a population over a specific time or age interval (Δt), accounting for the exponential decline in population due to total mortality.

Length-based cohort analysis

The length-based cohort analysis was developed by Jones (1984) and is popularly known as Jones length-based cohort analysis. This method relies on length frequency data to derive the von Bertalanffy Growth Function (VBGF), which is then used to convert length classes into corresponding age groups. Although these length-converted age groups, sampled at a single point in time (e.g., one year), may come from different cohorts, they are assumed to represent the entire lifespan of a single cohort, known as a pseudocohort. Unlike age-structured cohort analysis, this method uses length-classes as proxies for age-classes of a pseudocohort and estimates the number of fish in each length class historically, treating them as if they were age classes of a cohort. The model assumes a steady-state population with stable length composition over time. It is less sensitive to terminal fishing mortality (F) if F is greater than M, but is highly sensitive to natural mortality (M).

In this approach, the length classes are converted to age groups t(L) using the inverse Von Bertalanffy equation as follows:

$$t(L_1) = t_0 - \frac{1}{K} \times ln\left[1 - \frac{L_1}{L_\infty}\right] \text{ and } t(L_2) = t_0 - \frac{1}{K} \times ln\left[1 - \frac{L_2}{L_\infty}\right]$$

The time (age) interval is calculated as $\Delta t = t(L_2) - t(L_1) = \frac{1}{K} \times ln\left[\frac{L_\infty - L_1}{L_\infty - L_2}\right]$

Here, the inverse proportion of survivors due to natural mortality (M-factor, also expressed as H) until the middle of the age interval, i.e., exp(M*(Δt/2)), is calculated as:

$$M - factor(L_1, L_2) = H(L_1, L_2) = exp\left[M \times \frac{\Delta t}{2}\right] = exp\left[\frac{M}{2} \times \Delta t\right]$$

$$= exp\left[\frac{M}{2} \times \frac{1}{K} \times ln\left[\frac{L_\infty - L_1}{L_\infty - L_2}\right]\right] = \left[\frac{L_\infty - L_1}{L_\infty - L_2}\right]^{\frac{M}{2K}}$$

In length-based cohort analysis, this inverse proportion of survivors due to natural mortality (M-factor or H) can be substituted into **the general equation used in age-based cohort analysis** to get the number of survivors in the previous age group, N(L$_1$), from the current age group, N(L$_2$).

$$N(L_1) = [N(L_2) \times M - factor(L_1, L_2) + C(L_1, L_2)] \times M - factor(L_1, L_2)$$

Similarly, **the general equation for getting the fishing mortality rate** for the age groups (F$_{t,\,t+\Delta t}$) from age-based cohort analysis can be changed to calculate F for the length class, F(L$_1$,L$_2$) as follows:

$$F(L_1, L_2) = \frac{1}{\Delta t} \times ln\left[\frac{N(L_1)}{N(L_2)}\right] - M$$

Alternatively, the general equation for calculating the fishing mortality rate for age groups (F$_{t,t+\Delta t}$) in age-based cohort analysis can be modified to determine F for length class, F(L$_1$,L$_2$) as follows:

$$E(L_1,L_2) = \frac{Catch}{Total\ loss} = \frac{C(L_1,L_2)}{N_1 - N_2} = \frac{N_1 \times \dfrac{F(L_1,L_2)}{Z(L_1,L_2)} \times \left(1 - exp^{-Z(L_1,L_2)\times\Delta t}\right)}{N_1 - N_1 \times exp^{-Z(L_1,L_2)\times\Delta t}}$$

$$= \frac{N_1 \times \dfrac{F(L_1,L_2)}{Z(L_1,L_2)} \times \left(1 - exp^{-Z(L_1,L_2)\times\Delta t}\right)}{N_1 \times (1 - exp^{-Z(L_1,L_2)\times\Delta t})} = \frac{F(L_1,L_2)}{Z(L_1,L_2)}$$

The fishing mortality rate for the length class, $F(L_1,L_2)$ can be calculated using the following formula:

$$F(L_1,L_2) = M \times \left[\frac{\dfrac{F(L_1,L_2)}{Z(L_1,L_2)}}{1 - \dfrac{F(L_1,L_2)}{Z(L_1,L_2)}}\right] = M \times \left[\frac{E(L_1,L_2)}{1 - E(L_1,L_2)}\right]$$

The mean number of survivors $\bar{N}(L_1,L_2)$ in the population is calculated using the following formula:

$$\bar{N}(L_1,L_2) = \left[\frac{N(L_1) - N(L_2)}{Z}\right]$$

The above formula is used to estimate the mean or average number of individuals in a population over a specific length interval (ΔL), accounting for the exponential nature of population decline due to total mortality. The length class-wise catch, $C(L_1, L_2)$ and mean population number, $\bar{N}(L_1, L_2)$ are multiplied with respective mean body weight for the length class, $\bar{W}(L_1, L_2)$ to arrive at the yield (Y_i) and biomass (B_i) for the individual length classes as follows:

$$Y(L_1L_2) = C(L_1L_2) \times \bar{W}(L_1,L_2) \ \ and\ B(L_1L_2) = \ \bar{N}(L_1,L_2) \times \bar{W}(L_1,L_2)$$

The summation of these length class-wise yields and biomasses gives the total yield (Y) and biomass (B).

$$Y = \sum_i Y(L_iL_{i+1}) \ \ and \ \ B = \sum_i B(L_iL_{i+1})$$

The summation of the length class-wise biomasses on and above the length at capture (LC_{50}) is considered as spawning stock biomass (SSB).

Cohort Analysis: R Implementation

2.8.1. Requirement for cohort analysis (CA)

LFQ file

A length-frequency data file (LFQ) is required for the cohort analysis (e.g., my_data). Refer to the previously mentioned steps in ***'2.1.2 and 2.1.5. Creating a length frequency file (LFQ) on R'*** section to newly create a LFQ file if not created earlier.

Essential parameters

Growth and mortality parameters

The cohort analysis requires growth parameters (L_∞ and K) and the natural mortality rate (M). If these essential parameters are not already assigned, use the following code to add them to the LFQ file (e.g., my_data):

```
my_data$Linf<-13.95
my_data $K<-1.71
my_data$M<-2.75
```

LWR coefficients

To derive the LWR coefficients (a, and b) refer to ***'2.12. Length–Weight Relationship (LWR)'*** section. Assign the coefficients 'a', and 'b' to the LFQ file (e.g., my_data) using the following codes:

```
my_data $a <- 0.0064
my_data $b <- 3.0059
```

Maturity parameters

To derive the length and weight at maturity (LM_{50} and WM_{50}), refer to ***'2.13. Length at Maturity (LM_{50})'*** section. Assign the LM_{50}, WM_{50} values to the LFQ file (e.g., my_data) using the following codes:

```
my_data $Lmat <- 8.24
my_data$wmat <- my_data$a*(my_data$Lmat^my_data$b)
```

2.8.2. Cohort analysis for a combination of years using multiyear mean catch data

When population estimates (e.g., yield, biomass, spawning stock biomass, recruitment, and stock size) are needed for a specific combination of years, it is essential to use the length structured mean catch data (i.e., mean_catch_vector) for those years to get accurate estimates of population parameters. To prepare the mean_catch_vector for the required combination of years, refer to the '*Preparing a mean catch vector from the catch vector*' section in ***'2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data'***. For example, to prepare a mean catch vector for all available years (e.g., 4 years), use the following codes first to prepare the catch_vector:

```
plus_group <- my_data$midLengths[max(which(my_data$midLengths < my_data $Linf))]
catch_vec <- lfqModify(my_data, vectorise_catch = TRUE, plus_group = plus_group)
```

Then use the following code for preparing a **mean_catch_vector** from the **catch_vector** as follows:

```
mean_catch_vec<- catch_vec
mean_catch_vec$catch<-as.matrix(rowMeans(catch_vec$catch[,1:4]))
```

Note: *The highlighted numbers inside the 'catch_vec$catch[,**1:4**]' portion of the code can be changed to include or exclude year(s) from the analysis. For example, to include only the 1st, 2nd and 3rd year, change the highlighted numbers inside the 'catch_vec$catch[,**1:3**]'. Similarly, the 3rd, 4th and 5th year of data can be included for the analysis by changing it to 'catch_vec$catch[,**3:5**]'.*

To perform the cohort analysis (CA) for the required combination of years, use the following code:

```
cohortanalysis <- VPA(param = mean_catch_vec, catch_columns = 1, catch_unit=
"'000", terminalE = 0.5, analysis_type = "CA", plot= TRUE)
```

> ***Note:*** *The analysis type in the above code can be changed to VPA (analysis_type = "VPA") to conduct a length-based VPA analysis instead. Since the length structured catch data for all available years has been averaged to create the mean_catch_vec, it contains only a single length-structured catch data for the required combination of years. Therefore, the catch_columns portion of the code should always be set to 1 (catch_columns = 1), representing the sole length-structured mean catch data for the required years.*

2.8.3. Cohort analysis for a specific year

When population estimates (e.g., yield, biomass, spawning stock biomass, recruitment, and stock size) are needed for a specific year, the cohort analysis for the specific year can be done simply by using the catch_vec (containing annual aggregated catches of every available year) and specifying the required year (serial number of the year) in the catch_columns section of the code. To prepare the catch_vector, refer to the '*Preparing a catch vector*' section in ***'2.7.3.Catch curve analysis for a specific year'***. For example, to prepare a catch vector, use the following codes:

```
plus_group <- my_data$midLengths[max(which(my_data$midLengths < my_data
$Linf))]
```

```
catch_vec <- lfqModify(my_data, vectorise_catch = TRUE, plus_group = plus_group)
```

To perform Cohort Analysis (CA) for the **first year**, use the following code:

```
cohortanalysis <- VPA(param = catch_vec, catch_columns = 1, catch_unit= "'000",
terminalE = 0.5, analysis_type = "CA", plot= TRUE)
```

To perform Cohort Analysis (CA) for the **fourth year**, use the following code:

```
cohortanalysis <- VPA(param = catch_vec, catch_columns = 4, catch_unit= "'000",
terminalE = 0.5, analysis_type = "CA", plot= TRUE)
```

> ***Note:*** *The highlighted numbers inside the '**catch_columns= 4**' portion of the code can be changed to change the year of the analysis. For example, to include 2nd year, change the highlighted numbers inside the '**catch_columns = 2**' portion of the code. Similarly, if there are 5 years of data, the 5th year of data can be included for the analysis by changing it to '**catch_columns = 5**'.*

2.8.4. Summarizing the population estimates from the length-based cohort analysis

Regardless of the duration of the analysis (whether for a combination of years or a specific year), key population estimates can be derived using the following codes after performing the Cohort Analysis (CA):

Install and use the **dplyr** for better data handling

```
install.packages("dplyr") #Do not install again if already installed
```

```
library(dplyr)
Yield_CA<-sum(cohortanalysis$yieldTon)/1000000
Biomass_CA<-sum(cohortanalysis$meanBiomassTon)/1000000
SSB_data<- data.frame(Length=cohortanalysis$midLengths,
SSB=cohortanalysis$meanBiomassTon)
SSB_CA<- (sum(head(filter(SSB_data, Length> (my_data$Lmat-
(my_data$midLengths[2]-my_data$midLengths[1]))))))/1000000
Recruitment_CA<- (cohortanalysis$ survivors_L1[1])/1000
Stocksize_CA<-( sum(cohortanalysis$annualMeanNr))/1000
CA_results<-data.frame(Yield_CA, Biomass_CA, SSB_CA, Recruitment_CA,
Stocksize_CA)
CA_results
```

Note: *The* ***catch_unit = "'000"*** *is used to indicate that the catch numbers are in thousands. However, in this example, the actual catch numbers (without dividing by 1,000) have been used for the ease of analysis by specifying the catch numbers are in thousands. As a result, the recruitment and stock size, being numerical counts, are scaled up by 1000 times. Therefore, the recruitment and stock size are needed to be scaled down by 1,000 to adjust for the use of actual catch numbers instead of numbers in thousands (catch numbers/1000). Since the coefficients of the length-weight relationship (LWR) provided in the example are in 'centimeter to gram' resolution, the output for yield and biomass will be in grams. In spite of using the actual catch numbers (without dividing by 1,000), the estimates of yield and biomass are expressed in actual grams. Therefore, the yield and biomass outputs are required to be divided with 1,000,000 to express them in tonnes.*

Alternatively, same results will be obtained by dividing catch numbers with 1000 first and then using the same in the cohort analysis by mentioning ***catch_unit = "'000"*** *in the cohort analysis code. However, scaling the catch numbers to thousands (catch numbers/1000) is a little tricky, which should be done while preparing the catch vector or mean catch vector.*

For example, to create a ***catch vector*** *scaled to a thousand catch numbers (catch numbers/1000), use the following code:* `catch_vec$catch<- catch_vec$catch/1000`

And to create a ***mean catch vector*** *scaled to a thousand catch numbers (catch numbers/1000), use the following code:* `mean_catch_vec$catch<- mean_catch_vec$catch/1000`

The above ***two codes are not required*** *if the analysis is carried out with actual catch numbers by simply adjusting the outputs using the above-mentioned scaling factors.*

To create a data frame of the cohort analysis outputs, use the following codes:

```
CA_output<-data.frame(ML=cohortanalysis$classes.num,
Survivors1=cohortanalysis$survivors_L1/1000,
Survivors2=cohortanalysis$survivors_L2/1000,
Catch=cohortanalysis$catch_numbers/1000,
Natural_loss=cohortanalysis$natLoss/1000, F=cohortanalysis$FM_calc,
Z=cohortanalysis$Z, Annual_mean_number=cohortanalysis$annualMeanNr/1000,
Mean_BW_gram=cohortanalysis$meanBodyWeight,
```

Annual_mean_biomass_tonne=cohortanalysis$meanBiomassTon/1000000, Yield_tonne=cohortanalysis$yieldTon/1000000)

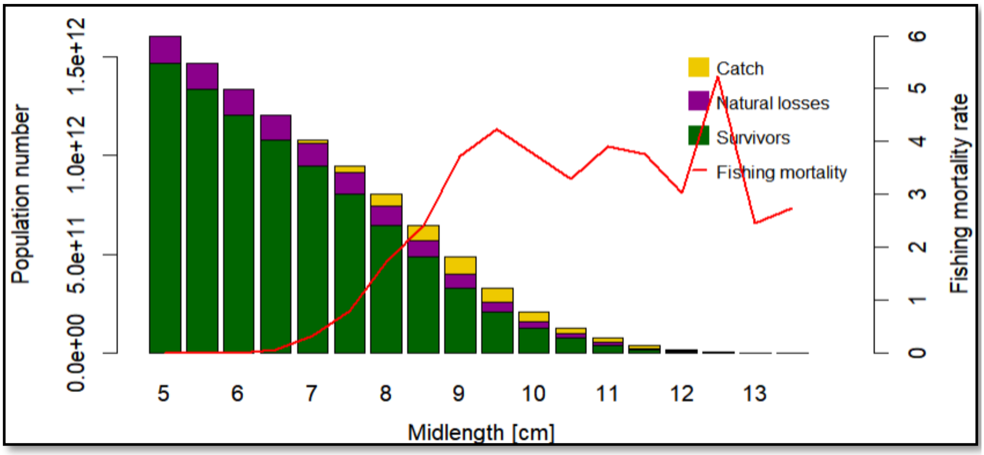
CA_output

2.8.5. Default graphical outputs from the cohort analysis results

Population number plot

Use the following code to produce a length class wise population number plot:

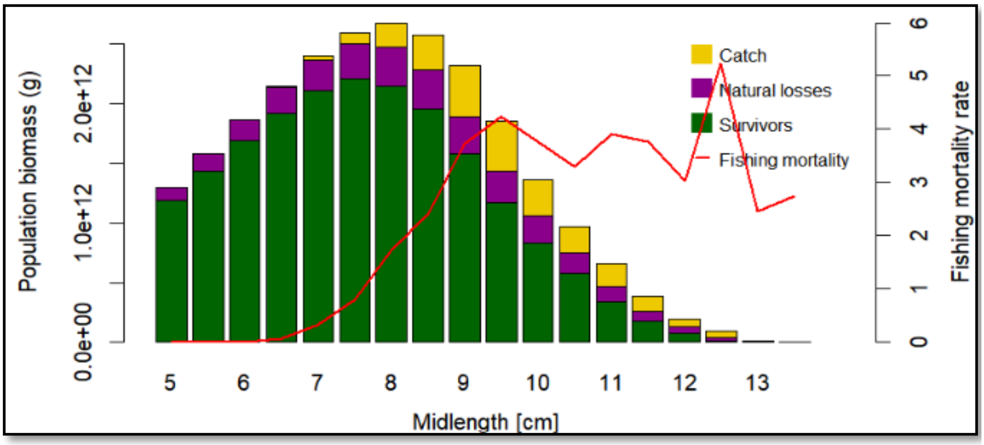
```
plot(cohortanalysis, yaxis = "numbers", display_last_class = TRUE, xlabel = NA, ylabel1 = "Population number", ylabel2 = "Fishing mortality rate", ylim = NA, ylim_FM = NA, plot.bars = TRUE, plot.FM = TRUE, plot.legend = TRUE)
```



Population biomass plot

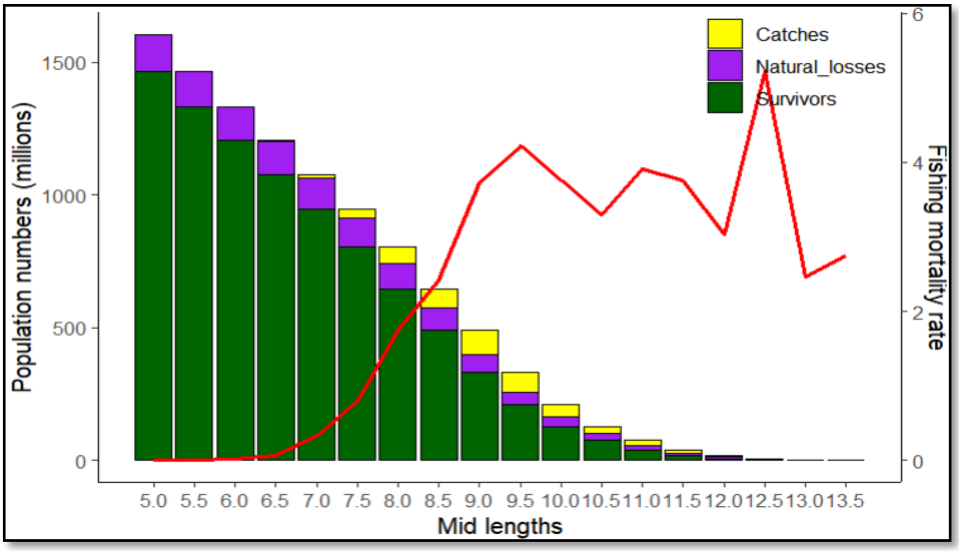
Use the following code to produce a length class-wise population biomass plot:

```
plot(cohortanalysis, yaxis = "biomass", display_last_class = TRUE, xlabel = NA, ylabel1 = "Population biomass (g)", ylabel2 = "Fishing mortality rate", ylim = NA, ylim_FM = NA, plot.bars = TRUE, plot.FM = TRUE, plot.legend = TRUE)
```



2.8.6. Enhanced visualization of the cohort analysis results

Population number plot (2D)



The *CA_output* data frame produced in the previous step already has the population indices (survivors, natural losses, catches) adjusted to actual numbers. To express these indices in millions, create a new data frame (e.g., cohort_number_data) by dividing these population indices by one million.

```
cohort_number_data<- data.frame (Lengths= CA_output$ML, Survivors= CA_output$Survivors2/1000000, Natural_losses= CA_output$Natural_loss/1000000, Catches= CA_output$Catch/1000000, F= CA_output$F)
```

Install and load the reshape2 package to melt and arrange the data for the plot and ggplot2 for better plotting (Do not install again if already installed).

```
install.packages("reshape2") #Do not install again if already installed
install.packages("ggplot2") #Do not install again if already installed
library(reshape2)
library(ggplot2)
numberdata<- melt(cohort_number_data, id.vars = c("Lengths", "F"))
```

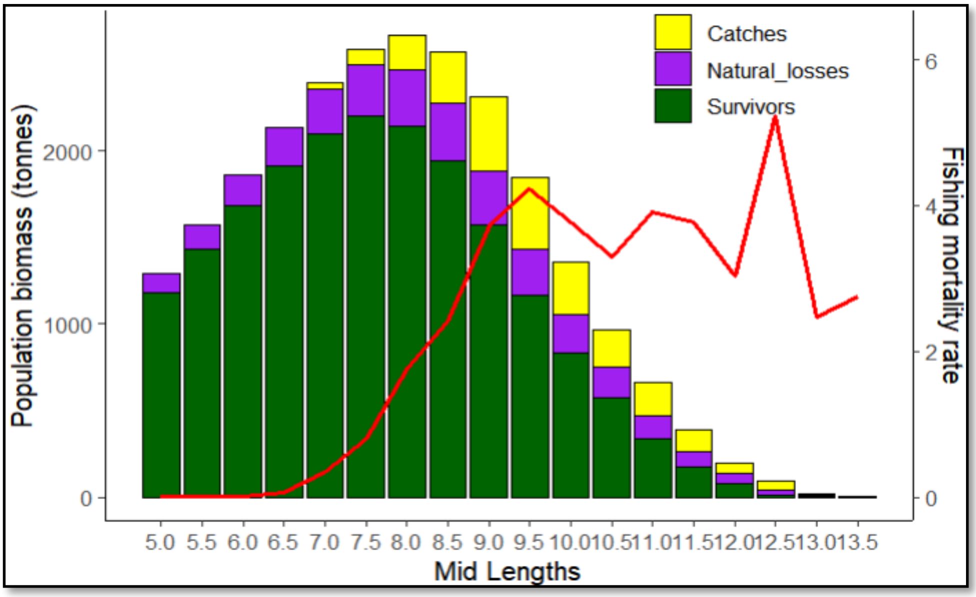
Use the following code to change and define the order of the variables in the stacks:

```
numberdata$variable <- factor(numberdata$variable, levels=c("Catches","Natural_losses", "Survivors"))
```

Use the following code to prepare a secondary Y-axis for fishing mortality rate (F):

```
scale1<-max(numberdata$value)/max(numberdata$F)
ggplot(numberdata, aes(x=Lengths, y=value, fill=variable)) + geom_bar(position="stack", stat="identity", color = "black") + scale_fill_manual(values=c("yellow", "purple",
"darkgreen"))+geom_line(aes(x=Lengths, y=F*scale1), stat="identity", color="red", linewidth=1)+ scale_y_continuous(sec.axis=sec_axis(~./scale1, name="Fishing mortality rate")) + scale_x_continuous(breaks = CC$midLengths)+ xlab("Mid lengths") + ylab("Population numbers (millions)")+ theme_classic(base_size = 12) + theme(legend.title = element_blank(), legend.background = element_rect(fill='transparent'), legend.position.inside=c(0.87, 0.9))
```

Population biomass plot (2D)



The CA_output data frame produced in the previous step has the population indices (survivors, natural losses, catches) already adjusted to actual numbers. Create a new data frame (e.g., cohort_biomass_data) by multiplying these population numbers by the mean body weight to get the biomass for each index in grams. Then, divide the resulting values by one million to express the biomass in tonnes.

```
cohort_biomass_data<- data.frame (Lengths= CA_output$ML, Survivors= CA_output$Survivors2*CA_output$Mean_BW_gram/1000000, Natural_losses= CA_output$Natural_loss*CA_output$Mean_BW_gram/1000000, Catches= CA_output$Catch*CA_output$Mean_BW_gram/1000000, F= CA_output$F)
```

Load the reshape2 package to melt and arrange the data for the plot

```
library(reshape2)
```

```
biomassdata<- melt(cohort_biomass_data, id.vars = c("Lengths", "F"))
```

Use the following code to change and define the order of the variables in the stacks:

```
biomassdata$variable <- factor(biomassdata$variable,
levels=c("Catches","Natural_losses", "Survivors"))
```

Use the following code to prepare a secondary Y-axis for F:

```
scale2<-max(biomassdata$value)/max(biomassdata$F)
```

Load the ggplot2 for better plotting:

```
library(ggplot2)

ggplot(biomassdata, aes(x=Lengths, y=value, fill=variable)) + geom_bar(position="stack", stat="identity", color = "black") + scale_fill_manual(values=c("yellow", "purple", "darkgreen"))+geom_line(aes(x=Lengths, y=F*scale2), stat="identity", color="red", linewidth=1)+ scale_y_continuous(sec.axis=sec_axis(~./scale2, name="Fishing mortality rate")) + scale_x_continuous(breaks = CC$midLengths)+ xlab("Mid Lengths") + ylab("Population biomass (tonnes)")+ theme_classic(base_size = 12)+ theme(legend.title = element_blank(),legend.background = element_rect(fill='transparent'), legend.position.inside=c(0.8, 0.9))
```

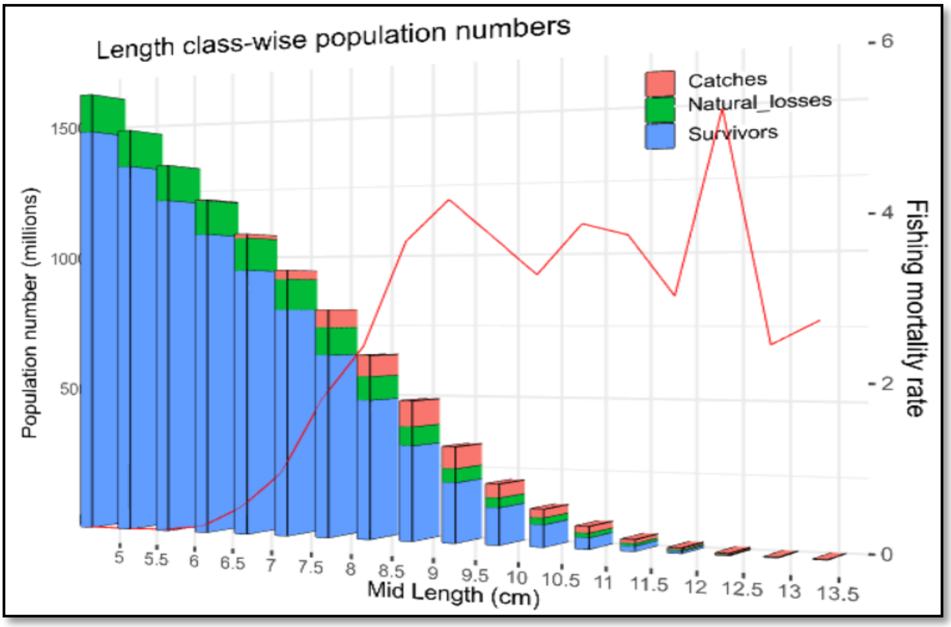
For 3D visualization, first install and load the following R-packages (Do not install again if already installed)

```
remotes::install_github('coolbutuseless/devout')

remotes::install_github('coolbutuseless/devoutrgl')

remotes::install_github('coolbutuseless/triangular')

remotes::install_github('coolbutuseless/snowcrash')

remotes::install_github('coolbutuseless/cryogenic')

remotes::install_github('coolbutuseless/ggrgl', ref='main')

library(rgl)

library(devout)

library(devoutrgl)

library(ggrgl)

library(ggplot2)
```

Population number plot (3D)

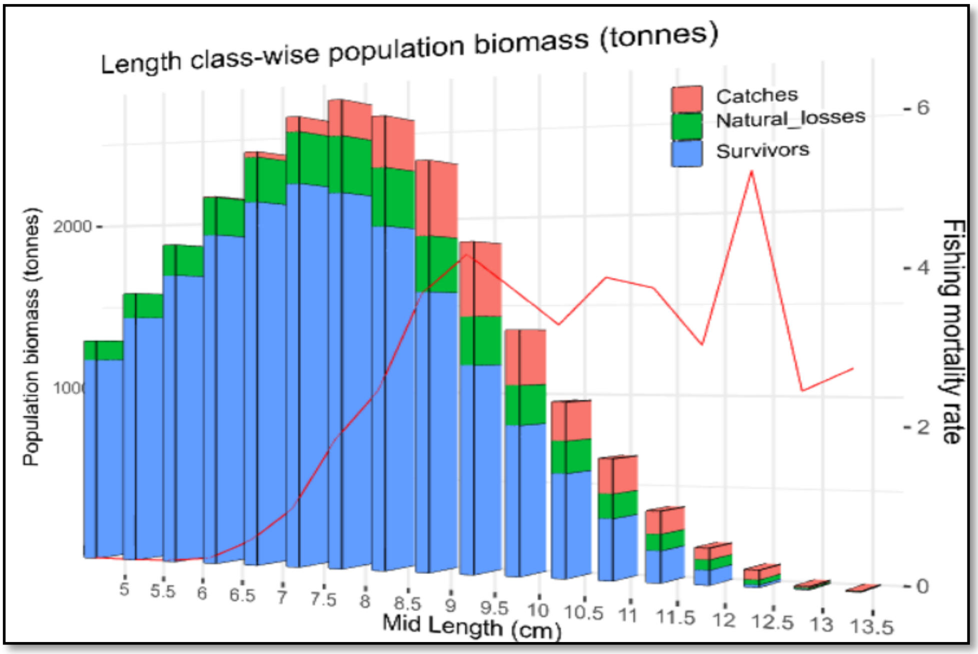
```
suppressWarnings({

fig1<- ggplot(numberdata, aes(fill = variable, y = value, x = as.factor(Lengths), z = 1, extrude_face_fill = variable)) + geom_bar_z(position = "stack", stat = "identity", width = 0.3, extrude = TRUE, color = "black", extrude_edge_color = "black") + geom_line_3d(aes(x = as.factor(Lengths), y = F * scale1), stat = "identity", group = 1, color = "red", linewidth = 1.5) + scale_y_continuous(sec.axis = sec_axis(~ . / scale1, name = "Fishing mortality rate")) + xlab("Mid Length (cm)") + ylab("Population number (millions)") + theme_ggrgl(base_size = 15) + labs(title = "Length class-wise population numbers") + theme( legend.title = element_blank(), legend.position.inside = c(0.85, 0.9) )

devoutrgl::rgldev(fov = 30, view3d_args = list(theta = 25, phi = 0, zoom = 0.7), dpi = 100)

print(fig1)

# Save as PNG with high resolution

rgl::rgl.snapshot("high_quality_plot.png", fmt = "png", top = TRUE)

})

# Close the rgl device after plotting
```

```
rgl::rgl.close()
```



Population biomass plot (3D)

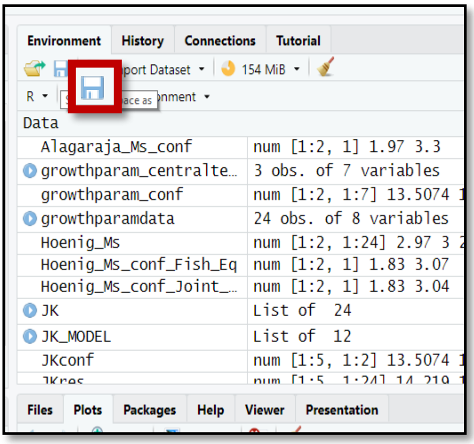
```
suppressWarnings({
fig2<- ggplot(biomassdata, aes(fill = variable, y = value, x = as.factor(Lengths), z = 1, extrude_face_fill = variable)) + geom_bar_z(position = "stack", stat = "identity", width = 0.3, extrude = TRUE, color = "black", extrude_edge_color = "black") + geom_line_3d(aes(x = as.factor(Lengths), y = F * scale2), stat = "identity", group = 1, color = "red", linewidth = 2.5) + scale_y_continuous(sec.axis = sec_axis(~ . / scale2, name = "Fishing mortality rate")) + xlab("Mid Length (cm)") + ylab("Population biomass (tonnes)") + theme_ggrgl(base_size = 15) + labs(title = "Length class-wise population biomass (tonnes)") + theme(legend.title = element_blank(), legend.position = c(0.85, 0.9))
devoutrgl::rgldev(fov = 30, view3d_args = list(theta = 25, phi = 0, zoom = 0.7), dpi = 100)
print(fig2)
# Save as PNG with high resolution
rgl::rgl.snapshot("high_quality_plot.png", fmt = "png", top = TRUE)
})
# Close the rgl device after plotting
rgl::rgl.close()
```



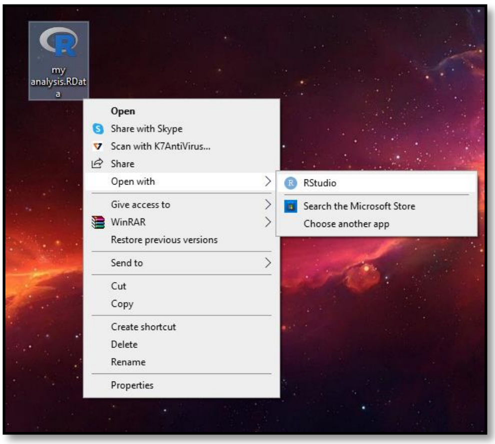
Switch off the 3D plotting window after the plots are generated using the following code:

invisible(dev.off())

Optional step: If the user needs to close the ongoing session, the work progress can be saved by clicking the '💾 **save workspace as'** tab in the **'Environment'** panel (located on the top right side). Give the workspace a name (e.g., **'my_analysis'**). In a new session, the user can open the **'my_analysis'** file in RStudio by right-clicking the file and selecting the **'Open with'** RStudio option. Reload the TropFishR library each time a new session starts, using the code: library(TropFishR)



```
Data
  Alagaraja_Ms_conf        num [1:2, 1] 1.97 3.3
● growthparam_centralte…   3 obs. of 7 variables
  growthparam_conf         num [1:2, 1:7] 13.5074 1
● growthparamdata          24 obs. of 8 variables
  Hoenig_Ms                num [1:2, 1:24] 2.97 3 2
  Hoenig_Ms_conf_Fish_Eq   num [1:2, 1] 1.83 3.07
  Hoenig_Ms_conf_Joint_…   num [1:2, 1] 1.83 3.04
● JK                       List of  24
● JK_MODEL                 List of  12
  JKconf                   num [1:5, 1:2] 13.5074 1
  JKres                    num [1:5, 1:24] 14.219 1
```



2.9. Stock Simulation by Thompson and Bell's model (TB)

Introduction

Unlike VPA and CA, which are retrospective models that back-calculate cohort yield and biomass, the predictive models are used to simulate and predict the effects of changes in fishing mortality rate (F), either alone or in combination with gear selectivity (LC_{50}), on stock status. There are two predictive models, also known as dynamic pool models or yield-per-recruit models, which are widely used in fisheries resource management: (1) the Thompson and Bell model (TB), and (2) the Beverton and Holt yield-per-recruit model (BHYPR).

Thompson and Bell's model (Thompson and Bell, 1934) is a widely used predictive tool in fisheries science for analyzing the effects of fishing on fish populations. Initially developed as an age-based model for temperate fisheries, it was later adapted into a length-based model to address data-limited tropical fisheries. This model facilitates the evaluation of various fishing strategies, helping to identify optimal levels of fishing mortality and gear selectivity that maximize yield while ensuring the sustainability of fish populations. A key feature of the Thompson and Bell model is its ability to incorporate bio-economic analysis when the value of the catch is provided as input. Additionally, unlike the Beverton and Holt model, which estimates yield on a per-recruit basis, the Thompson and Bell model can predict yield, biomass, and spawning stock biomass in absolute terms.

The model essentially uses length-class-specific fishing mortalities derived either (1) from VPA/CA or (2) from selectivity information generated through length-converted catch curve analysis. These fishing mortalities are then multiplied by an F-multiplier to simultaneously increase or decrease mortality rates across all length classes, simulating the effect of changes in F on yield and biomass. Adjusting the F-multiplier alters the fishing mortality rate (F_i) for each length class (L_i) as follows:

$$F_i = F_i \times F - multiplier$$

This alters the total mortality rate (Z_i) for each length class (L_i) as follows:

$$Z_i = M_i + F_i$$

The inverse proportion of survivors due to natural mortality (M-factor$_i$, also expressed as H_i) for each length class (L_i) is calculated as follows:

$$M - factor_i = H_i = \left[\frac{L_\infty - L_1}{L_\infty - L_2}\right]^{\frac{M}{2K}}$$

This M-factor$_i$ or H_i is used to calculate the population size for the successive length class (N_{i+1}) as follows:

$$N_{i+1} = N_i \times \left[\frac{\frac{1}{H_i} - \frac{F_i}{H_i}}{H_i - - \frac{F_i}{H_i}}\right]$$

The mean catch ($\bar{C}_i$) for each length class (L_i) is calculated as follows:

$$C_i = [N_i - N_{i+1}] \times \frac{F_i}{Z_i}$$

The mean population size ($\bar{N}_i$) for each length class (N_i) is calculated as follows:

$$\bar{N}_i = \left[\frac{N_i - N_{i+1}}{Z}\right]$$

The length class-wise mean catch ($\bar{C}_i$) and mean population number ($\bar{N}_i$) are multiplied by the corresponding mean body weight for the length class ($\bar{W}_i$) to calculate the yield (Y_i) and biomass (B_i) for the length classes. The length class-wise yield (Y_i) is then multiplied by the price per kg (₹) for the corresponding length class to derive economic yield (EY_i) for each length-class.

$$Y_i = \sum_i \bar{C}_i \times \bar{W}_i \quad and \quad EY_i = \sum_i \bar{C}_i \times \bar{W}_i \times ₹ \quad and \quad B_i = \sum_i \bar{N}_i \times \bar{W}_i$$

The summation of these length class-wise yields, economic yield and biomasses provides the total yield, total economic yield, and total biomass.

$$Y = \sum_i Y_i \quad and \quad EY = \sum_i EY_i \quad and \quad B = \sum_i B_i$$

The summation of the length class-wise biomasses for lengths equal to or greater than the length at maturity (LM_{50}) is considered the spawning stock biomass (SSB).

Thompson and Bell's model (TB): R Implementation

2.9.1. Requirement for Thompson and Bell's model (TB)

LFQ file

A length-frequency data file (LFQ) is required for the Thompson and Bell analysis (e.g., my_data). Refer to the previously mentioned steps in ***'2.1.2 and 2.1.5. creating a length frequency file (LFQ) on R'*** section to newly create a LFQ file if not created earlier.

Essential parameters

Growth and mortality parameters

The Thompson and Bell analysis requires growth parameters (L_∞ and K) and the natural mortality rate (M). If these essential parameters are not already assigned, use the following code to add them to the LFQ file (e.g., my_data):

```
my_data$Linf<-13.95

my_data $K<-1.71

my_data$M<-2.75
```

LWR coefficients

The Thompson and Bell analysis requires the LWR coefficients (a and b). To derive the LWR coefficients, refer to ***'2.12.Length–Weight Relationship (LWR)'*** section. Assign the coefficients 'a', and 'b' to the LFQ file (e.g., my_data) using the following codes:

```
my_data $a <- 0.0064

my_data $b <- 3.0059
```

Maturity parameters

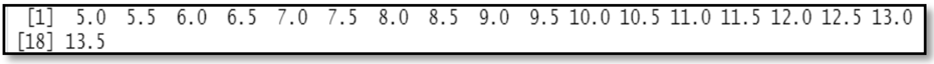
The Thompson and Bell analysis requires the maturity parameters (LM_{50} and WM_{50}). To derive the length and weight at maturity (LM_{50} and WM_{50}) refer to **'*2.13.Length at Maturity (LM_{50})*'** section. Assign the LM_{50}, WM_{50} values to the LFQ file (e.g., my_data) using the following codes:

```
my_data $Lmat <- 8.24
my_data$wmat <- my_data$a*(my_data$Lmat^my_data$b)
```

Economic value (optional)

The length class-wise value (price in INR/kg) information for the species is an optional requirement for the Thompson and Bell analysis, which is only needed for calculating economic yield during stock simulation. First check how many mid-lengths are there in the LFQ using the following code:

```
my_data$midLengths
```



```
 [1]  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0
[18] 13.5
```

There are 18 mid-lengths in this LFQ data frame example (e.g., my_data). Therefore, the user needs to assign 18 separate prices in INR/kg, one price for each length class. For example, user may assign a lower actual realized price of 80 INR/kg for the smaller 5 cm mid-length prawn group which can gradually increase to a higher realized price of 250 INR/kg for the larger 13.5 cm mid-length prawn group. Please note that the actual price information for each length class may vary depending on the market demand of the species. Prepare the actual market price vector for the species using the following code:

```
my_data $ meanValue<-
c(80,80,80,100,100,100,150,150,150,175,175,175,200,200,200,250,250,250)
```

2.9.2. Simulating the effect of changes in FM on the stock status (TB1)

This approach is used to simulate the effect of changes in fishing mortality rate (FM) on stock status.

Step-1: Assign the fishing mortality rates (FMs)

The length class-wise fishing mortality rates are required for the TB analysis, which can be derived using either of the following two approaches:

Catch curve analysis derived FMs

This approach uses the single annual fishing mortality rate (FM) and gear selectivity information obtained from a catch curve analysis to derive the length class-wise fishing mortality rates (FMs).

First perform a catch curve analysis for the specified combination of years (e.g., all available years using **mean_catch_vec**) using the following code to get the probability of capture, which is then used to derive length class-wise fishing mortality rates (FMs).

```
CC <- catchCurve(mean_catch_vec, catch_columns = c(1), calc_ogive = TRUE)
```

Assign the annual fishing mortality rate (FM) for the stock, derived from the catch curve analysis, to the **my_data.**

```
my_data$FM<-as.numeric(CC$FM)
```

Cohort analysis derived FMs

This approach uses length class-wise fishing mortality rates (FMs) derived directly from cohort analysis.

First, perform a cohort analysis for the specified combination of years (e.g., all available years using the **mean_catch_vec**) to derive length class-wise fishing mortality rates (FMs).

```
cohortanalysis <- VPA(param = mean_catch_vec, catch_columns = 1, catch_unit= "'000", terminalE = 0.5, analysis_type = "CA", plot= TRUE)
```

Assign the length class-wise fishing mortality rates (FMs) generated from the cohort analysis to the **my_data** using the following code:

```
my_data$FM<- cohortanalysis$FM_calc
```

Note: *This is a very crucial step in the analysis. The FMs can be derived either through the cohort analysis or a catch curve analysis. The above example shows the procedure for deriving FMs for a specific combination of years (e.g., all available years using* ***mean_catch_vec****) using the length converted catch curve analysis or the cohort analysis. Alternatively, the user can derive FMs for any specified year using the* ***catch_vec*** *in the length converted catch curve analysis or the cohort analysis by specifying the serial number of the required year in the catch column of the respective codes* ***(catch_columns =).*** *To perform catch curve analysis for the required combination of years, refer to the '****2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data****'. To perform catch curve analysis for the required year, refer to the '****2.7.3.Catch curve analysis for a specific year****'. To perform cohort analysis for the required combination of years, refer to the '****2.8.2.Cohort analysis for a combination of years using multiyear mean catch data****'. To perform cohort analysis for the required year, refer to the '****2.8.3.Cohort analysis for a specific year****'.*

Step-2: Perform TB1 with a catch curve analysis derived FM under the knife-edge selection assumption

Ensure that the my_data contains the single annual FM derived from a catch curve analysis. To ensure the function follows the knife-edge selection assumption instead of the gear selectivity assumption, use the TB code by providing a knife-edge selection list (s_list = knife-edge).

To prepare a knife-edge selection list using the LC_{50} from the catch curve analysis, use the following code:

```
knife_edge <- list(selecType = "knife_edge", L50 = CC$L50)
```

To perform the TB1 analysis with knife-edge selection assumption, use the following code:

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = knife_edge, plot = TRUE)
```

Step-3: Perform TB1 with a catch curve analysis derived FMs under the gear selection assumption

Ensure that the my_data contains the single annual FM derived from the catch curve analysis. Provide selectivity information (s_list = gear_selection) in the TB code to ensure the function follows the gear selectivity assumption instead of the knife-edge selection assumption.

Prepare a gear selection list (gear_selection) using the LC_{50} and LC_{75} information derived from a catch curve analysis with the following code:

```
gear_selection <- list(selecType = "trawl_ogive", L50 = CC$L50, L75 = CC$L75)
```

To perform the TB1 analysis with gear selection assumption, use the following code:

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = gear_selection, plot = TRUE)
```

Step-4: Perform TB1 with the cohort analysis derived FMs under the knife-edge selection assumption

Ensure that the my_data includes the length class-specific FMs derived from a cohort analysis. To ensure the function follows the knife-edge selection assumption instead of the gear selectivity assumption, use the TB code without providing any selectivity information (s_list = NA) or by providing a knife-edge selection list (s_list = knife-edge).

To prepare a knife-edge selection list using the LC_{50} from the catch curve analysis, use the following code:

```
knife_edge <- list(selecType = "knife_edge", L50 = CC$L50)
```

To perform the TB1 analysis with the cohort analysis derived FMs under the knife-edge selection assumption, use the following code:

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = NA, plot = TRUE)
```

Alternatively,

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = knife_edge, plot = TRUE)
```

Step-5: Perform TB1 with the cohort analysis derived FMs under the gear selection assumption

Ensure that the my_data contains the single annual FM derived from the catch curve analysis. Provide selectivity information (s_list = gear_selection) in the TB code to ensure the function follows the gear selectivity assumption instead of the knife-edge selection assumption. The use of length class-specific FMs from the cohort analysis effectively overrides any other selectivity information provided through the 's_list = gear_selection' in the TB code. Nevertheless, the 's_list = gear_selection' is essential for the gear selection assumption without which the code follows a knife-edge selection assumption.

Prepare a gear selection list (gear_selection) using the LC_{50} and LC_{75} information derived from a catch curve analysis with the following code:

```
gear_selection <- list(selecType = "trawl_ogive", L50 = CC$L50, L75 = CC$L75)
```

To perform the TB1 analysis with the cohort analysis derived FMs under the gear selection assumption, use the following code:

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = gear_selection, plot = TRUE)
```

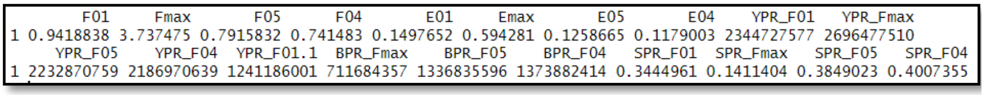
Note: *In the above-mentioned TB1 codes,* ***type*** *represents the Thompson and Bell analysis* ***(type = "ThompBell")****. For the Beverton and Holt analysis, the user can use* ***type = "ypr"****, after providing the required parameters for the analysis. The FM range, over which the effect of a change in the fishing mortality rates (FM) on stock status is evaluated, is supplied in FM_change* ***[e.g., FM_change = seq(0, 5, 0.01)]****. The FM range could be absolute* ***(FM_relative = FALSE)*** *or relative* ***(FM_relative = TRUE)*** *in the Thompson and Bell analysis. In this example, the FM changes are in relative format* ***(FM_relative = TRUE)****, simulating the effect of changes in FM (F-multiplier, not the absolute F) from 0 to 5 with a gradual increase of 0.01 (step size 0.01 indicates a sequential FM change of 0, 0.01, 0.02, ..., 4.99, and 5.00 times the actual F). If the FM changes are in absolute format* ***(FM_relative = FALSE)****, it will simulate the effect of changes in FM (actual F, not the F-multiplier) from 0 to 5 with a gradual increase of 0.01 (step size 0.01 indicates a sequential FM change (actual F) of 0, 0.01, 0.02, ..., 4.99, and 5.00). The user can also use an equivalent range of absolute exploitation rates (E_change) instead of the fishing mortality rate (FM_change) to simulate the same effect. However, if E_change is used instead of FM_change, the range is capped at E = 0.9, as higher values of E correspond to unrealistically high fishing mortality rates. The recruitment number is supplied in* ***stock_size_1*** *in the Thompson and Bell analysis. In this example, the number of survivors in the first (smallest) length group from the cohort analysis has been used as the recruitment number* ***[e.g., stock_size_1 = as.numeric ((cohortanalysis $survivors_L1[1])/1000)]****. If no value is provided* ***(stock_size_1 = NA)****, a default recruitment number of 1000 recruits is used for the simulation. The current exploitation rate* ***(curr.E)*** *or the current exploitation rate* ***(curr.E)*** *or both can be provided to obtain management reference points. However, the user can supply the calculated E (i.e., F/Z)* ***[e.g., curr.E = (as.numeric(CC$FM)/CC$Z)]*** *and/or LC50 values* ***[e.g., curr.Lc = CC$L50]*** *from the catch curve analysis to assess the current stock status. The default graphical output from the simulation can be turned on* ***(plot = TRUE)*** *or off* ***(plot = FALSE)****.*

2.9.3. Biological reference points from TB1

Use the following code to get the biological reference points:

```
TB1$df_Es
```

e.g., when the catch curve analysis derived FM is used with the gear selection assumption



```
        F01      Fmax       F05       F04       E01      Emax       E05       E04    YPR_F01    YPR_Fmax
1 0.9418838 3.737475 0.7915832 0.741483 0.1497652 0.594281 0.1258665 0.1179003 2344727577 2696477510
     YPR_F05    YPR_F04  YPR_F01.1  BPR_Fmax     BPR_F05    BPR_F04   SPR_F01  SPR_Fmax   SPR_F05   SPR_F04
1 2232870759 2186970639 1241186001 711684357 1336835596 1373882414 0.3444961 0.1411404 0.3849023 0.4007355
```

Note: *The stock status and the management reference points may differ under these two different assumptions. Similarly, the stock status and the management reference points may differ when TB1 is performed with the catch curve analysis derived FMs compared to the cohort analysis derived FMs. The yield (YPR) and biomass (BPR) values obtained from this analysis are not the relative yield (yield per recruit, YPR) or biomass per recruit (biomass per recruit, BPR). In fact, these are absolute values raised to the entire stock size (YPR × recruits and BPR × recruits), as the recruitment number from the cohort analysis* ***[stock_size_1 = as.numeric ((cohortanalysis$ survivors_L1[1])/1000)]*** *has been used in the analysis. In the above TB1 code, the stock size of the smallest length class from the cohort analysis* ***[stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1]) /1000)]*** *has been used as the recruitment number to express the stock estimates in absolute terms. Therefore, the cohort analysis should be performed prior to the TB analysis to derive absolute stock estimates. If no recruitment number is provided in the code* ***[stock_size_1 = NA],*** *then the analysis derives relative stock estimates (YPR and BPR) assuming a default stock size (recruitment) of 1,000 individuals. The yield and biomass values have been expressed in grams. Divide these values by 1,000,000 to express them in tonnes. To learn more about other management reference points (e.g., F_{01}, F_{max}, F_{04}, F_{05}, SPR, etc.), refer to* ***2.11.Understanding the stock simulation outputs (fisheries management reference points)****'.*

2.9.4. Current stock status estimates from TB1

To get current estimates for yield, biomass, spawning stock biomass, SPR, and revenue levels, user need to provide either the current exploitation rate (curr.E=) or the current length at capture (curr.Lc=) or both the values in the above mentioned TB1 codes. Use the calculated E (i.e., F/Z) [e.g., curr.E= (as.numeric(CC$FM)/CC$Z)] and LC_{50} values [e.g., curr.Lc =CC$L50] from the catch curve analysis in TB1 code as follows:

To perform the TB1 analysis under the knife-edge selection assumption, use the following codes:

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = knife_edge, plot = TRUE)
```

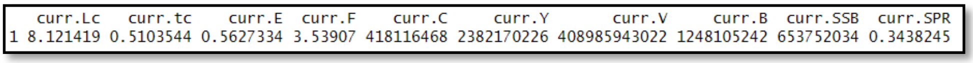
To perform the TB1 analysis under the gear selection assumption, use the following codes:

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = gear_selection, plot = TRUE)
```

After entering the values for the current exploitation rate and the current length at capture in the TB1 code, use the following code to retrieve the current values:

```
TB1$currents
```

e.g., when the catch curve analysis derived FM is used with the gear selection assumption

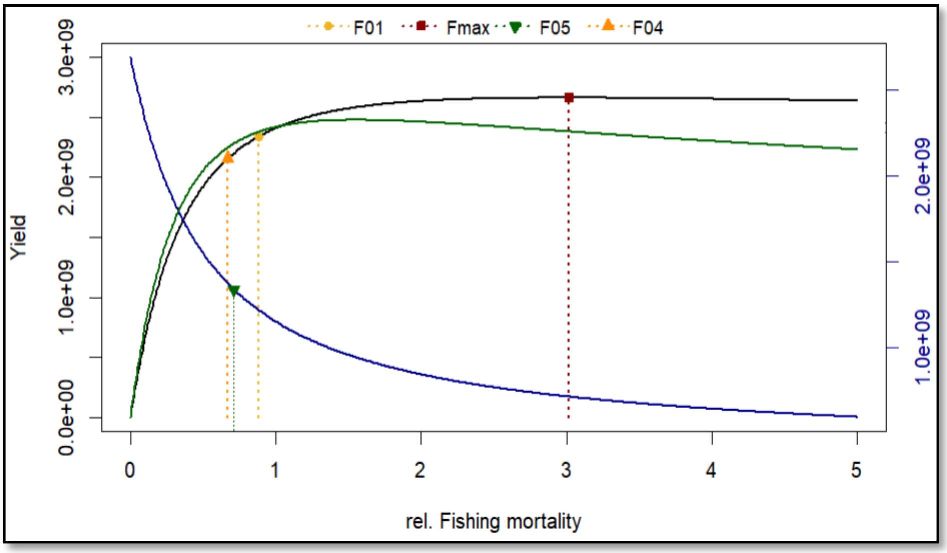


```
   curr.Lc   curr.tc    curr.E  curr.F    curr.C     curr.Y       curr.V     curr.B  curr.SSB  curr.SPR
1 8.121419 0.5103544 0.5627334 3.53907 418116468 2382170226 408985943022 1248105242 653752034 0.3438245
```

> ***Note:*** *The current stock estimates, such as yield (curr.Y), value (curr.V), biomass (curr.B), and spawning stock biomass (curr.SSB), obtained from this analysis are not relative estimates per recruit. In fact, these are absolute values raised to the entire stock size, as the recruitment number from the cohort analysis [stock_size_1 =* ***as.numeric ((cohortanalysis$ survivors_L1[1])/1000)****] has been used for the analysis. The yield, biomass and SSB values have been expressed in grams. Divide these values by 1,000,000 to express them in tonnes. The revenue (economic value of the yield) should also be divided by 1,000,000,000 to convert it to million INR. For more information on these and other management reference points, refer to* ***2.11.Understanding the stock simulation outputs (fisheries management reference points)****'*

2.9.5. Default graphical output from TB1

The default graphical output from the simulation can be turned on by mentioning plot = TRUE in the TB1 code, which will produce the following graph.



2.9.6. Simulate the combined effect of changes in FM and LC$_{50}$ on stock status (TB2)

This approach is used to simulate the combined effect of simultaneous changes in fishing mortality rate (FM) and the length at capture or selectivity (LC$_{50}$) on the stock status.

Step-1: Define the range of LC (length at capture)

Define the range of LC (length at capture) over which the effect is evaluated.

```
LC_min<- min(my_data$midLengths)
LC_max<- max(my_data$midLengths)
```

> ***Note:*** *The LC range provided in this example spans from the minimum to the maximum observed length. Users have the flexibility to specify any other length range.*

Step-2: Assign the fishing mortality rates (FMs)

The length class-wise fishing mortality rates are required for the TB analysis, which can be derived using either of the following two approaches:

Catch curve analysis derived FM

This approach uses the single annual fishing mortality rate (FM) and gear selectivity information obtained from a catch curve analysis to derive the length class-wise fishing mortality rates (FMs). First perform a catch curve analysis for the specified combination of years (e.g., all available years using mean_catch_vec) using the following code to get the probability of capture, which is then used to derive the length class-wise fishing mortality rates (FMs).

```
CC <- catchCurve(mean_catch_vec, catch_columns = 1, calc_ogive = TRUE)
```

Assign the annual fishing mortality rate (FM) for the stock, derived from the catch curve analysis, to the my_data.

```
my_data$FM<-as.numeric(CC$FM)
```

Cohort analysis derived FMs

This approach uses the length class-wise fishing mortality rates (FMs) derived directly from a cohort analysis.

First, perform a cohort analysis for the specified combination of years (e.g., all available years using the **mean_catch_vec**) to derive the length class-wise fishing mortality rates (FMs).

```
cohortanalysis <- VPA(param = mean_catch_vec, catch_columns = 1, catch_unit=
"'000", terminalE = 0.5, analysis_type = "CA", plot= TRUE)
```

Assign the length class-wise fishing mortality rates (FMs) generated from the cohort analysis to the **my_data** using the following code:

```
my_data$FM<- cohortanalysis$FM_calc
```

> ***Note:*** *This is a very crucial step in the analysis. The FMs can be derived either through the cohort analysis or a catch curve analysis. The above example shows the procedure for deriving FMs for a specific combination of years (e.g., all available years using **mean_catch_vec**) using the length converted catch curve analysis or the cohort analysis. Alternatively, the user can derive FMs for any specified year using the **catch_vec** in the length converted catch curve analysis or the cohort analysis by specifying the serial number of the required year in the catch column of the respective codes **(catch_columns =)**. To perform catch curve analysis for the required combination of years, refer to the '**2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data**'. To perform catch curve analysis for the required year, refer to the '**2.7.3.Catch curve analysis for a specific year**'. To perform cohort analysis for the required combination of years, refer to the '**2.8.2.Cohort analysis for a combination of years using multiyear mean catch data**'. To perform cohort analysis for the required year, refer to the '**2.8.3.Cohort analysis for a specific year**'.*

Step-3: Perform TB1 with catch curve analysis derived FMs under the knife-edge selection assumption

Ensure that the my_data contains the single annual FM derived from a catch curve analysis. To ensure the function follows the knife-edge selection assumption instead of the gear selectivity assumption, use the TB code by providing a knife-edge selection list (s_list = knife-edge).

To prepare a knife-edge selection list using the LC_{50} from a catch curve analysis, use the following code:

```
knife_edge <- list(selecType = "knife_edge", L50 = CC$L50)
```

To perform the TB2 analysis with a catch curve analysis derived FM under the knife-edge selection assumption, use the following code:

```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = knife_edge, plot = TRUE)
```

Step-4: Perform TB2 with a catch curve analysis derived FM under the gear selection assumption

Ensure that the my_data contains the single annual FM derived from the catch curve analysis. Provide selectivity information (s_list = gear_selection) in the TB code to ensure the function follows the gear selectivity assumption instead of the knife-edge selection assumption. Prepare a gear selection list (gear_selection) using the LC_{50} and LC_{75} information derived from a catch curve analysis with the following code:

```
gear_selection <- list(selecType = "trawl_ogive", L50 = CC$L50, L75 = CC$L75)
```

To perform the TB2 analysis with a catch curve analysis derived FM under the gear selection assumption, use the following code:

```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = gear_selection, plot = TRUE)
```

Step-5: Perform TB2 with the cohort analysis derived FMs under the knife-edge selection assumption

Ensure that the my_data includes the length class-specific FMs derived from a cohort analysis. To ensure the function follows the knife-edge selection assumption instead of the gear selectivity assumption, use the TB code without providing any selectivity information (s_list = NA) or by providing a knife-edge selection list (s_list = knife-edge).

To prepare a knife-edge selection list using the LC_{50} from a catch curve analysis, use the following code:

```
knife_edge <- list(selecType = "knife_edge", L50 = CC$L50)
```

To perform the TB2 analysis with the cohort analysis derived FMs under the knife-edge selection assumption, use the following code:

```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = NA, plot = TRUE)
```

Alternatively,

```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = knife_edge, plot = TRUE)
```

Step-6: Perform TB2 with the cohort analysis derived FMs under the gear selection assumption

Ensure that the my_data contains the single annual FM derived from the catch curve analysis. Provide selectivity information (s_list = gear_selection) in the TB code to ensure the function follows the gear selectivity assumption instead of the knife-edge selection assumption. Using the length class-specific FMs from the cohort analysis effectively overrides any other selectivity information provided through the 's_list = gear_selection' in the TB code. Nevertheless, the 's_list = gear_selection' is essential for the gear selection assumption, without which the code follows a knife-edge selection assumption.

Prepare a gear selection list (gear_selection) using the LC_{50} and LC_{75} information derived from a catch curve analysis with the following code:

```
gear_selection <- list(selecType = "trawl_ogive", L50 = CC$L50, L75 = CC$L75)
```

To perform the TB2 analysis with cohort analysis derived FMs under the gear selection assumption, use the following code:

```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E = NA, curr.Lc = NA, s_list = gear_selection, plot = TRUE)
```

Note: *In the above TB2 codes,* ***type*** *represents the Thompson and Bell analysis* ***(type = "ThompBell")****. For the Beverton and Holt analysis, the user can use* ***type = "ypr"****, after providing the required parameters for the analysis. The FM range, over which the effect of a change in the fishing mortality rates (FM) on stock status is evaluated, is supplied in* ***FM_change [e.g., FM_change = seq(0, 20, 0.5)]****. The FM range could be absolute* ***(FM_relative = FALSE)*** *or relative* ***(FM_relative = TRUE)*** *in the Thompson and Bell analysis. In this example, the FM changes are in absolute format* ***(FM_relative = FALSE)****, simulating the effect of changes in FM (absolute F, not the F-multiplier) from 0 to 20 with a gradual increase of 0.5 (step size 0.5 shows a sequential FM change of 0, 0.5, 1.0, ..., 19.5, and 20.0). If the FM changes are in relative format* ***(FM_relative = TRUE)****, it will simulate the effect of changes in FM (F-multiplier, not the absolute F) from 0 to 20 with a gradual increase of 0.5 (step size 0.5 indicates a sequential FM change of 0, 0.5, 1.0, ..., 19.5, and 20.0 times the actual F). The user can also use an equivalent range of absolute exploitation rates (E_change) instead of the fishing mortality rate (FM_change) to simulate the same effect. However, if E_change is used instead of FM_change, the range is capped at E = 0.9, as higher values of E correspond to unrealistically high fishing mortality rates.*

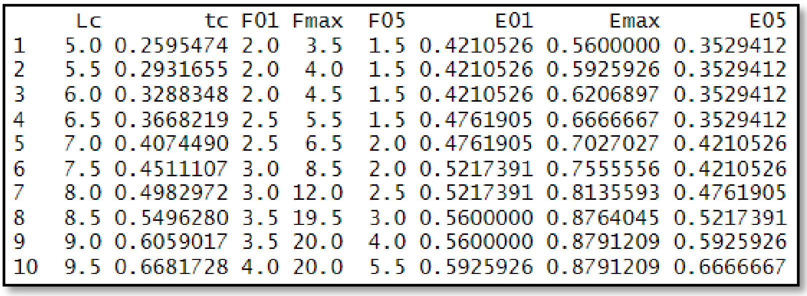
In addition to the FM, the used can simulate the effect of change in length at capture (LC) by providing a LC range in ***Lc_change [Lc_change = seq(LC_min, LC_max, 0.5)]****. The LC range provided in this example ranges from the minimum to the maximum observed length with a gradual increase of 0.5 (step size 0.5 indicates a sequential LC change of 5, 5.5, 1.0, …, 13.0, and 13.5). Users have the flexibility to specify any other length range and step size. The recruitment number is supplied in* ***stock_size_1*** *in the Thompson and Bell analysis. In this example, the number of survivors in the first (smallest) length group from the cohort analysis has been used as the recruitment number [e.g., stock_size_1 =* ***as.numeric ((cohortanalysis $survivors_L1[1])/1000)****]. If no value is provided* ***(stock_size_1 = NA)****, a default recruitment number of 1000 recruits is used for the simulation. The current exploitation rate* ***(curr.E)*** *or the current exploitation rate* ***(curr.E)*** *or both the values are required to obtain management reference points. However, the user can supply the calculated E (i.e., F/Z) [e.g., curr.E =* ***(as.numeric(CC$FM)/CC$Z)****] and LC50 values [e.g., curr.Lc =* ***CC$L50****] from the catch curve analysis to assess the current stock status. The default graphical output from the simulation can be turned on* ***(plot = TRUE)*** *or off* ***(plot = FALSE)****.*

2.9.7. Biological reference points from TB2

Use the following code to derive the biological reference points from TB2 analysis:

```
TB2$df_Es
```

When a catch curve analysis derived FM is used with the gear selection assumption



```
     Lc        tc F01 Fmax  F05       E01      Emax       E05
1   5.0 0.2595474 2.0  3.5  1.5 0.4210526 0.5600000 0.3529412
2   5.5 0.2931655 2.0  4.0  1.5 0.4210526 0.5925926 0.3529412
3   6.0 0.3288348 2.0  4.5  1.5 0.4210526 0.6206897 0.3529412
4   6.5 0.3668219 2.5  5.5  1.5 0.4761905 0.6666667 0.3529412
5   7.0 0.4074490 2.5  6.5  2.0 0.4761905 0.7027027 0.4210526
6   7.5 0.4511107 3.0  8.5  2.0 0.5217391 0.7555556 0.4210526
7   8.0 0.4982972 3.0 12.0  2.5 0.5217391 0.8135593 0.4761905
8   8.5 0.5496280 3.5 19.5  3.0 0.5600000 0.8764045 0.5217391
9   9.0 0.6059017 3.5 20.0  4.0 0.5600000 0.8791209 0.5925926
10  9.5 0.6681728 4.0 20.0  5.5 0.5925926 0.8791209 0.6666667
```

Note: *The output demonstrates the effect of changes in length at capture (LC$_{50}$) or age at capture (tc$_{50}$) on biological management reference points (e.g., fishing mortality rates: F_{01}, F_{max} and F_{05} or exploitation rates: E_{01}, E_{max} and E_{05}). For more information on these and other management reference points, refer to* ***Understanding the stock simulation outputs (Management Reference Points)****.*

2.9.8. Current stock status estimates from TB2

To get current estimates for yield, biomass, spawning stock biomass, SPR, and revenue levels, the user needs to provide either only the current exploitation rate (**curr.E=**) or both the current exploitation rate (**curr.E=**) and the current length at capture (**curr.Lc=**) in the above mentioned TB code. Use the calculated E (i.e., F/Z) [e.g., curr.E = **(as.numeric(CC$FM)/CC$Z)**] and LC$_{50}$ values [e.g., curr.Lc = **CC$L50**] from the catch curve analysis in TB code as follows:

To perform the TB2 analysis under the knife-edge selection assumption, use the following codes:

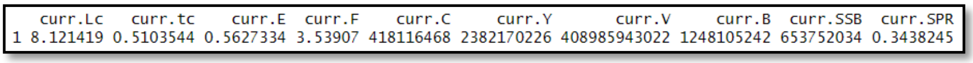
```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = knife_edge, plot = TRUE)
```

To perform the TB2 analysis under the gear selection assumption, use the following codes:

```
TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), E_change = NA, FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = gear_selection, plot = TRUE)
```

After entering the values for the current exploitation rate and the current length at capture in the TB1 code, use the following code to retrieve the current values:

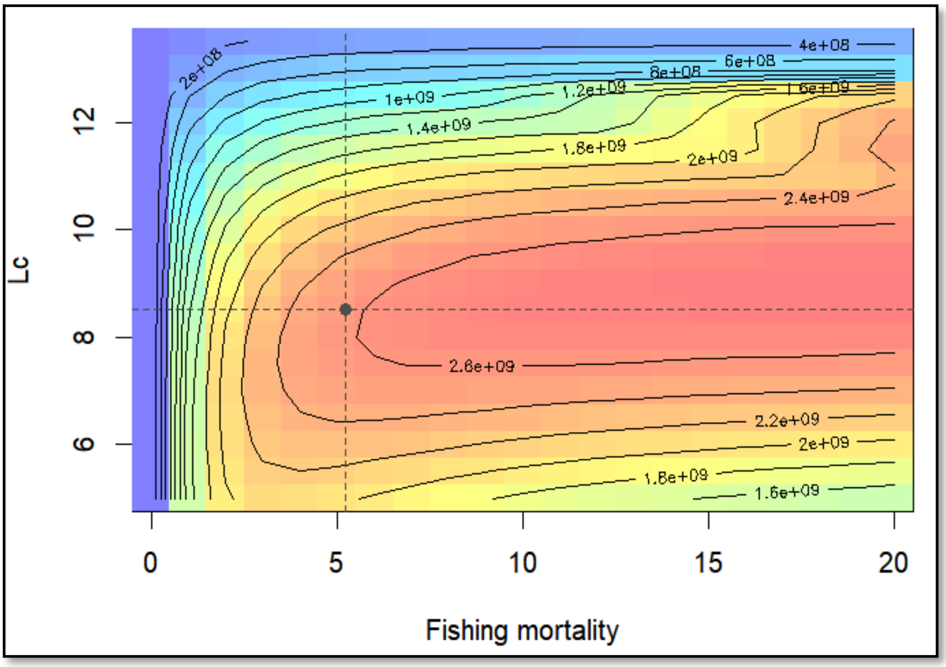
```
TB2$currents
```



```
    curr.Lc   curr.tc    curr.E  curr.F    curr.C     curr.Y       curr.V     curr.B  curr.SSB  curr.SPR
1 8.121419 0.5103544 0.5627334 3.53907 418116468 2382170226 408985943022 1248105242 653752034 0.3438245
```

2.9.9. Default graphical output from TB2

The default graphical output from the simulation can be turned on by mentioning **plot = TRUE** in the TB2 code, which will produce the following graph.



Note: *The current stock estimates, such as yield (curr.Y), value (curr.V), biomass (curr.B), and spawning stock biomass (curr.SSB), obtained from this analysis are not relative estimates per recruit. In fact, these are absolute values raised to the entire stock size, as the recruitment number from the cohort analysis [stock_size_1 =* ***as.numeric((cohortanalysis$ survivors_L1[1])/1000)****] has been used for the analysis. The yield, biomass and SSB values have been expressed in grams. Divide these values by 1,000,000 to express them in tonnes. The revenue (economic value of the yield) should also be divided by 1,000,000,000 to convert it to million INR. For more information on these and other management reference points, refer to* ***2.11.Understanding the stock simulation outputs (fisheries management reference points)****'.*

2.9.10. Enhanced visualization of Thompson and Bell prediction model

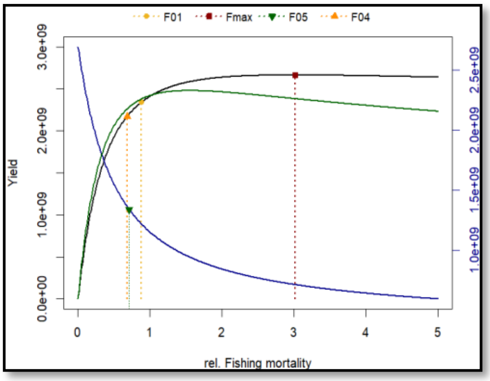
Plotting the effect of a change in FM on stock status (TB1 graph)

The previously mentioned TB1 code when used with plot option set as true (plot=true) produces a default graphical output which may not produce a complete visualization of all the outputs, especially when revenue outputs (economic value of the yield) are plotted on a secondary Y-axis. It happens mainly because of the non-availability of adequate space on the right side margin. As a default, R uses a margin setup of par(mar=c(5, 4, 4, 2)), which is 5, 4, 4 and 2 lines spaces for the bottom, left, top and right side margins, respectively. To accommodate adequate space for the right side margin for plotting the extra Y-axis, use the required TB1 code after setting the margin space using the following code:

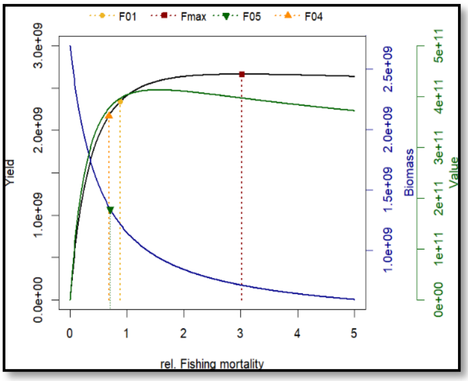
```
par(mar=c(5, 4, 4, 7))
```

```
TB1 <- predict_mod (my_data, type = "ThompBell", FM_change = seq(0,5,0.01), E_change = NA, FM_relative = TRUE, stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = gear_selection, plot = TRUE)
```

Default graphical output with par(mar=c(5, 4, 4, 2))



Better graphical output with par(mar=c(5, 4, 4, 7))



After plotting is over, set the default margin of R using the following code:

```
par(mar=c(5, 4, 4, 2))
```

Including the required SPR threshold as precautionary management reference point in TB1 graph

The default graphical output of the TB1 depicts the 40% SPR level (F04) as the default minimum threshold level for the SPR. The actual required threshold level of SPR depends on the regenerative/reproductive capacity (resilience) of the species. In order to use any other required minimum SPR threshold level, the TB1 output can be manually plotted using the following steps.

Step-1: After completing the TB1 analysis, create a data frame by multiplying the yield, biomass, and SSB values generated from TB1 by 0.000001 to express them in tonnes, and multiplying the revenue (values) by 0.000000001 to express it in million INR.

```
TB1_output<-data.frame(F=TB1$FM_change, E=TB1$E_change,
Catch=TB1$totC*0.000001, Yield=TB1$totY*0.000001,
Revenue=TB1$totV*0.000000001, Biomass=TB1$meanB*0.000001,
SSB=TB1$meanSSB*0.000001, SPR=TB1$SPR)
```

Step-2: Prepare a secondary Y-axis for plotting the revenue using the following code:

```
scale<-max(TB1_output$Yield)/ max(TB1_output$Revenue)
```

Step-3: Plot the TB1 outputs (2D plot) using the following code:

```
library(ggplot2)
```

```
fig1<- ggplot(TB1_output) + geom_line(aes(x=F, y=Yield), stat="identity", color="blue",
linewidth=1)+ geom_line(aes(x=F, y=Biomass), stat="identity", color="red", linewidth=1)+
geom_line(aes(x=F, y=SSB), stat="identity", color="darkorange",
linewidth=1)+geom_line(aes(x=F, y= Revenue*scale), stat="identity", color="darkgreen",
linewidth=1) + scale_x_continuous(expand = c(0,0), breaks =
seq(0,max(TB1_output$F),0.5)) + scale_y_continuous(expand = c(0,0),
sec.axis=sec_axis(~./scale, name="Revenue (million INR)"))+ xlab("F-multiplier") +
ylab("Yield, Biomass and SSB (tonnes)")+ theme_classic(base_size = 12)
```

Step-4: Add management reference points using the following code:

```
fig2<- fig1+geom_point(aes(x=TB1$df_Es$Fmax, y=TB1$df_Es$YPR_Fmax*0.000001),
size=2, shape = 25, fill="blue")+geom_point(aes(x=TB1$df_Es$F01,
y=TB1$df_Es$YPR_F01*0.000001), size=3, shape = 23,
fill="blue")+geom_point(aes(x=TB1$df_Es$F05, y=TB1$df_Es$BPR_F05*0.000001),
size=3, shape = 21, fill="red")
```

Step-4: Add the required SSB level as a management reference point using the following code:

First, extract the reference values of F and SSB based on the required SPR percentage (e.g., 25%) to plot the SSB reference point.

```
library(dplyr)
```

```
my_SSB_percentage<-tail (filter (TB1_output, SPR>0.25), n=1)
```

Note: *The Thompson and Bell analysis (TB1) by default produces an SPR (Spawning Potential Ratio, SSB_0/SSB) of 40% (i.e., $F_{0.4}$). The code provided above plots the SPR at the 25% level.*

*However, the user can adjust the SPR threshold by changing the condition (e.g., **SPR > 0.30** for a 30% SPR level, or **SPR > 0.35** for a 35% SPR level) in place of the **SPR > 0.25** example in the above code (Step-4).*

```
fig3<-fig2+geom_point (aes(x=my_SSB_percentage$F, y=my_SSB_percentage$SSB),
size=3, shape = 21, fill="darkorange")
```

Step-5: Add the maximum revenue level as a management reference point using the following code:

```
Revenue_max<-filter (TB1_output, Revenue==max(Revenue))
```

```
fig4<-fig3+geom_point (aes(x= Revenue_max$F, y= Revenue_max$Revenue*scale),
size=3, shape = 21, fill=" darkgreen")
```

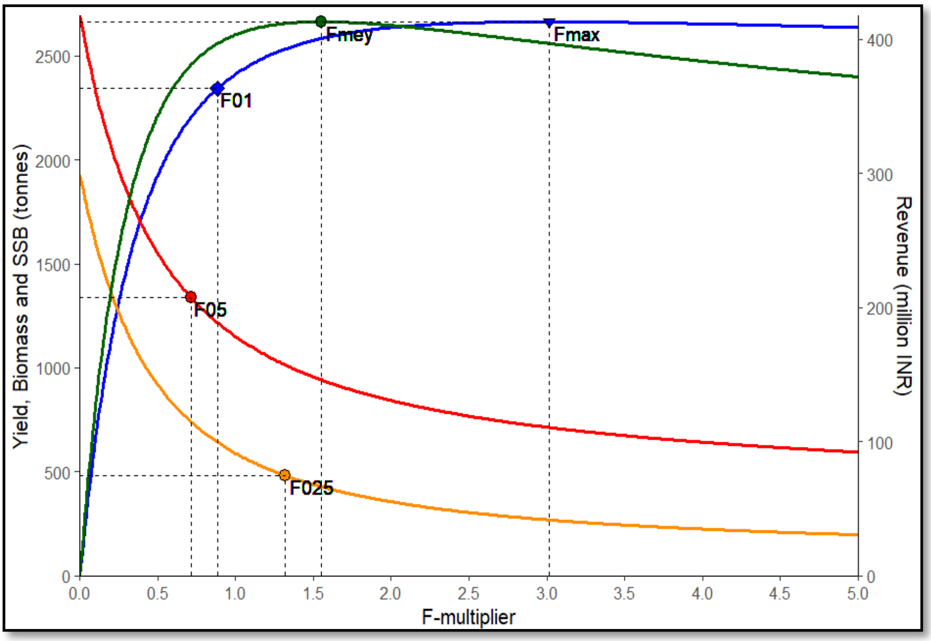
Step-6: Add management reference lines using the following code:

```
fig5<-fig4+geom_segment(aes(x=0, y= TB1$df_Es$YPR_Fmax*0.000001), xend=
TB1$df_Es$Fmax, yend= TB1$df_Es$YPR_Fmax*0.000001, linetype="dashed") +
geom_segment(aes(x= TB1$df_Es$Fmax, y=0.0), xend= TB1$df_Es$Fmax, yend=
TB1$df_Es$YPR_Fmax*0.000001, linetype="dashed") + geom_segment(aes(x=0, y=
TB1$df_Es$YPR_F01*0.000001), xend= TB1$df_Es$F01, yend=
TB1$df_Es$YPR_F01*0.000001, linetype="dashed") + geom_segment(aes(x=
TB1$df_Es$F01, y=0.0), xend= TB1$df_Es$F01, yend= TB1$df_Es$YPR_F01*0.000001,
linetype="dashed") + geom_segment(aes(x=0, y= TB1$df_Es$BPR_F05*0.000001),
xend= TB1$df_Es$F05, yend= TB1$df_Es$BPR_F05*0.000001, linetype="dashed") +
geom_segment(aes(x= TB1$df_Es$F05, y=0.0), xend= TB1$df_Es$F05, yend=
TB1$df_Es$BPR_F05*0.000001, linetype="dashed") + geom_segment(aes(x=0, y=
my_SSB_percentage$SSB), xend= my_SSB_percentage$F, yend=
my_SSB_percentage$SSB, linetype="dashed") + geom_segment(aes(x=
my_SSB_percentage$F, y=0.0), xend= my_SSB_percentage$F, yend=
my_SSB_percentage$SSB, linetype="dashed")+ geom_segment(aes(x=0, y=
Revenue_max$Revenue*scale), xend= Revenue_max$F, yend=
Revenue_max$Revenue*scale, linetype="dashed") + geom_segment(aes(x=
Revenue_max$F, y=0.0), xend= Revenue_max$F, yend= Revenue_max$Revenue*scale,
linetype="dashed")
```

Step-7: Add management reference labels using the following code:

```
fig6<-fig5+geom_text(aes(x=TB1$df_Es$Fmax, y=TB1$df_Es$YPR_Fmax*0.000001),
label="Fmax", size=4, hjust=-0.1, vjust=1.2) + geom_text (aes(x=TB1$df_Es$F01,
y=TB1$df_Es$YPR_F01*0.000001), label="F01", size=4, hjust=-0.1, vjust=1.2) +
geom_text(aes(x=TB1$df_Es$F05, y=TB1$df_Es$BPR_F05*0.000001), label="F05",
size=4, hjust=-0.1, vjust=1.2) + geom_text(aes(x=my_SSB_percentage$F,
y=my_SSB_percentage$SSB), label="F025", size=4, hjust=-0.1, vjust=1.2) +
geom_text(aes(x=Revenue_max$F, y=Revenue_max$Revenue*scale), label="Fmey",
size=4, hjust=-0.1, vjust=1.2)
```

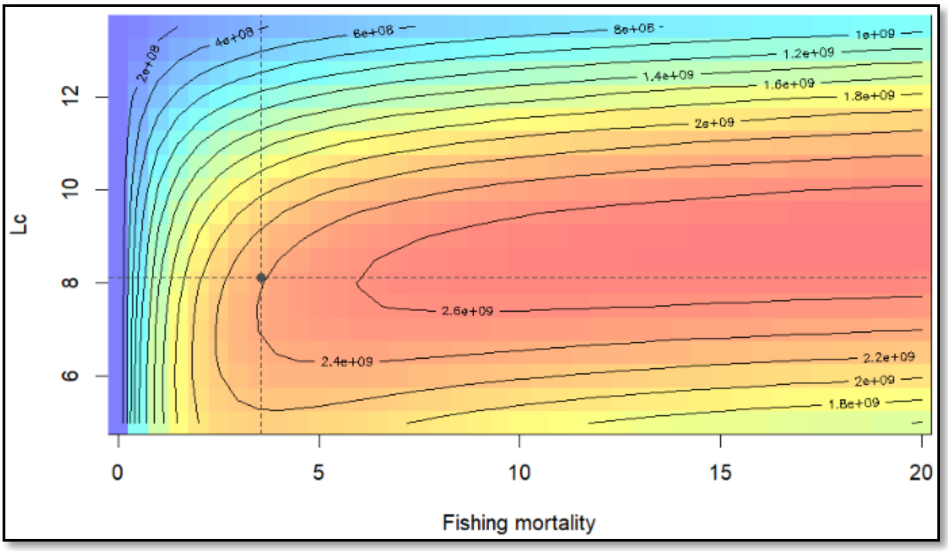
***Note**: Change the label (e.g., label="F025") in the above code (Step-7) based on the defined SPR threshold level. For example, for a 30% SPR level, use label="F030"; for a 35% SPR level, use label="F035".*



Plotting the effect of change in FM and LC on stock status (TB2 graph)

The previously mentioned TB2 code, when used with plot option set as true (plot=true) produces a default graphical output which depicts only the yield isopleth.

TB2 <- predict_mod(my_data, type = "ThompBell", FM_change = seq(0,20,0.5), FM_relative=FALSE, Lc_change = seq(LC_min, LC_max, 0.5), stock_size_1 = as.numeric((cohortanalysis$ survivors_L1[1])/1000), curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = gear_selection, **plot = TRUE**)

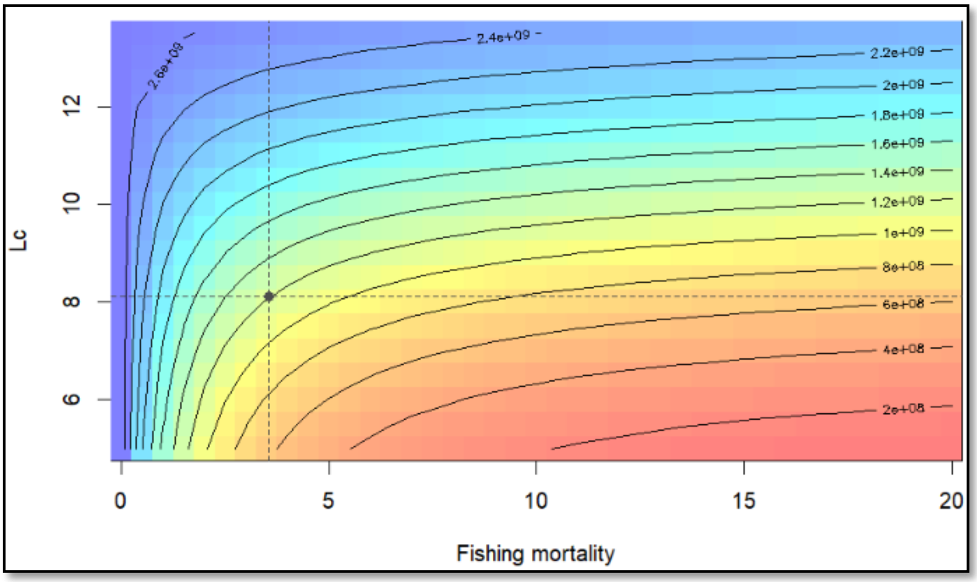


Once the TB2 outputs are generated, the yield isopleths can also be plotted using the following code:

```
plot(TB2, xaxis1 = "FM", yaxis_iso = "Lc", yaxis1 = "Y_R", mark = TRUE, identify = FALSE)
```

Once the TB2 outputs are generated, the biomass isopleths can be ploted using the following code:

```
plot(TB2, xaxis1 = "FM", yaxis_iso = "Lc", yaxis1 = "B_R", mark = TRUE, identify = FALSE)
```



Note: *Identity can be set on (**identity=TRUE**) in the above codes to find out the values of LC and FM for any point on the graph just by clicking on it.*

Enhanced Yield Isopleth visualization from TB2 (2D plot)

Step-1: Prepare a new matrix (Yield_change) for the change in the yield in response to the change in FM and LC from the Thompson and Bell prediction model (TB2). To convert the yield in 'gram' to "tonnes" multiply with 0.000001.

```
Yield_change<- as.matrix((TB2$mat_FM_Lc_com.Y)* 0.000001)
```

Step-2: Arrange the data for plotting using the following code:

```
library(reshape2)
Yield_isopleth_data <-melt(Yield_change, id.vars=c("Var1", "Var2"),
measure.vars="value")
names(Yield_isopleth_data)<-c("F", "LC", "Yield")
```

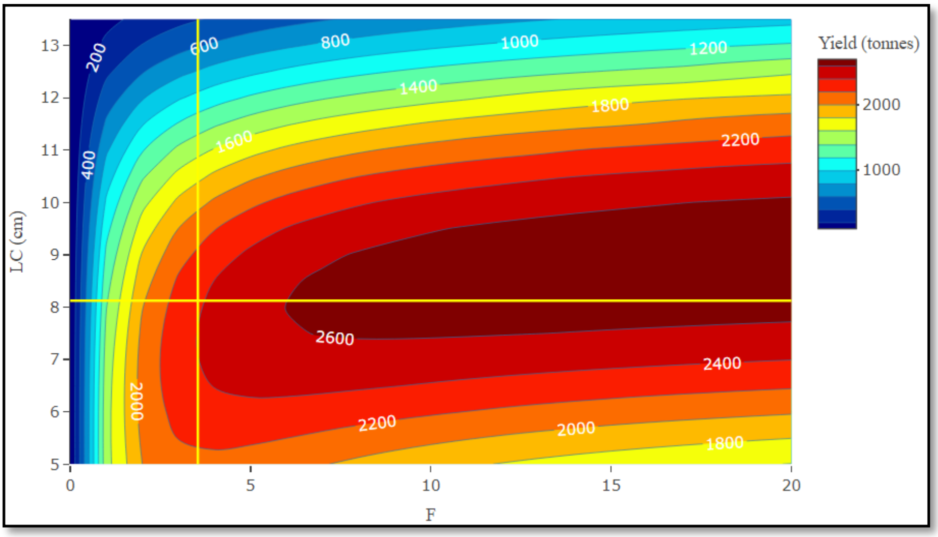
Step-3: Plot the 2D plot using the following code:

```
library(plotly)
```

```
fig<- plot_ly(Yield_isopleth_data, x=~ Yield_isopleth_data$F, y=~
Yield_isopleth_data$LC, z=~ Yield_isopleth_data$Yield) %>% add_trace(type="contour",
contours = list(showlabels = TRUE, labelfont = list(size = 12, color = "white")), colorscale
= "Jet", colorbar=list(title= list(text="Yield (tonnes)", font=list(size="14", family="Times
New Roman"))))%>% layout(xaxis=list(title= list(text= "F", font=list(size="14",
family="Times New Roman"))), yaxis=list(title= list(text= "LC (cm)", font=list(size="14",
family="Times New Roman"))))
```

Step-4: Add current reference lines using the following code:

```
vline <- function(x = 0, color = "red") {
list(type = "line", y0 = 0, y1 = 1, yref = "paper", x0 = x, x1 = x, line = list(color = "yellow",
dash="dash"))
}
hline <- function(y = 0, color = "blue") {
list(type = "line", x0 = 0, x1 = 1, xref = "paper", y0 = y, y1 = y, line = list(color = "yellow",
dash="dash"))
}
#Show the catch curve analysis derived FM and LC on the plot using the following code:
fig%>%layout(shapes = list(vline(as.numeric(CC$FM)), hline(CC$L50)))
```



Note: The colour gradient can be customized by changing the colorscale = "Jet" to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet".

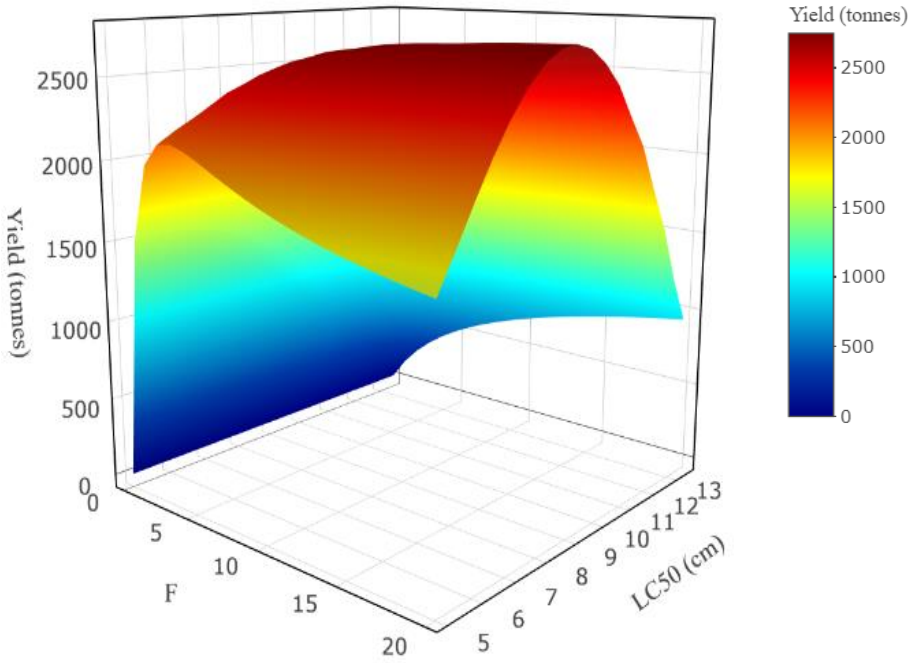
Enhanced Yield Isopleth visualization from TB2 (3D plot)

Step-1: Prepare a new matrix (Yield_change) for the change in the yield in response to the change in FM and LC from the Thompson and Bell prediction model (TB2). To convert the yield in 'gram' to "tonnes" multiply with 0.000001.

```
Yield_change<- as.matrix((TB2$mat_FM_Lc_com.Y)* 0.000001)
```

Step-2: Plot the 3D plot using the following code:

```
library(plotly)
fig<- plot_ly(z= Yield_change, x = ~ as.numeric(colnames(Yield_change)), y = ~
as.numeric(rownames(Yield_change)), type = "surface", opacity = 1.0, colorscale = "Jet",
colorbar=list(title= list(text= "Yield (tonnes)", font=list(size="14", family="Times New
Roman"))))%>% layout(scene=list(xaxis=list(autorange = "reversed", nticks = 10,
tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="LC50
(cm)", font=list(size="14", family="Times New Roman"))), yaxis=list(nticks = 10,
tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="F",
font=list(size="14", family="Times New Roman"))), zaxis=list(nticks = 10, tickangle= 0,
linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="Yield (tonnes)",
font=list(size="14", family="Times New Roman")))))
```



Note: *The colour gradient can be customized by changing the colorscale = "Jet" to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet".*

Enhanced Biomass Isopleth Visualization from TB2 (2D plot)

Step-1: Prepare a new matrix (Biomass_change) for the change in the biomass in response to the change in FM and LC from the Thompson and Bell prediction model (TB2). To convert the yield in 'gram' to "tonnes" multiply with 0.000001.

```
Biomass_change<- as.matrix((TB2$mat_FM_Lc_com.B)* 0.000001)
```

Step-2: Arrange the data for plotting using the following code:

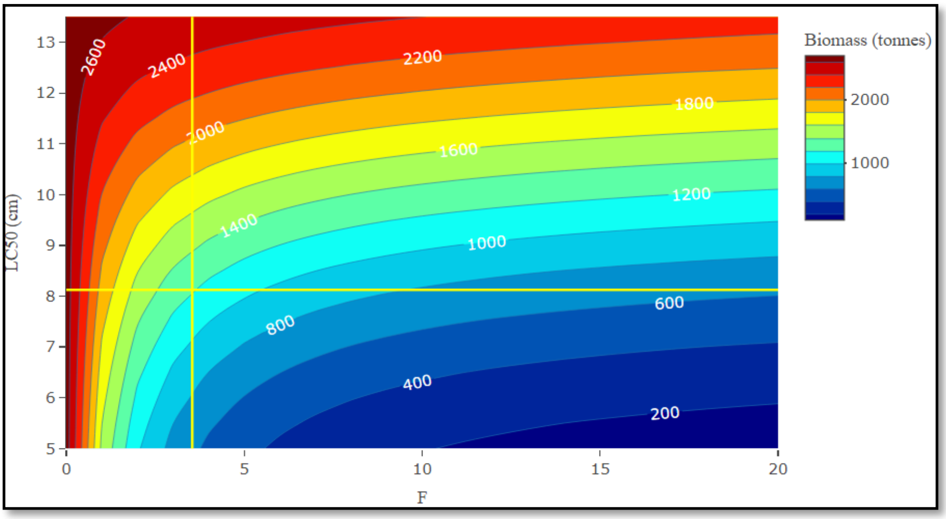
```
library(reshape2)
Biomass_isopleth_data <-melt(Biomass_change, id.vars=c("Var1", "Var2"), measure.vars="value")
names(Biomass_isopleth_data)<-c("F", "LC", "Biomass")
```

Step-3: Plot the 2D plot using the following code:

```
fig<- plot_ly(Biomass_isopleth_data, x=~Biomass_isopleth_data$F, y=~Biomass_isopleth_data$LC, z=~Biomass_isopleth_data$Biomass) %>% add_trace(type="contour", contours = list(showlabels = TRUE, labelfont = list(size = 12, color = "white")), colorscale = "Jet", colorbar=list(title= list(text="Biomass (tonnes)", font=list(size="14", family="Times New Roman"))))%>% layout(xaxis=list(title= list(text= "F", font=list(size="14", family="Times New Roman"))), yaxis=list(title= list(text= "LC50 (cm)", font=list(size="14", family="Times New Roman"))))
```

Step-4: Add current reference lines using the following code:

```
vline <- function(x = 0, color = "red") {
list(type = "line", y0 = 0, y1 = 1, yref = "paper", x0 = x, x1 = x, line = list(color = "yellow", dash="dash"))
}
hline <- function(y = 0, color = "blue") {
list(type = "line", x0 = 0, x1 = 1, xref = "paper", y0 = y, y1 = y, line = list(color = "yellow", dash="dash"))
}
#Show the catch curve analysis derived FM and LC on the plot using the following code:
fig%>%layout(shapes = list(vline(as.numeric(CC$FM)), hline(CC$L50)))
```



Note: *The colour gradient can be customized by changing the colorscale = "Jet" to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet".*

Enhanced Biomass Isopleth Visualization from TB2 (3D plot)

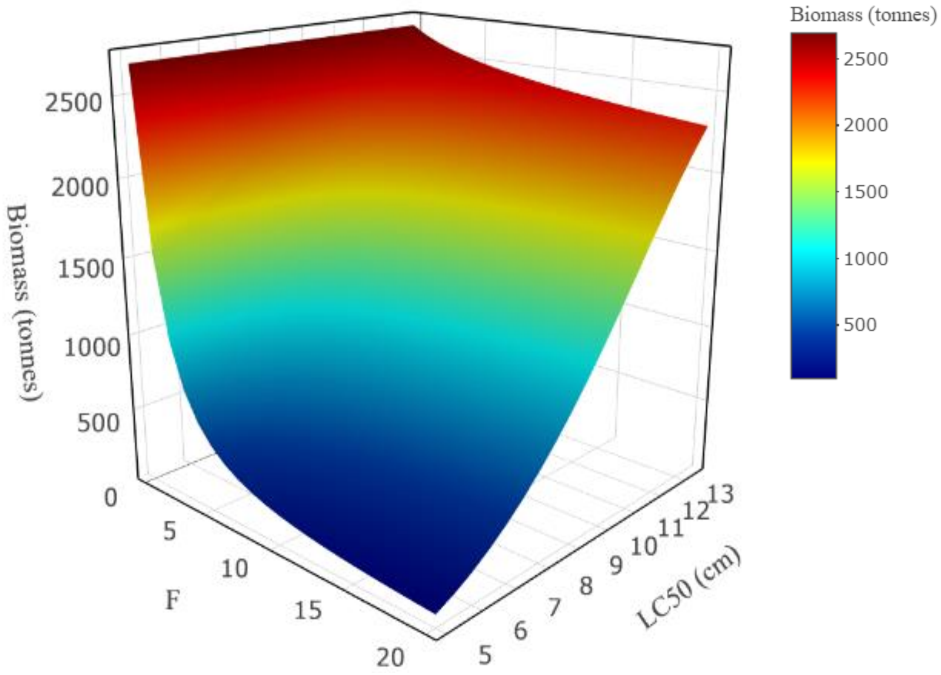
Step-1: Prepare a new matrix (Biomass_change) for the change in the biomass in response to the change in FM and LC from the Thompson and Bell prediction model (TB2). To convert the yield in 'gram' to "tonnes" multiply with 0.000001.

```
Biomass_change<- as.matrix((TB2$mat_FM_Lc_com.B)* 0.000001)
```

Step-2: Plot the 3D plot using the following code:

```
library(plotly)
```

```
fig<- plot_ly(z= Biomass_change, x = ~ as.numeric(colnames(Biomass_change)), y = ~ as.numeric(rownames(Biomass_change)), type = "surface", opacity = 1.0, colorscale = "Jet", colorbar=list(title= list(text= "Biomass (tonnes)", font=list(size="14", family="Times New Roman"))))%>% layout(scene=list(xaxis=list(autorange = "reversed", nticks = 10, tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="LC50 (cm)", font=list(size="14", family="Times New Roman"))), yaxis=list(nticks = 10, tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="F", font=list(size="14", family="Times New Roman"))), zaxis=list(nticks = 10, tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="Biomass (tonnes)", font=list(size="14", family="Times New Roman")))))
```



Note: *The colour gradient can be customized by changing the colorscale = "Jet" to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet".*

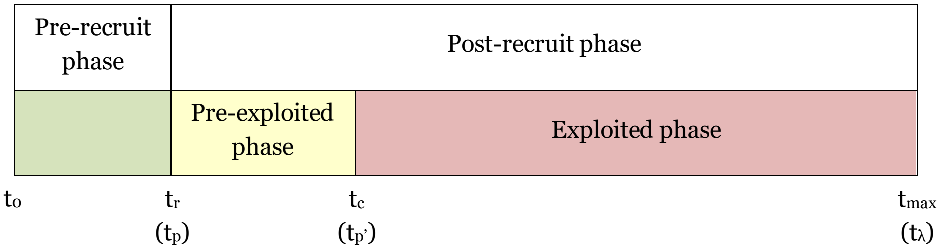
2.10. Stock simulation by Beverton and Holt's yield per recuit model (BHYPR)

Introduction

The yield per recuit model (Y/R) was originally developed by Beverton and Holt (1957) as an age-structured model for temperate fisheries. Later, Beverton and Holt (1964) introduced a length-structured version of the model, i.e., relative yield per recuit model (Y'/R) to address the data-poor fisheries of the tropical region. The computational aspects of the model were thoroughly interpreted by Ricker (1975), and a practical extension was later developed by Pauly and Soriano (1986).

The model assumes that all fish in a cohort are suddenly recruited to the fishing grounds at the same time (knife-edge recruitment) when they reach the age-at-recruitment (t_r, also referred to as t_p in the original description of the authors). Until recruitment, fish experience only natural mortality (M). After recruitment, at a specific age, the cohort is assumed to be suddenly exposed to fishing mortality (F) (knife-edge selection), which remains constant for the rest of their lives. This age is called the age-at-first capture (t_c, also referred to as tp' in the original description of the authors). Fish exit the fishery at a later maximum age, known as the age-at-derecruitment (t_{max}, also called t_λ in the original description of the authors). The period from t_c to t_{max} (t_{max} - t_c, or t_λ - $t_{p'}$ in the original description of the authors) is referred to as the exploited phase (exploited lifespan), while the period from t_o to t_c (t_c - t_o, or $t_{p'}$ - t_o in the original description of the authors) is known as the unexploited phase (unexploited lifespan). Within the unexploited phase, the period from t_r to t_c, i.e., t_c – t_r (or $t_{p'}$- t_p in the original description of the authors)—is called the pre-exploited phase.

In the case of temperate fish, the pre-exploited phase is usually longer, spanning several years, whereas in the case of tropical fish, which are often caught soon after recruitment, t_c is typically equal to t_r (i.e., t_p = $t_{p'}$). The period from t_o to t_{max} (t_{max} - t_o, or t_λ - t_o in the original description of the authors) is referred to as the entire lifespan of fish.



Age-based yield per recruit model and biomass per recruit model

As the fish in a cohort experience only natural mortality (M) between the ages of t_r and t_c, the number of survivors (N_{tc}) at age t_c can be expressed as follows:

$$N_{t_c} = R \times exp^{-M(t_c - t_r)}$$

After tc, as the fish in a cohort experience fishing mortality (F) along with natural mortality (M) (i.e., total mortality Z=F+M), the number of survivors (N_t) at a later age (t) can be expressed as:

$$N_t = N_{t_c} \times exp^{-(F+M)(t-t_c)} = N_{tc} \times exp^{-Z(t-t_c)}$$

Similarly, the number of fish caught between a very small time interval (t,t+Δt) can be calculated using Baranov's catch equation as follows:

$$C_{t,t+\Delta t} = N_t \times \frac{F}{Z} \times (1 - exp^{Z\times\Delta t})$$

However, when the time interval (Δt) is very small (e.g., Δt=t/100, or t/1000), the Baranov's catch equation can be simplified as follows:

$$C_{t,t+\Delta t} = N_t \times F \times \Delta t = N_{t_c} \times exp^{-Z(t-t_c)} \times F \times \Delta t$$

The corresponding yield can be calculated by multiplying the number of fish caught by the mean body weight (w_t), which is assumed to remain constant over the small time interval. The corresponding yield is calculated as follows:

$$Y_{t,t+\Delta t} = C_{t,t+\Delta t} \times w_t = N_{t_c} \times exp^{-Z(t-t_c)} \times F \times \Delta t \times w_t$$

In the above equation, the mean weight (w_t) of the age group can be calculated using the weight-form of the von Bertalanffy growth equation (VBGF), assuming isometric growth (b=3), as follows:

$$w_t = W_\infty \times \left(1 - exp^{-K(t-t_0)}\right)^3$$

The above equation can be further expanded as follows:

$$w_t = W_\infty \times \left(1 - 3 \times exp^{-K(t-t_0)} + 3 \times exp^{-2K(t-t_0)} - exp^{-3K(t-t_0)}\right)$$

Substituting w_t into the above equation will produce the following equation:

$$\begin{aligned} Y_{t,t+\Delta t} = N_{t_c} \times exp^{-Z(t-t_c)} \times F \times \Delta t \times w_t \\ = N_{t_c} \times exp^{-Z(t-t_c)} \times F \times \Delta t \times W_\infty \\ \times \left(1 - 3 \times exp^{-K(t-t_0)} + 3 \times exp^{-2K(t-t_0)} - exp^{-3K(t-t_0)}\right) \end{aligned}$$

To find the total yield over the exploited lifespan of the fish (from t_c to t_{max}, the age at which fish are derecruited from the fishery), the yield-per-age over the exploited lifespan (exploitation phase from t_c to the maximum age t_{max}) can be integrated as follows:

$$Y = N_{t_c} \times F \times W_\infty \times \int_{t_c}^{t_{max}} exp^{-Z(t-t_c)} \times \left(1 - 3 \times exp^{-K(t-t_0)} + 3 \times exp^{-2K(t-t_0)} - exp^{-3K(t-t_0)}\right) \times dt$$

Solving the integral will produce the following equation:

$$\begin{aligned} Y = N_{t_c} \times F \times W_\infty \times \Bigg[\frac{1 - exp^{-Z(t_{max}-t_c)}}{Z} - \frac{3 \times exp^{-K(t_c-t_0)} \times \left(1 - exp^{-(Z+K)(t_{max}-t_c)}\right)}{Z+K} \\ + \frac{3 \times exp^{-2K(t_c-t_0)} \times \left(1 - exp^{-(Z+2K)(t_{max}-t_c)}\right)}{Z+2K} \\ - \frac{exp^{-3K(t_c-t_0)} \times \left(1 - exp^{-(Z+3K)(t_{max}-t_c)}\right)}{Z+3K}\Bigg] \end{aligned}$$

The value of the derecruitment term containing t_{max}–t_c in the above equation can be considered close to unity by assuming t_{max}=∞, especially when Z is high (Ricker, 1975; Pauly and Soriano, 1986), and can be ignored. This further simplifies the above equation as follows:

$$Y = N_{t_c} \times F \times W_\infty \times \left[\frac{1}{Z} - \frac{3 \times exp^{-K(t_c-t_0)}}{Z+K} + \frac{3 \times exp^{-2K(t_c-t_0)}}{Z+2K} - \frac{exp^{-3K(t_c-t_0)}}{Z+3K}\right]$$

Now, substituting the N_{tc} with the R in the above equation allows it to be rewritten as:

$$Y = R \times exp^{-M(t_c-t_r)} \times F \times W_\infty \times \left[\frac{1}{Z} - \frac{3 \times exp^{-K(t_c-t_0)}}{Z+K} + \frac{3 \times exp^{-2K(t_c-t_0)}}{Z+2K} - \frac{exp^{-3K(t_c-t_0)}}{Z+3K}\right]$$

Finally, the yield per recruit for the exploited lifespan of the fish can be derived as:

$$\frac{Y}{R} = exp^{-M(t_c-t_r)} \times F \times W_\infty \times \left[\frac{1}{Z} - \frac{3 \times exp^{-K(t_c-t_0)}}{Z+K} + \frac{3 \times exp^{-2K(t_c-t_0)}}{Z+2K} - \frac{exp^{-3K(t_c-t_0)}}{Z+3K}\right]$$

The model can be further shortened by using the S for the term exp$^{-K(tc-to)}$ as follows:

$$\frac{Y}{R} = exp^{-M(t_c-t_r)} \times F \times W_\infty \times \left[\frac{1}{Z} - \frac{3 \times S}{Z+K} + \frac{3 \times S^2}{Z+2K} - \frac{S^3}{Z+3K}\right]$$

The biomass per recruit (B/R) can be derived from the yield per recruit (Y/R) by removing the fishing mortality from the equation:

$$\frac{B}{R} = exp^{-M(t_c-t_r)} \times W_\infty \times \left[\frac{1}{Z} - \frac{3 \times S}{Z+K} + \frac{3 \times S^2}{Z+2K} - \frac{S^3}{Z+3K}\right]$$

Length-based yield per recruit model and biomass per recruit model

The age terms in the above mentioned Y/R can be converted to the length-terms to develop the length-based yield per recruit model and biomass per recruit model.

The VBGF equation can be rearranged to solve the term S, i.e., exp$^{-K(tc-to)}$ as follows:

$$L_c = L_\infty \times \left(1 - exp^{-K(t_c-t_0)}\right) \text{ which can be solved to } 1 - \frac{L_c}{L_\infty} = exp^{-K(t_c-t_0)}$$

Using the length-based form of the term exp$^{-K(tc-to)}$, the age-based Y/R can be rewritten as:

$$\frac{Y}{R} = exp^{-M(t_c-t_r)} \times F \times W_\infty \times \left[\frac{1}{Z} - \frac{3 \times \left(1 - \frac{L_c}{L_\infty}\right)}{Z+K} + \frac{3 \times \left(1 - \frac{L_c}{L_\infty}\right)^2}{Z+2K} - \frac{\left(1 - \frac{L_c}{L_\infty}\right)^3}{Z+3K}\right]$$

Taking 1/Z as a common factor from the terms after W∞, the equation can be rewritten as:

$$\frac{Y}{R} = exp^{-M(t_c-t_r)} \times F \times W_\infty \times \frac{1}{Z} \times \left[1 - \frac{3 \times \left(1 - \frac{L_c}{L_\infty}\right)}{\left(1 + \frac{K}{Z}\right)} + \frac{3 \times \left(1 - \frac{L_c}{L_\infty}\right)^2}{\left(1 + \frac{2 \times K}{Z}\right)} - \frac{\left(1 - \frac{L_c}{L_\infty}\right)^3}{\left(1 + \frac{3 \times K}{Z}\right)}\right]$$

The term exp$^{-M(tc-tr)}$ in the above equation can be expanded as follows:

$$exp^{-M(t_c-t_r)} = exp^{-M(t_c-t_0-t_r+t_0)} = exp^{-M(t_c-t_0)+M(t_r-t_0)} = \frac{exp^{-M(t_c-t_0)}}{exp^{-M(t_r-t_0)}}$$

The above expanded form of the term exp$^{-M(tc-tr)}$ can be substituted into the Y/R equation as follows:

$$\frac{Y}{R} = \frac{exp^{-M(t_c-t_0)}}{exp^{-M(t_r-t_0)}} \times F \times W_\infty \times \frac{1}{Z} \times \left[1 - \frac{3 \times \left(1 - \frac{L_c}{L_\infty}\right)}{\left(1 + \frac{K}{Z}\right)} + \frac{3 \times \left(1 - \frac{L_c}{L_\infty}\right)^2}{\left(1 + \frac{2 \times K}{Z}\right)} - \frac{\left(1 - \frac{L_c}{L_\infty}\right)^3}{\left(1 + \frac{3 \times K}{Z}\right)}\right]$$

The age-based form of the term exp$^{-M(tc-to)}$ in the above equation can be converted to the length-based form using the VBGF as follows:

$$L_c = L_\infty \times \left(1 - exp^{-K(t_c-t_0)}\right) \text{, which can be solved to for } \mathrm{t_c - t_0} \text{ as follows}$$

$$t_c - t_0 = -\frac{1}{\mathrm{K}} \times \ln\left(1 - \frac{L_c}{L_\infty}\right) = exp^{-K(t_c-t_0)}$$

Multiplying the term exp^{-M} on both the side will produce the following equation:

$$exp^{-M(t_c-t_0)} = exp^{-M\times\left(-\frac{1}{K}\right)\times ln\left(1-\frac{L_c}{L_\infty}\right)} = \left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}}$$

The length-based form of term exp$^{-M(tc-to)}$ can be substituted into the Y/R equation as follows:

$$\frac{Y}{R} = \frac{\left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}}}{\boldsymbol{exp^{-M(t_r-t_0)}}} \times F \times \boldsymbol{W_\infty} \times \frac{1}{Z} \times \left[1 - \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)}{\left(1+\frac{K}{Z}\right)} + \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)^2}{\left(1+\frac{2\times K}{Z}\right)} - \frac{\left(1-\frac{L_c}{L_\infty}\right)^3}{\left(1+\frac{3\times K}{Z}\right)}\right]$$

Using a similar length-based transformation of the term exp$^{-M(tr-to)}$, a purely length-based form of the above equation can be expressed as:

$$\frac{Y}{R} = \frac{\left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}}}{\left(\boldsymbol{1-\frac{L_r}{L_\infty}}\right)^{\boldsymbol{\frac{M}{K}}}} \times F \times \boldsymbol{W_\infty} \times \frac{1}{Z} \times \left[1 - \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)}{\left(1+\frac{K}{Z}\right)} + \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)^2}{\left(1+\frac{2\times K}{Z}\right)} - \frac{\left(1-\frac{L_c}{L_\infty}\right)^3}{\left(1+\frac{3\times K}{Z}\right)}\right]$$

The above equation is a length-based form of the Beverton and Holt yield per recruitment model. This length-based Y/R was further refined by Beverton and Holt (1964) to develop a length-based relative yield per recruit model (Y′/R) for tropical data-limited conditions. Now, by multiplying the above equation (Y/R) with the term **exp$^{-M(tr-to)}$** or **[1-(L$_r$/L$_\infty$)]$^{M/K}$**, and then dividing by **W$_\infty$**, a new equation is produced, which is known as the Beverton and Holt relative yield per recruit model (Y′/R).

$$\frac{Y'}{R} = \frac{Y}{R} \times \frac{\boldsymbol{exp^{-M(t_r-t_0)}}}{\boldsymbol{W_\infty}} = \frac{Y}{R} \times \frac{\left(\boldsymbol{1-\frac{L_r}{L_\infty}}\right)^{\boldsymbol{\frac{M}{K}}}}{\boldsymbol{W_\infty}}$$

$$\frac{Y'}{R} = \left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}} \times \frac{F}{Z} \times \left[1 - \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)}{\left(1+\frac{K}{Z}\right)} + \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)^2}{\left(1+\frac{2\times K}{Z}\right)} - \frac{\left(1-\frac{L_c}{L_\infty}\right)^3}{\left(1+\frac{3\times K}{Z}\right)}\right]$$

The above equation for Y′/R can be rewritten using the term C for L$_c$/L$_\infty$ and E for F/Z. Sometimes, K/Z is also expressed as (1−E)/(M/K). Including these terms will change the equation as follows:

$$\frac{Y'}{R} = (1-C)^{\frac{M}{K}} \times E \times \left[1 - \frac{3\times(1-C)}{\left(1+\frac{1-E}{\frac{M}{K}}\right)} + \frac{3\times(1-C)^2}{\left(1+\frac{2\times(1-E)}{\frac{M}{K}}\right)} - \frac{(1-C)^3}{\left(1+\frac{3\times(1-E)}{\frac{M}{K}}\right)}\right]$$

The relative biomass per recruit (B’/R) can also be expressed using similar modifications to the biomass per recruit (B/R) model as follows:

$$\frac{B'}{R} = \frac{B}{R} \times \frac{\boldsymbol{exp^{-M(t_r-t_0)}}}{\boldsymbol{W_\infty}} = \frac{B}{R} \times \frac{\left(\boldsymbol{1-\frac{L_r}{L_\infty}}\right)^{\boldsymbol{\frac{M}{K}}}}{\boldsymbol{W_\infty}}$$

The relative biomass per recruit (B′/R) can also be calculated from the relative yield per recruit model (Y′/R) by dividing the latter by F, as follows:

$$\frac{B'}{R} = \frac{Y'}{R} \times \frac{1}{F}$$

In the Beverton and Holt’s prediction model (BHYPR) of TropFishR, the effect of the relative fishing mortality rate on the yield and biomass per recruit cannot be determined by setting the FM_relative to TRUE. The model simulates the population indices (yield and biomass) in response to the absolute changes in the fishing mortality rate only. It will provide stock estimates (yield and biomass) in terms of yield per recruit (Y/R) and biomass per recruit (B/R), rather than absolute values, even if the recruitment number from the cohort analysis is provided in the ‘*stock_size_1 =*’ portion of the code.

Beverton and Holt’s yield per recuit model: R Implementation

2.10.1. Requirement for Beverton and Holt's yield per recruit model (BHYPR)

LFQ file

A length-frequency data file (LFQ) is required for the Beverton and Holt’s yield per recruit (BHYPR) analysis (e.g., my_data). Refer to the previously mentioned steps in ***‘2.1.2 and 2.1.5. Creating a length frequency file (LFQ) on R’*** section to create a LFQ file if not created earlier.

Essential parameters

Growth and mortality parameters

The Beverton and Holt’s yield per recruit (BHYPR) analysis requires growth parameters (L_∞ and K) and the natural mortality rate (M). If these essential parameters are not already assigned, use the following code to add them to the LFQ file (e.g., my_data):

```
my_data$Linf<-13.95
my_data $K<-1.71
my_data$M<-2.75
```

LWR coefficients

Beverton and Holt’s yield per recruit (BHYPR) analysis requires LWR coefficients (a, and b). To derive the LWR coefficients, refer to ***‘2.12.Length–Weight Relationship (LWR)’*** section. Assign the coefficients ‘a’, and ‘b’ to the LFQ file (e.g., my_data) using the following codes:

```
my_data $a <- 0.0064
my_data $b <- 3.0059
```

Maturity parameters

Beverton and Holt’s yield per recruit (BHYPR) analysis requires maturity parameters (LM_{50} and WM_{50}). To derive the length and weight at maturity (LM_{50} and WM_{50}) refer to ***‘2.13.Length at Maturity (LM_{50})’*** section. Assign the LM_{50}, WM_{50} values to the LFQ file (e.g., my_data) using the following codes:

```
my_data $Lmat <- 8.24
my_data$wmat <- my_data$a*(my_data$Lmat^my_data$b)
```

2.10.2. Simulating the effect of changes in FM on the stock status (BHYPR1)

This approach is used to simulate the effect of changes in fishing mortality rate (FM) on the stock status.

Step-1: Assign the additional parameters for BHYPR1

The Beverton and Holt yield per recruit (BHYPR) analysis requires two additional parameters: (1) the length at recruitment (Lr) which is the mid-length of the first (smallest) length class, and (2) the length at first capture (Lc) also known as LC_{50} (the length at which fish have a 50% probability to be caught). The LC_{50} values from the catch curve analysis are often used in the BHYPR analysis.

First, perform the catch curve analysis to get the LC50 using the following code:

```
CC <- catchCurve(mean_catch_vector, catch_columns = c(1), calc_ogive = TRUE)
```

Assign the LC_{50} derived from the catch curve analysis to the LFQ file (e.g., my_data) using the following code:

```
my_data $Lc <- CC$L50
```

> ***Note:*** *This is a very crucial step in the analysis. The LC50 can be derived either for a specific combination of years or a specific year. The above example shows the procedure for deriving LC50 for a specific combination of years (e.g., all available years using* ***mean_catch_vec****) using the length converted catch curve analysis. Alternatively, the user can derive LC50 for any specified year using the* ***catch_vec*** *in the length converted catch curve analysis by specifying the serial number of the required year in the catch column of the catch curve code* ***(catch_columns =)****. To perform catch curve analysis for the required combination of years, refer to the '****2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data'****. To perform catch curve analysis for the required year, refer to the '****2.7.3.Catch curve analysis for a specific year'.***

Assign the length at recruitment (Lr) to the LFQ file (e.g., my_data) using the following code:

```
my_data $Lr <- as.numeric(min(my_data $midLengths))
```

Step-2: Perform BHYPR1 under the knife-edge selection assumption

To ensure the function follows the knife-edge selection assumption instead of the gear selectivity assumption, use the BHYPR code providing no selectivity information (s_list = NA).

```
BHYPR1 <- predict_mod (my_data, type = "ypr", FM_change = seq(0,20,0.1),
E_change = NA, curr.E = NA, curr.Lc = NA, s_list = NA, plot = TRUE)
```

Step-3: Perform BHYPR1 under the gear selection assumption

Provide selectivity information (s_list = selectivity_list) in the BHYPR code to ensure the function follows the gear selectivity assumption instead of the knife-edge selection assumption. Prepare a gear selection list (gear_selection) using the LC_{50} and LC_{75}

information derived from a catch curve analysis. First, perform the catch curve analysis to get the gear selectivity information using the following code:

```
CC <- catchCurve(mean_catch_vector, catch_columns = c(1), calc_ogive = TRUE)
```

Then, prepare a gear selection list using the following code:

```
gear_selection <- list(selecType = "trawl_ogive", L50 = CC$L50, L75 = CC$L75)
```

Note: *This is a very crucial step in the analysis. The gear selection list can be derived either for a specific combination of years or a specific year. The above example shows the procedure for deriving gear selection list for a specific combination of years (e.g., all available years using* ***mean_catch_vec****) using the length converted catch curve analysis. Alternatively, the user can derive the gear selection list for any specified year using the* ***catch_vec*** *in the length converted catch curve analysis by specifying the serial number of the required year in the catch column of the catch curve code* ***(catch_columns =)****. To perform catch curve analysis for the required combination of years, refer to the '****2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data'.*** *To perform catch curve analysis for the required year, refer to the '****2.7.3.Catch curve analysis for a specific year'.***

To perform the BHYPR1 analysis under the gear selection assumption, use the following code:

```
BHYPR1 <- predict_mod (my_data, type = "ypr", FM_change = seq(0,20,0.01), E_change = NA, curr.E = NA, curr.Lc = NA, s_list = gear_selection, plot = TRUE)
```

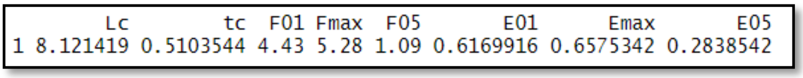
Note: *The stock estimates, i.e., yield per recruit (Y/R), biomass per recruit (B/R), relative yield per recruit (Y'/R), relative biomass per recruit (B'/R) and management reference points derived under knife-edge selection assumption may differ from those under gear selection assumption. In the above BHYPR1 codes,* ***type*** *represents the Beverton and Holt YPR analysis (****type = "ypr"****). For the Thompson and Bell analysis, the user can use* ***type = "ThompBell"****, after providing the required parameters for the analysis. The FM range, over which the effect of a change in the fishing mortality rates (FM) on stock status is evaluated, is supplied in* ***FM_change*** *[e.g.,* ***FM_change = seq(0, 20, 0.5)****]. The FM range can not be relative in Beverton and Holt YPR analysis and therefore only the absolute (****FM_relative = FALSE****) is allowed for the simulation. In this example, the FM changes are in absolute term (****FM_relative = FALSE****), simulating the effect of changes in FM (absolute F, not the F-multiplier) from 0 to 20 with a gradual increase of 0.5 (step size 0.5 shows a sequential FM change of 0, 0.5, 1.0, ..., 19.5, and 20.0). The user can also use an equivalent range of absolute exploitation rates (E_change) instead of the fishing mortality rate (FM_change) to simulate the same effect. However, if E_change is used instead of FM_change, the range is capped at E = 0.9, as higher values of E correspond to unrealistically high fishing mortality rates. The recruitment number, if supplied with* ***stock_size_1,*** *does not work in Beverton and Holt YPR analysis and, therefore, the analysis provides stock simulation in a relative Y/R and B/R format. The current exploitation rate (****curr.E****) and the current length at capture (****curr.Lc****) are not required to derive management reference points. However, the user must supply both the calculated E (i.e., F/Z) [e.g.,* ***curr.E = (as.numeric(CC$FM)/CC$Z)****] and LC50 values [e.g.,* ***curr.Lc = CC$L50****] from the catch curve analysis to assess the current stock status. The default graphical output from the simulation can be turned on (****plot = TRUE****) or off (****plot = FALSE****).*

2.10.3. Biological reference points from BHYPR1

Use the following code to get the biological reference points from BHYPR1.

BHYPR1$df_Es

e.g., when the gear selection assumption is used



```
         Lc        tc  F01 Fmax  F05       E01       Emax       E05
1 8.121419 0.5103544 4.43 5.28 1.09 0.6169916 0.6575342 0.2838542
```

Note: *The management reference points may differ under these two different assumptions. For more information on these and other management reference points, refer to '**2.11.Understanding the stock simulation outputs (fisheries management reference points)**'.*

2.10.4. Current stock status estimates from BHYPR1

To get current values for stock estimates (i.e., yield per recruit (Y/R), biomass per recruit (B/R), relative yield per recruit (Y'/R), and relative biomass per recruit (B'/R)), user need to provide both the current exploitation rate (curr.E=) and the current length at capture (curr.Lc=) in the above mentioned BHYPR code. Use the calculated E (i.e., F/Z) [e.g., curr.E= (as.numeric(CC$FM)/CC$Z)] and LC_{50} values [e.g., curr.Lc =CC$L50] from the catch curve analysis in BHYPR1 code. To perform the BHYPR1 analysis under the knife-edge selection assumption, use the following codes:

BHYPR1 <- predict_mod (my_data, type = "ypr", FM_change = seq(0,20,0.1), E_change = NA, **curr.E= (as.numeric(CC$FM)/CC$Z)**, **curr.Lc =CC$L50**, **s_list = NA**, plot = TRUE)

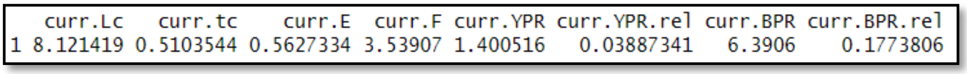
To perform the BHYPR1 analysis under the gear selection assumption, use the following codes:

BHYPR1 <- predict_mod (my_data, type = "ypr", FM_change = seq(0,20,0.01), **curr.E= (as.numeric(CC$FM)/CC$Z)**, **curr.Lc =CC$L50**, **s_list = gear_selection**, plot = TRUE)

After entering the values for the current exploitation rate and the current length at capture in the BHYPR1 code, use the following code to retrieve the current values:

BHYPR1$currents

e.g., when the gear selection assumption is used

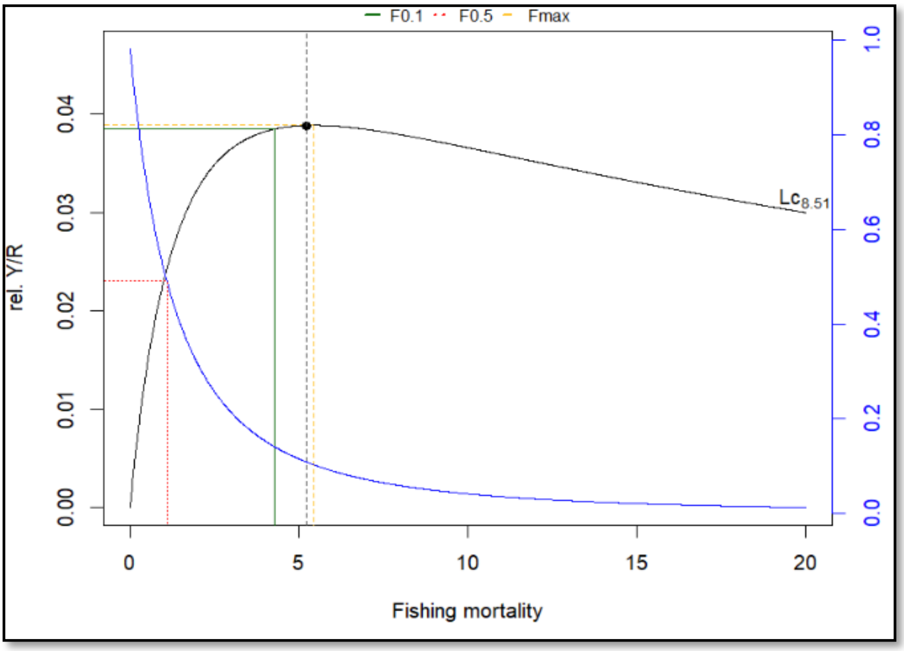


```
   curr.Lc   curr.tc    curr.E  curr.F curr.YPR curr.YPR.rel curr.BPR curr.BPR.rel
1 8.121419 0.5103544 0.5627334 3.53907 1.400516   0.03887341   6.3906    0.1773806
```

Note: *The stock estimates, i.e., yield per recruit (Y/R), biomass per recruit (B/R), relative yield per recruit (Y'/R) and relative biomass per recruit (B'/R) may differ under these two different assumptions. The yield (YPR) and biomass (BPR) values are the values per recruit expressed in grams.*

2.10.5. Default graphical output from BHYPR1

The default graphical output from the simulation can be turned on by mentioning ***plot*** = ***TRUE*** in the BHYPR1 code, which will produce the following graph.



2.10.6. Simulating the combined effect of changes in FM and LC50 on stock status (BHYPR2)

It is used to simulate the combined effect of simultaneous changes in fishing mortality rate (FM) along with the length at capture (LC_{50}) on the stock status.

Step-1: Define the LC (length at capture) range

Define the range of LC (length at capture) over which the effect is evaluated.

```
LC_min<- min(my_data$midLengths)
LC_max<- max(my_data$midLengths)
```

> **Note:** *for demonstration, the LC range provided in this example spans from the minimum to the maximum observed length. The user can define any other length range.*

Step-2: Assign the additional parameters for BHYPR

The Beverton and Holt yield per recruit (BHYPR) analysis requires two additional parameters: (1) the length at recruitment (Lr) which is the mid-length of the first (smallest) length class and (2) the length at first capture (Lc) also known as LC_{50} (the length at which fish have a 50% probability to be caught). The LC_{50} values from the catch curve analysis are often used in the BHYPR analysis. First, perform the catch curve analysis to get the LC50 using the following code:

```
CC <- catchCurve(mean_catch_vector, catch_columns = c(1), calc_ogive = TRUE)
```

Assign the LC_{50} derived from the catch curve analysis to the LFQ file (e.g., my_data) using the following code: `my_data$Lc <- CC$L50`

Note: *This is a very crucial step in the analysis. The LC50 can be derived either for a specific combination of years or a specific year. The above example shows the procedure for deriving LC50 for a specific combination of years (e.g., all available years using* ***mean_catch_vec****) using the catch curve analysis. Alternatively, the user can derive LC50 for any specified year using the* ***catch_vec*** *in the length converted catch curve analysis by specifying the serial number of the required year in the catch column of the catch curve code (***catch_columns** *=). To perform catch curve analysis for the required combination of years, refer to the '****2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data****'. To perform catch curve analysis for the required year, refer to the '****2.7.3.Catch curve analysis for a specific year****'.*

Assign the length at recruitment (Lr) to the LFQ file (e.g., my_data) using the following code: my_data$Lr <- as.numeric(min(my_data$midLengths))

Step-3: Perform BHYPR2 under the knife-edge selection assumption

This is the default assumption of the BHYPR function. To ensure the function follows the knife-edge selection assumption instead of gear selectivity assumption, use the BHYPR code providing no selectivity information **(s_list = NA)**. To perform the BHYPR2 analysis under the knife-edge selection assumption, use the following codes:

BHYPR2 <- predict_mod(my_data, type = "ypr", **FM_change = seq(0,20,0.5)**, E_change = NA, **Lc_change = seq(LC_min, LC_max,0.5)**, curr.E = NA, curr.Lc = NA, **s_list = NA**, plot = TRUE)

Step-4: Perform BHYPR2 under the gear selectivity assumption

Provide selectivity information **(s_list = gear_selection)** in the BHYPR code to ensure the function follows the gear selectivity assumption instead of the knife-edge selection assumption. Prepare a gear selection list (**gear_selection**) using the LC_{50} and LC_{75} information derived from a catch curve analysis. First, perform the catch curve analysis to get the gear selectivity information using the following code:

CC <- catchCurve(**mean_catch_vector**, catch_columns = c(1), calc_ogive = TRUE)

Then, prepare a gear selection list using the following code:

gear_selection <- list(selecType = "trawl_ogive", L50 = CC$L50, L75 = CC$L75)

Note: *This is a very crucial step in the analysis. The gear selection list can be derived either for a specific combination of years or a specific year. The above example shows the procedure for deriving gear selection list for a specific combination of years (e.g., all available years using* ***mean_catch_vec****) using the catch curve analysis. Alternatively, the user can derive the gear selection list for any specified year using the* ***catch_vec*** *in the catch curve analysis by specifying the serial number of the required year in the catch column of the catch curve code* ***(catch_columns =)****. To perform catch curve analysis for the required combination of years, refer to the* ***2.7.2.Catch curve analysis for a combination of years using multiyear mean catch data.*** *To perform catch curve analysis for the required year, refer to the* ***2.7.3.Catch curve analysis for a specific year.***

To perform the BHYPR2 analysis under the gear selection assumption, use the following codes:

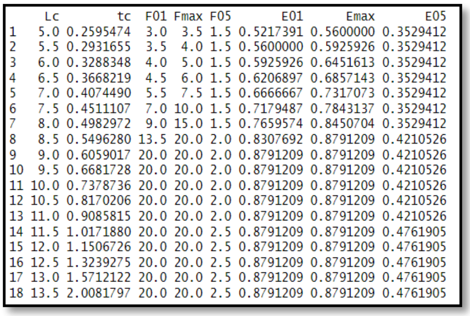
BHYPR2 <- predict_mod(my_data, type = "ypr", **FM_change = seq(0,20,0.5)**, E_change = NA, Lc_change = seq(**LC_min, LC_max,0.5)**, curr.E = NA, curr.Lc = NA, **s_list = gear_selection**, plot = TRUE)

> ***Note:*** *The stock estimates, i.e., yield per recruit (Y/R), biomass per recruit (B/R), relative yield per recruit (Y'/R), relative biomass per recruit (B'/R) and management reference points derived under knife-edge selection assumption may differ from those under gear selection assumption. In the above BHYPR2 codes,* ***type*** *represents the Beverton and Holt YPR analysis* ***(type = "ypr")****. For the Thompson and Bell analysis, the user can use* ***type = "ThompBell"****, after providing the required parameters for the analysis. The FM range, over which the effect of a change in the fishing mortality rates (FM) on stock status is evaluated, is supplied in* ***FM_change*** *[e.g.,* ***FM_change = seq(0, 20, 0.5)****]. The FM range can not be relative (****FM_relative = TRUE****) in Beverton and Holt YPR analysis and therefore, only the absolute (****FM_relative = FALSE****) is allowed for the simulation. In this example, the FM changes are in absolute format (****FM_relative = FALSE****), simulating the effect of changes in FM (absolute F, not the F-multiplier) from 0 to 20 with a gradual increase of 0.5 (step size 0.5 shows a sequential FM change of 0, 0.5, 1.0, ..., 19.5, and 20.0). The user can also use an equivalent range of absolute exploitation rates (****E_change****) instead of the fishing mortality rate (****FM_change****) to simulate the same effect. However, if E_change is used instead of FM_change, the range is capped at E = 0.9, as higher values of E correspond to unrealistically high fishing mortality rates. In addition to the FM, the used can simulate the effect of a change in length at capture (LC) by providing a LC range in the BHYPR2 code [****Lc_change = seq(LC_min, LC_max, 0.5)****]. The LC range provided in this example ranges from the minimum to the maximum observed length with a gradual increase of 0.5 (step size 0.5 shows a sequential LC change of 5, 5.5, 1.0, ..., 13.0, and 13.5). Users have the flexibility to specify any other length range and step size. The recruitment number, if supplied with* ***stock_size_1,*** *does not work in Beverton and Holt YPR analysis and therefore, the analysis provides stock simulation in a relative Y/R and B/R terms. The current exploitation rate* ***(curr.E)*** *and the current length at capture* ***(curr.Lc)*** *are not required to derive management reference points. However, the user must supply both the calculated E (i.e., F/Z) [e.g.,* ***curr.E = (as.numeric(CC$FM)/CC$Z)****] and LC50 values [e.g.,* ***curr.Lc = CC$L50****] from the catch curve analysis to assess the current stock status. The default graphical output from the simulation can be turned on (****plot = TRUE****) or off (****plot = FALSE****).*

2.10.7. Biological reference points from BHYPR2

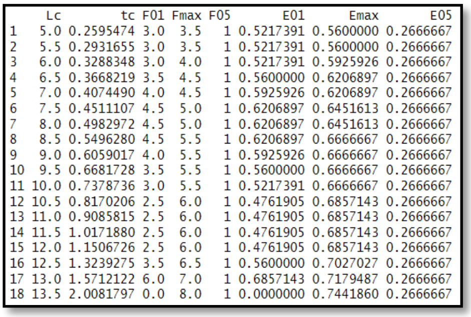
Use the following code to get the biological reference points from BHYPR2: BHYPR2$df_Es

BHYPR2 with knife-edge selection



```
       Lc        tc  F01 Fmax F05       E01      Emax       E05
1    5.0 0.2595474  3.0  3.5 1.5 0.5217391 0.5600000 0.3529412
2    5.5 0.2931655  3.5  4.0 1.5 0.5600000 0.5925926 0.3529412
3    6.0 0.3288348  4.0  5.0 1.5 0.5925926 0.6451613 0.3529412
4    6.5 0.3668219  4.5  6.0 1.5 0.6206897 0.6857143 0.3529412
5    7.0 0.4074490  5.5  7.5 1.5 0.6666667 0.7317073 0.3529412
6    7.5 0.4511107  7.0 10.0 1.5 0.7179487 0.7843137 0.3529412
7    8.0 0.4982972  9.0 15.0 1.5 0.7659574 0.8450704 0.3529412
8    8.5 0.5496280 13.5 20.0 2.0 0.8307692 0.8791209 0.4210526
9    9.0 0.6059017 20.0 20.0 2.0 0.8791209 0.8791209 0.4210526
10   9.5 0.6681728 20.0 20.0 2.0 0.8791209 0.8791209 0.4210526
11  10.0 0.7378736 20.0 20.0 2.0 0.8791209 0.8791209 0.4210526
12  10.5 0.8170206 20.0 20.0 2.0 0.8791209 0.8791209 0.4210526
13  11.0 0.9085815 20.0 20.0 2.0 0.8791209 0.8791209 0.4210526
14  11.5 1.0171880 20.0 20.0 2.5 0.8791209 0.8791209 0.4761905
15  12.0 1.1506726 20.0 20.0 2.5 0.8791209 0.8791209 0.4761905
16  12.5 1.3239275 20.0 20.0 2.5 0.8791209 0.8791209 0.4761905
17  13.0 1.5712122 20.0 20.0 2.5 0.8791209 0.8791209 0.4761905
18  13.5 2.0081797 20.0 20.0 2.5 0.8791209 0.8791209 0.4761905
```

BHYPR2 with gear selection



```
       Lc        tc F01 Fmax F05       E01      Emax       E05
1    5.0 0.2595474 3.0  3.5   1 0.5217391 0.5600000 0.2666667
2    5.5 0.2931655 3.0  3.5   1 0.5217391 0.5600000 0.2666667
3    6.0 0.3288348 3.0  4.0   1 0.5217391 0.5925926 0.2666667
4    6.5 0.3668219 3.5  4.5   1 0.5600000 0.6206897 0.2666667
5    7.0 0.4074490 4.0  4.5   1 0.5925926 0.6206897 0.2666667
6    7.5 0.4511107 4.5  5.0   1 0.6206897 0.6451613 0.2666667
7    8.0 0.4982972 4.5  5.0   1 0.6206897 0.6451613 0.2666667
8    8.5 0.5496280 4.5  5.5   1 0.6206897 0.6666667 0.2666667
9    9.0 0.6059017 4.0  5.5   1 0.5925926 0.6666667 0.2666667
10   9.5 0.6681728 3.5  5.5   1 0.5600000 0.6666667 0.2666667
11  10.0 0.7378736 3.0  5.5   1 0.5217391 0.6666667 0.2666667
12  10.5 0.8170206 2.5  6.0   1 0.4761905 0.6857143 0.2666667
13  11.0 0.9085815 2.5  6.0   1 0.4761905 0.6857143 0.2666667
14  11.5 1.0171880 2.5  6.0   1 0.4761905 0.6857143 0.2666667
15  12.0 1.1506726 2.5  6.0   1 0.4761905 0.6857143 0.2666667
16  12.5 1.3239275 3.5  6.5   1 0.5600000 0.7027027 0.2666667
17  13.0 1.5712122 6.0  7.0   1 0.6857143 0.7179487 0.2666667
18  13.5 2.0081797 0.0  8.0   1 0.0000000 0.7441860 0.2666667
```

> ***Note:*** *The output shows the effect of change in length at capture (LC_{50}) or age at capture (t_{50}) on biological management reference points (e.g., fishing mortality rates: F_{01}, F_{max} and F_{05} or exploitation rates: E_{01}, E_{max} and E_{05}). For more information on these and other management reference points, refer to* ***2.11.Understanding the stock simulation outputs (fisheries management reference points)'.***

2.10.8. Current stock status estimates from BHYPR2

To get current values for stock estimates (i.e., yield per recruit (Y/R), biomass per recruit (B/R), relative yield per recruit (Y'/R), and relative biomass per recruit (B'/R)), user need to provide both the current exploitation rate (**curr.E=**) and the current length at capture (**curr.Lc=**) in the above mentioned BHYPR2 code. Use the calculated E (i.e., F/Z) [e.g., **curr.E= (as.numeric(CC$FM)/CC$Z)**] and LC_{50} values [e.g., **curr.Lc =CC$L50**] from the catch curve analysis in BHYPR2 code as follows:

To perform the BHYPR2 analysis under the knife-edge selection assumption, use the following codes:

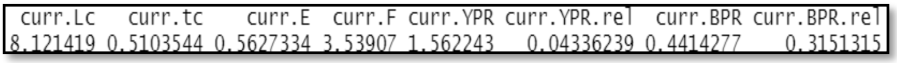
```
BHYPR2 <- predict_mod(my_data, type = "ypr", FM_change = seq(0,20,0.5), E_change = NA, Lc_change = seq(LC_min, LC_max,0.5), curr.E = (as.numeric(CC$FM)/CC$Z), curr.Lc = CC$L50, s_list = NA, plot = TRUE)
```

To perform the BHYPR2 analysis under the gear selection assumption, use the following codes:

```
BHYPR2 <- predict_mod(my_data, type = "ypr", FM_change = seq(0,20,0.5), E_change = NA, Lc_change = seq(LC_min, LC_max,0.5), curr.E = (as.numeric(CC$FM)/CC$Z), curr.Lc = CC$L50, s_list = gear_selection, plot = TRUE)
```

After entering the values for the current exploitation rate and the current length at capture in the BHYPR2 code, use the following code to retrieve the current values:

```
BHYPR2$currents
```

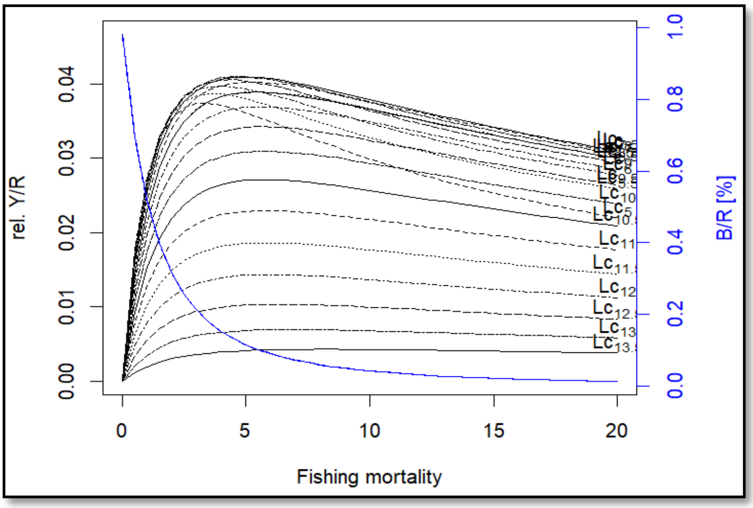


```
 curr.Lc   curr.tc    curr.E  curr.F curr.YPR curr.YPR.rel  curr.BPR curr.BPR.rel
8.121419 0.5103544 0.5627334 3.53907 1.562243   0.04336239 0.4414277    0.3151315
```

> ***Note:*** *The stock estimates, i.e., yield per recruit (Y/R), biomass per recruit (B/R), relative yield per recruit (Y'/R) and relative biomass per recruit (B'/R) may differ under these two different assumptions. The yield (YPR) and biomass (BPR) values are the values per recruit expressed in grams.*

2.10.9. Default graphical output from BHYPR2

The default graphical output from the simulation can be turned on by mentioning ***plot = TRUE*** in the BHYPR2 code, which will produce the following graph.



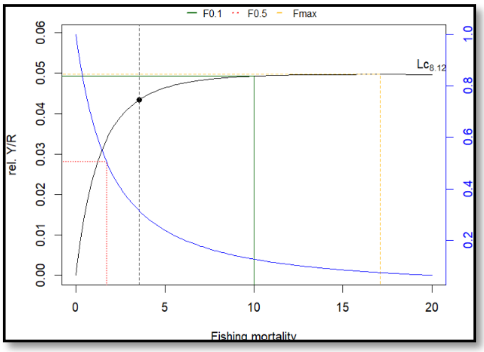
2.10.10. Enhanced visualization of Beverton and Holt's yield per recuit model (BHYPR)

Plotting the effect of a change in FM on stock status (BHYPR1 graph)

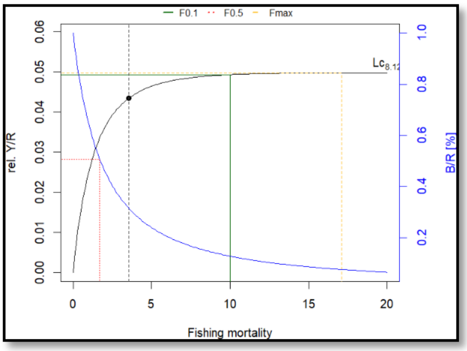
The previously mentioned BHYPR1 code when used with plot option set as true (**plot=true**) produces the relative Y/R (Y'/R) and the relative B/R (B'/R) trajectory in response to change in FM as a default graphical output. The default BHYPR1 graphical output may not produce a complete visualization of all the outputs, especially when any output is plotted on a secondary Y-axis. It happens mainly due to the non-availability of adequate space on the right side margin. As a default, R uses a margin setup of **par(mar=c(5, 4, 4, 2))**, which is 5, 4, 4 and 2 line spaces for the bottom, left, top and right side margins, respectively. To accommodate adequate space for the right side margin for plotting the extra Y-axis, use the required BHYPR1 code after setting the margin space using the following code: par(mar=c(5, 4, 4, 5))

```
BHYPR1 <- predict_mod (my_data, type = "ypr", FM_change = seq(0,20,0.1), E_change = NA, curr.E= (as.numeric(CC$FM)/CC$Z), curr.Lc =CC$L50, s_list = NA, plot = TRUE)
```

Default graphical output with par(mar=c(5, 4, 4, 2))



Better graphical output with par(mar=c(5, 4, 4, 5))

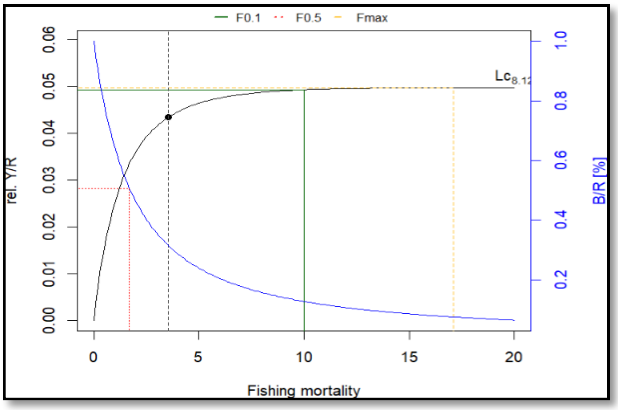


Controlling the BHYPR1 plotting parameters for primary and secondary Y-axis

The default graphical output from BHYPR1 plots ‘relative Y/R’ in Y-axis and ‘relative B/R’ as proportion of the highest level of B’/R (when F=0) in the secondary Y-axis.

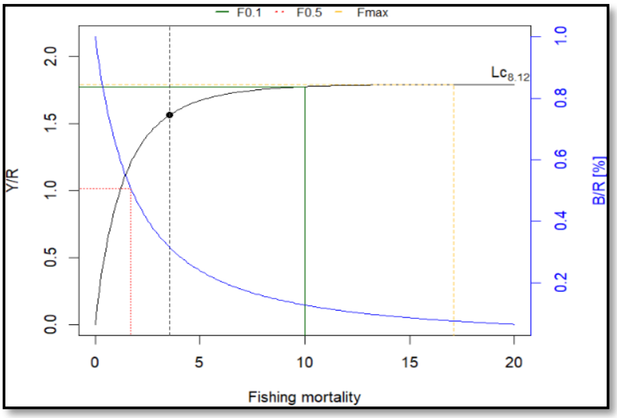
Default BHYPR1 plots relative Y/R (Y’/R) and relative B/R (B’/R) as proportion in the plot, which can be reproduced by defining relative Y/R in yaxis1 and relative B/R in yaxis2.

```
par(mar=c(5,4,4,5))
plot(BHYPR1, type = "ypr", xaxis1 = "FM", yaxis1 = "Y_R.rel", yaxis2 = "B_R.rel", identify = FALSE)
```



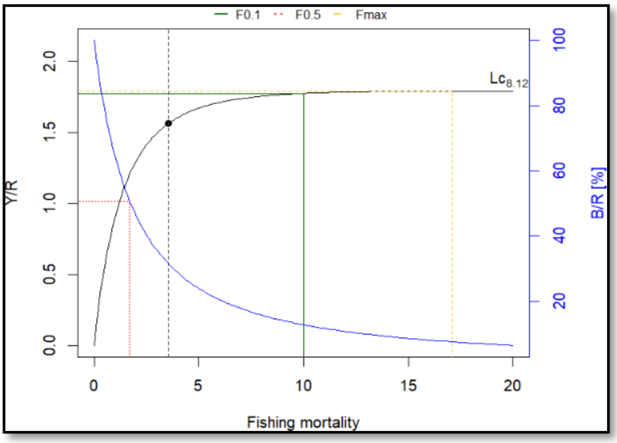
To plot Y/R instead of relative Y/R (Y’/R), define Y/R in yaxis1. If yaxis2 is not defined or changed, as a default, it will plot the relative B/R (B’/R) as proportion

```
par(mar=c(5,4,4,5))
plot(BHYPR1, type = "ypr", xaxis1 = "FM", yaxis1 = "Y_R", identify = FALSE)
```



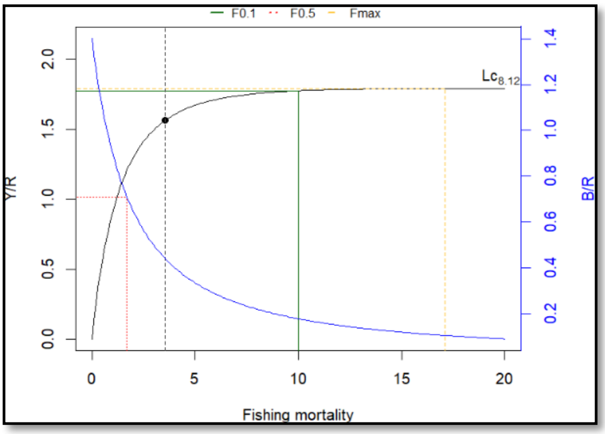
To plot Y/R instead of relative Y/R (Y’/R) and B/R as percentage instead of relative B/R (B’/R) as proportion, define Y/R in yaxis1 and B/R percent in yaxis2.

```
par(mar=c(5,4,4,5))
plot(BHYPR1, type = "ypr", xaxis1 = "FM", yaxis1 = "Y_R", yaxis2 = "B_R.percent", identify = FALSE)
```



To plot Y/R instead of relative Y/R (Y'/R) and B/R instead of relative B/R (B'/R) or B/R as proportion or percentage, define Y/R in yaxis1 and B/R in yaxis2.

```
par(mar=c(5,4,4,5))
plot(BHYPR1, type = "ypr", xaxis1 = "FM", yaxis1 = "Y_R", yaxis2 = "B_R", identify = FALSE)
```



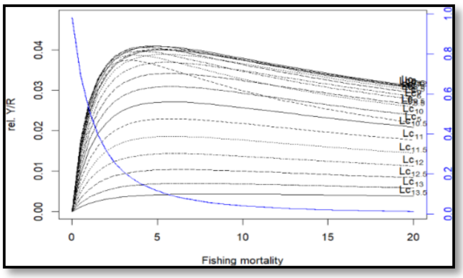
> ***Note:*** *Unlike the relative Y/R (Y'/R), the relative B/R (B'/R) in TropFishR is expressed as a proportion or fraction of the highest level of B'/R (when F=0). Therefore, the B/R when expressed as a percentage of the highest level of B/R (when F=0) is same as the B'/R expressed in proportion to the highest level of B'/R (when F=0).*

Plotting the effect of a change in FM and LC on stock status (BHYPR2 outputs)

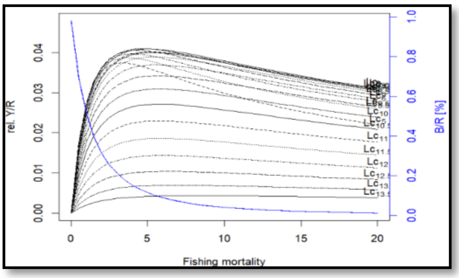
The previously mentioned BHYPR2 code when used with plot option set as true (**plot=true**) also produces the relative Y/R (Y'/R) and the relative B/R (B'/R) trajectories in response to change in FM and LC (line plot) as a default graphical output. The default BHYPR2 graphical output may not produce a complete visualization of all the outputs, especially when any output is plotted on a secondary Y-axis. It happens mainly due to the non-availability of adequate space on the right side margin of the plot. As a default, R uses a margin setup of **par(mar=c(5, 4, 4, 2))**, which is 5, 4, 4 and 2 line spaces for the bottom, left, top and right side margins, respectively. To accommodate adequate space for the right side margin for plotting the extra Y-axis, use the required BHYPR2 code after setting the margin space using the following code: par(mar=c(5, 4, 4, 5))

```
BHYPR2 <- predict_mod(my_data, type = "ypr", FM_change = seq(0,20,0.5), E_change = NA, Lc_change = seq(LC_min, LC_max,0.5), curr.E = (as.numeric(CC$FM)/CC$Z), curr.Lc = CC$L50, s_list = gear_selection, plot = TRUE)
```

Default graphical output with par(mar=c(5, 4, 4, 2))



Better graphical output with par(mar=c(5, 4, 4, 5))



Controlling the BHYPR2 plotting parameters for primary and secondary Y-axis

The default graphical output from BHYPR2 plots 'relative Y/R' in Y-axis and 'relative B/R' as proportion or fraction of the highest level of B'/R (when F=0) in the secondary Y-axis. The primary Y-axis (yaxis1) and secondary Y-axis (yaxis2) can be controlled to plot Y/R and B/R or Y'/R and B'/R. For example, to plot Y/R instead of relative Y/R (Y'/R) and B/R instead of relative B/R (B'/R) or B/R as proportion or percentage, define Y/R in yaxis1 and B/R in yaxis2.

```
par(mar=c(5,4,4,5))
plot(BHYPR2, type = "ypr", xaxis1 = "FM", yaxis1 = "Y_R", yaxis2 = "B_R", identify = FALSE)
```

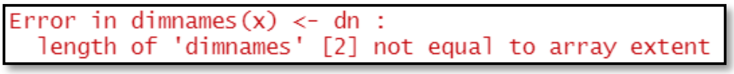
For more details, refer to ***'Controlling the BHYPR1 plotting parameters for primary and secondary Y-axis'*** to plot the similar plots with BHYPR2 outputs.

Plotting and controlling yield and biomass isopleths from BHYPR2

The previously mentioned BHYPR2 code, when used with plot option set as true (plot=true) does not produce the yield or biomass isopleths as a default graphical output. The Isopleths for the yield (Y/R), relative yield (Y'/R), biomass (B/R) and relative biomass (B'/R) can be produced by changing the chart type to "Isopleth" from the default of "ypr".

```
plot(BHYPR2, type = "Isopleth", xaxis1 = "FM", yaxis1 = "Y_R", yaxis2 = "B_R", identify = FALSE)
```

However, the code will produce the following error because of the presence of an extra row of Y/R, Y/R, B/R and B'/R values for the previously defined current LC.



```
Error in dimnames(x) <- dn :
  length of 'dimnames' [2] not equal to array extent
```

First, remove the already defined Lc to fix the above error using code: `my_data$Lc <-NULL`

Then, redo the BHYPR2 simulation (following either the knife-edge selection assumption or the gear selection assumption) turning the plot off (**plot = FALSE**) to suppress unwanted graphs in the absence of predefined LC.

To redo the BHYPR2 analysis under the knife-edge selection assumption, use the following codes:

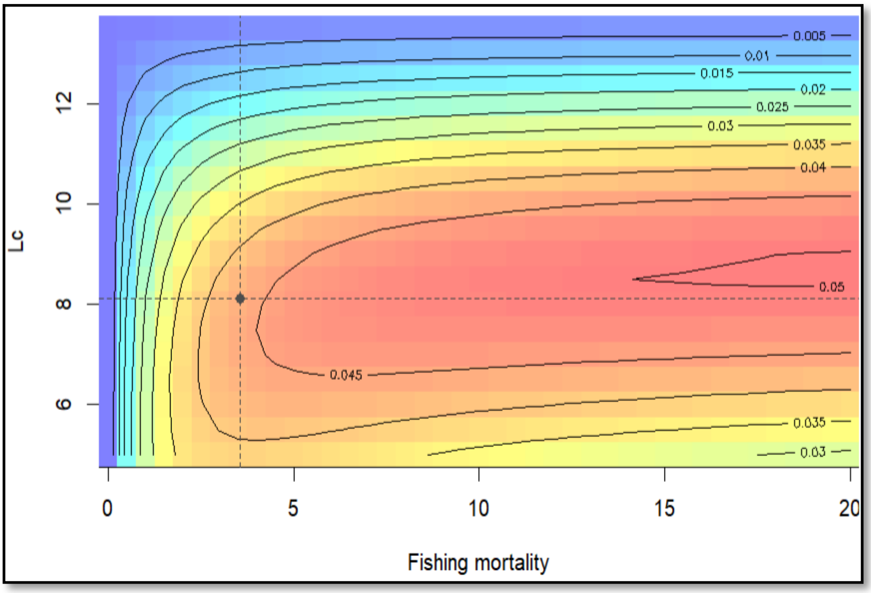
```
BHYPR2 <- predict_mod(my_data, type = "ypr", FM_change = seq(0,20,0.5), E_change = NA, Lc_change = seq(LC_min, LC_max,0.5), curr.E = (as.numeric(CC$FM)/CC$Z), curr.Lc = CC$L50, s_list = NA, plot = FALSE)
```

To redo the BHYPR2 analysis under the gear selection assumption, use the following codes:

```
BHYPR2 <- predict_mod(my_data, type = "ypr", FM_change = seq(0,20,0.5), E_change = NA, Lc_change = seq(LC_min, LC_max,0.5), curr.E = (as.numeric(CC$FM)/CC$Z), curr.Lc = CC$L50, s_list = gear_selection, plot = FALSE)
```

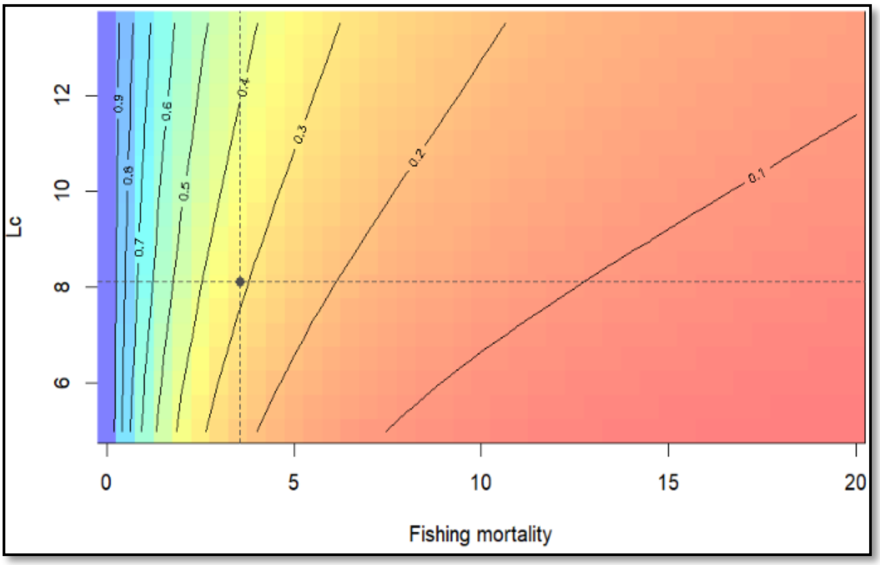
Once the BHYPR2 outputs are generated, the relative Y/R (Y'/R) isopleths can be plotted using the following code:

```
plot(BHYPR2, type = "Isopleth", xaxis1 = "FM", yaxis1 = "Y_R.rel", mark = TRUE, identify = FALSE)
```



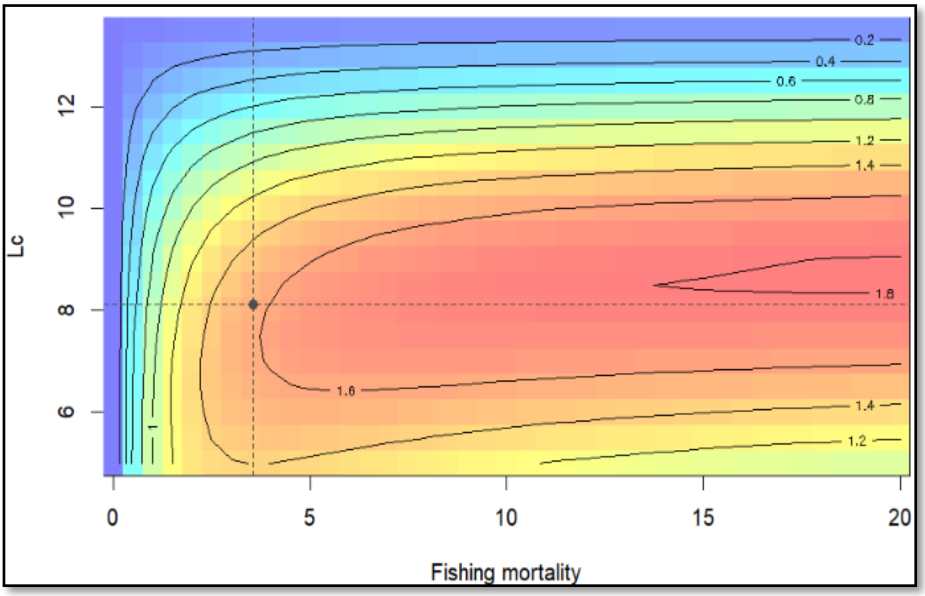
Once the BHYPR2 outputs are generated, the **relative B/R (B'/R) isopleths** can be plotted using the following code:

```
plot(BHYPR2, type = "Isopleth", xaxis1 = "FM", yaxis1 = "B_R.rel", mark = TRUE, identify = FALSE)
```



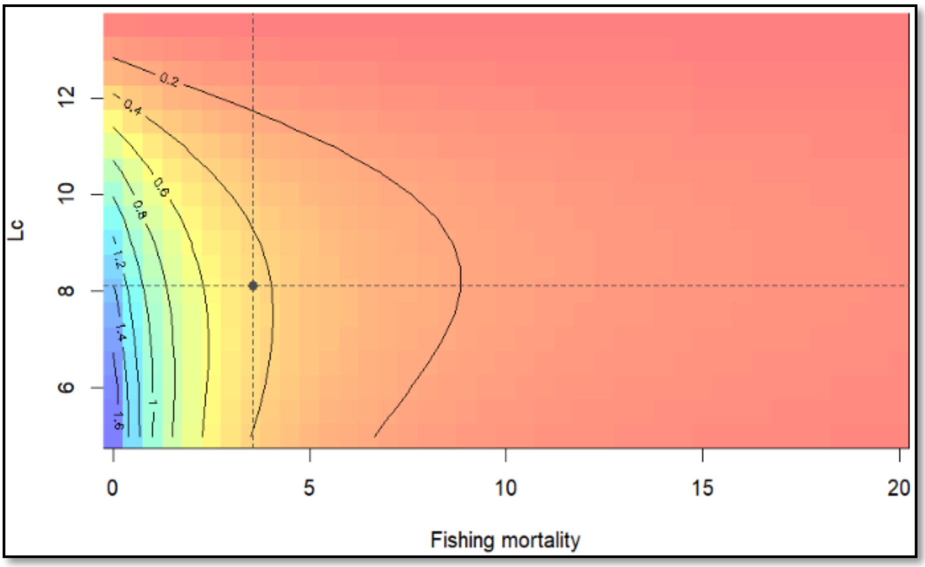
Similarly, once the BHYPR2 outputs are generated, the **Y/R isopleths** can be plotted using the following code:

plot(BHYPR2, type = "Isopleth", xaxis1 = "FM", yaxis1 = "Y_R", mark = TRUE, identify = FALSE)



Similarly, once the BHYPR2 outputs are generated, the **B/R isopleths** can be plotted using the following code:

plot(BHYPR2, type = "Isopleth", xaxis1 = "FM", yaxis1 = "B_R", mark = TRUE, identify = FALSE)



Note: *Identity can be set on (**identity=TRUE**) in the above codes to find out the values of LC and*

FM for any point on the graph just by clicking on it.

Enhanced yield isopleth visualization from BHYPR2 (2D plot)

Step-1: Prepare a new data frame (BHYPR2_output) for the change in the yield (Y/R) in response to the change in FM and LC from the Beverton and Holt's yield per recuit model output (BHYPR2)

```
BHYPR2_output<-as.data.frame(do.call(rbind, BHYPR2$list_Lc_runs))
```

Step-2: Arrange the data for plotting using the following codes:

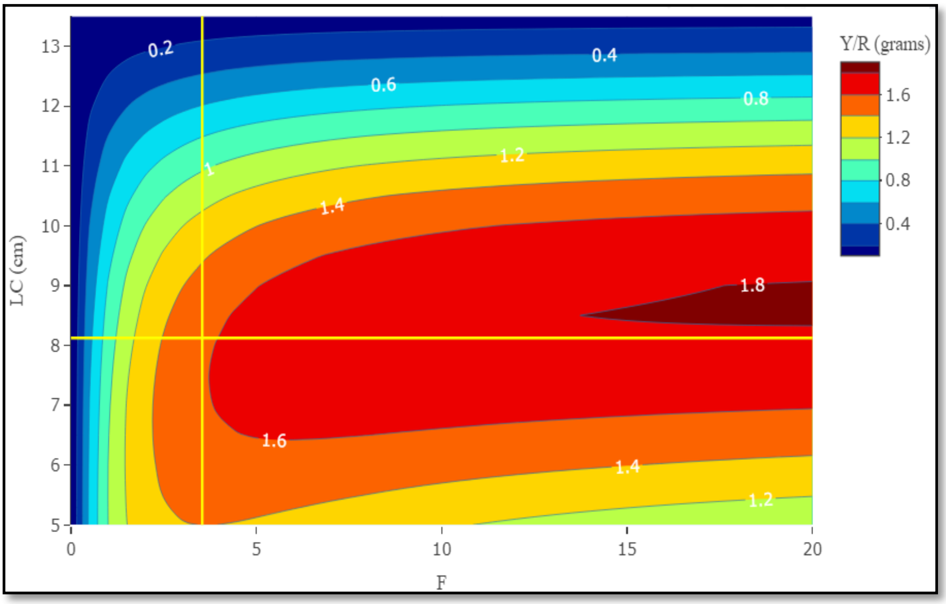
```
LC<-rep(BHYPR2$Lc, each=length(BHYPR2$FM_change))
names(LC)<-"LC"
BHYPR2_output <-cbind(LC, BHYPR2_output)
rownames(BHYPR2_output)<-c(1:length(BHYPR2_output$LC))
```

Step-3: Plot the 2D plot using the following code:

```
library(plotly)
fig<- plot_ly(BH2_output, x=~ BH2_output$FM, y=~ BH2_output$LC, z=~ BH2_output$Y_R) %>% add_trace(type="contour", contours = list(showlabels = TRUE, labelfont = list(size = 12, color = "white")), colorscale = "Jet", colorbar=list(title= list(text="Y/R (grams)", font=list(size="14", family="Times New Roman"))))%>% layout(xaxis=list(title= list(text= "F", font=list(size="14", family="Times New Roman"))), yaxis=list(title= list(text= "LC (cm)", font=list(size="14", family="Times New Roman"))))
fig
```

Step-4: Add the current reference lines using the following codes:

```
vline <- function(x = 0, color = "red") {
list(type = "line", y0 = 0, y1 = 1, yref = "paper", x0 = x, x1 = x, line = list(color = "yellow"))
}
hline <- function(y = 0, color = "blue") {
list(type = "line", x0 = 0, x1 = 1, xref = "paper", y0 = y, y1 = y, line = list(color = "yellow"))
}
#Show the catch curve analysis derived FM and LC on the plot using the following code:
fig%>%layout(shapes = list(vline(CC$FM), hline(CC$L50)))
```



Note: *The colour gradient can be customized by changing the* ***colorscale*** *=* ***"Jet"*** *to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet". Use* ***Y/R.rel*** *in the place of* ***Y/R*** *(****in Step-3****) to plot the graph for relative yield per recruit (Y'/R).*

Enhanced yield isopleth visualization from BHYPR2 (3D plot)

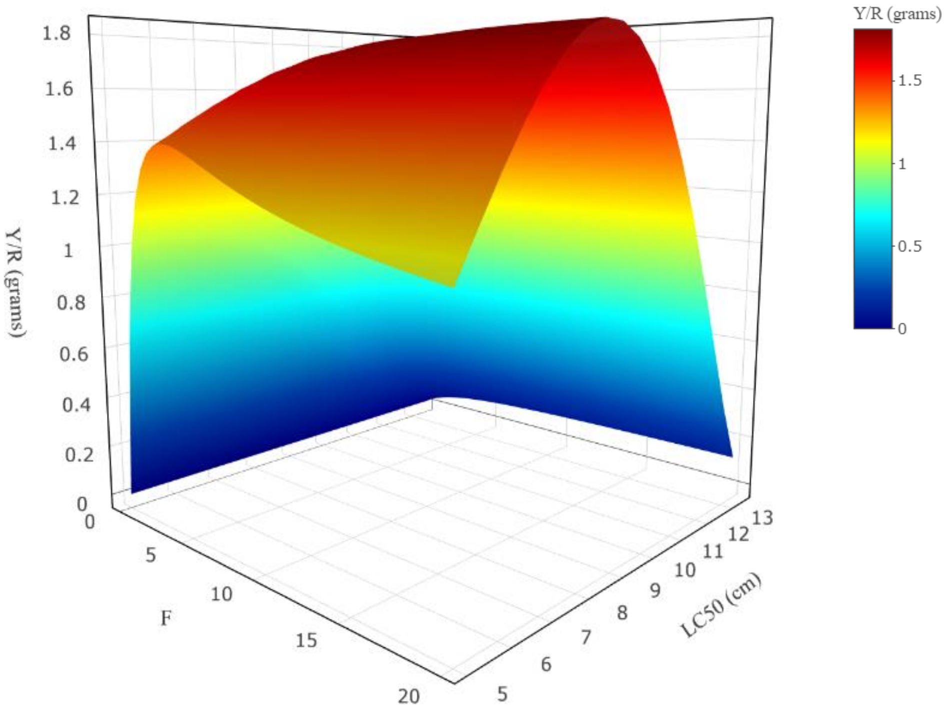
Step-1: Prepare a new data frame (BHYPR2_output) for the change in the yield (Y/R) in response to the change in FM and LC from the Beverton and Holt's yield per recuit model output (BHYPR2) using the above mentioned Step-1 and 2.

Step-2: Prepare a new matrix (Yield_R_change) from the data frame (BHYPR2_output) previously prepared from the Beverton and Holt's yield per recuit model output (BHYPR2) for 3D plot.

```
Yield_R_change<-matrix(c(BHYPR2_output$Y_R), nrow=length(BHYPR2$FM_change), ncol=length(BHYPR2$list_Lc_runs))
rownames(Yield_R_change)<-c(BHYPR2$FM_change)
colnames(Yield_R_change)<-c(BHYPR2$Lc)
```

Step-3: Plot the 3D plot using the following code:

```
library(plotly)
fig<- plot_ly(z= Yield_R_change, x = ~ as.numeric(colnames(Yield_R_change)), y = ~ as.numeric(rownames(Yield_R_change)), type = "surface", opacity = 1.0, colorscale = "Jet", colorbar=list(title= list(text= "Y/R (grams)", font=list(size="14", family="Times New Roman"))))%>% layout(scene=list(xaxis=list(autorange = "reversed", nticks = 10, tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="LC50 (cm)", font=list(size="14", family="Times New Roman"))), yaxis=list(nticks = 10,
tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="F", font=list(size="14", family="Times New Roman"))), zaxis=list(nticks = 10, tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="Y/R (grams)", font=list(size="14", family="Times New Roman")))))
fig
```



> ***Note:*** *The colour gradient can be customized by changing the* ***colorscale = "Jet"*** *to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet". Use* ***Y/R.rel*** *in the place of* ***Y/R*** *(****in Step-2 and 3****) to plot the graph for relative yield per recruit (Y'/R).*

Enhanced biomass isopleth visualization from BHYPR2 (2D plot)

Step-1: Prepare a new data frame (BHYPR2_output) for the change in the yield (Y/R) in response to the change in FM and LC from the Beverton and Holt's yield per recuit model output (BHYPR2).

```
BHYPR2_output<-as.data.frame(do.call(rbind, BHYPR2$list_Lc_runs))
```

Step-2: Arrange the data for plotting using the following codes:

```
LC<-rep(BHYPR2$Lc, each=length(BHYPR2$FM_change))
names(LC)<-"LC"
BHYPR2_output <-cbind(LC, BHYPR2_output)
rownames(BHYPR2_output)<-c(1:length(BHYPR2_output$LC))
```

Step-3: Plot the 2D plot using the following code:

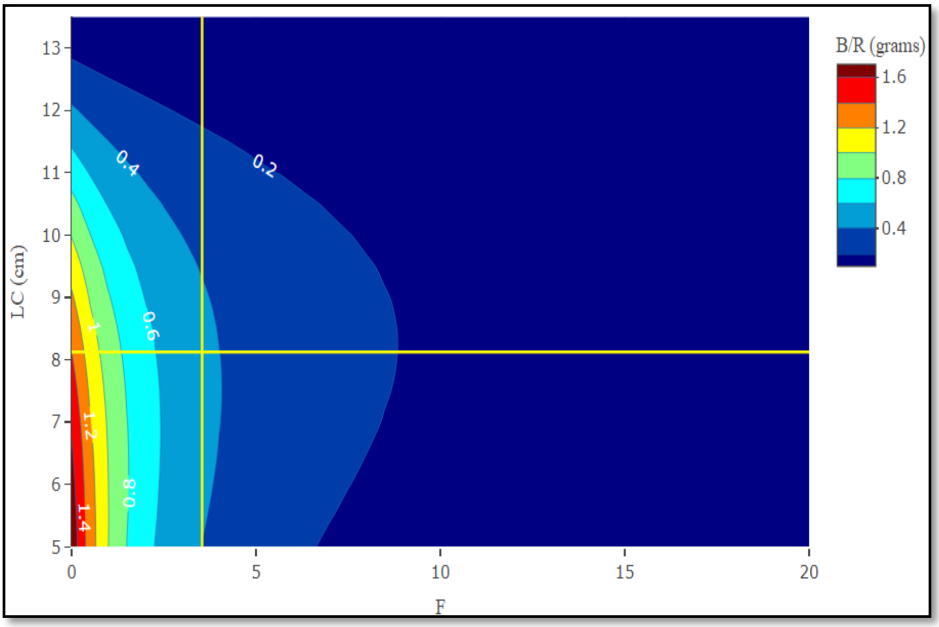
```
library(plotly)
fig<- plot_ly(BHYPR2_output, x=~ BHYPR2_output$FM, y=~ BHYPR2_output$LC, z=~
BHYPR2_output$B_R) %>% add_trace(type="contour", contours = list(showlabels =
TRUE, labelfont = list(size = 12, color = "white")), colorscale = "Jet", colorbar=list(title=
list(text="B/R (grams)", font=list(size="14", family="Times New Roman"))))%>%
layout(xaxis=list(title= list(text= "F", font=list(size="14", family="Times New Roman"))),
yaxis=list(title= list(text= "LC (cm)", font=list(size="14", family="Times New Roman"))))
fig
```

Step-4: Add the current reference lines using the following codes:

```
vline <- function(x = 0, color = "red") {
list(type = "line", y0 = 0, y1 = 1, yref = "paper", x0 = x, x1 = x, line = list(color = "yellow"))
}
hline <- function(y = 0, color = "blue") {
list(type = "line", x0 = 0, x1 = 1, xref = "paper", y0 = y, y1 = y, line = list(color = "yellow"))
}
```

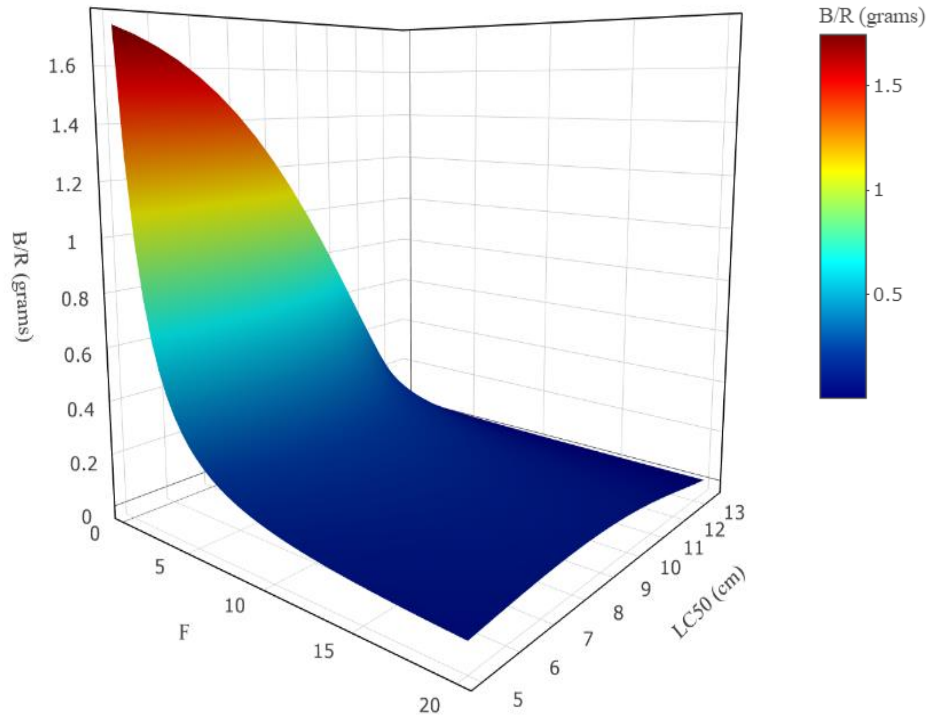
#Show the catch curve analysis derived FM and LC on the plot using the following code:

```
fig%>%layout(shapes = list(vline(CC$FM), hline(CC$L50)))
```



***Note: :** The colour gradient can be customized by changing the **colorscale = "Jet"** to other colour scales like "Viridis", "Rainbow", "RdBu" in place of "Jet". Use **B/R.rel** in the place of **B/R** (**in Step-3**) to plot the graph for relative yield per recruit (B'/R).*

Enhanced biomass isopleth visualization from BHYPR2 (3D plot)



> ***Note:*** *The colour gradient can be customized by changing the* ***colorscale = "Jet"*** *to other colour schemes like "Viridis", "Rainbow", "RdBu" in place of "Jet". Use* ***B/R.rel*** *in the place of* ***B/R (in Step-2 and 3****) to plot the graph for relative yield per recruit (B'/R).*

Step-1: Prepare a new data frame (BHYPR2_output) for the change in the yield (Y/R) in response to the change in FM and LC from the Beverton and Holt's yield per recuit model output (BHYPR2) using the above mentioned Step-1 and 2.

Step-2: Prepare a new matrix (Yield_R_change) from the data frame (BHYPR2_output) previously prepared from the Beverton and Holt's yield per recuit model output (BHYPR2) for 3D plot.

```
Biomass_R_change<-matrix(c(BHYPR2_output$B_R),
nrow=length(BHYPR2$FM_change), ncol=length(BHYPR2$list_Lc_runs))
rownames(Biomass_R_change)<-c(BHYPR2$FM_change)
colnames(Biomass_R_change)<-c(BHYPR2$Lc)
```

Step-3: Plot the 3D plot using the following code:

```
library(plotly)
```

```
fig<- plot_ly(z= Biomass_R_change, x = ~ as.numeric(colnames(Biomass_R_change)), y =
~ as.numeric(rownames(Biomass_R_change)), type = "surface", opacity = 1.0, colorscale
= "Jet", colorbar=list(title= list(text= "B/R (grams)", font=list(size="14", family="Times
New Roman"))))%>% layout(scene=list(xaxis=list(autorange = "reversed", nticks = 10,
tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="LC50
(cm)", font=list(size="14", family="Times New Roman"))), yaxis=list(nticks = 10,
tickangle= 0, linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="F",
font=list(size="14", family="Times New Roman"))), zaxis=list(nticks = 10, tickangle= 0,
linecolor="black", linewidth=2, showline=T, mirror = T, title= list(text="B/R (grams)",
font=list(size="14", family="Times New Roman")))))
fig
```

2.11. Understanding the stock simulation outputs (fisheries management reference points)

F01 and E01: The fishing mortality rate (or F-multiplier if FM_relative = TRUE) and the corresponding exploitation at which the rate of increase in yield is only 10% of the highest rate of increase. It serves as a precautionary reference point to reduce the risk of growth overfishing.

YPR_F0.1: The yield that would be produced if F01 is implemented.

BPR_F0.1: The biomass that would remain if F01 is implemented.

SPR_F0.1: The spawning potential ratio, i.e., spawning stock biomass ratio (SSB/SSB$_0$) that would remain if F01 is implemented.

Fmax and Emax: The fishing mortality rate (or F-multiplier if FM_relative = TRUE) and the corresponding exploitation rate at which the yield is maximum. It is an optimistic reference point that requires careful evaluation, as it may lead to very low biomass and overfishing.

YPR_Fmax: The yield that would be produced if Fmax is implemented.

BPR_Fmax: The biomass that would remain if Fmax is implemented.

SPR_Fmax: The spawning potential ratio, i.e., spawning stock biomass ratio (SSB/SSB$_0$) that would remain if Fmax is implemented.

F05 and E05: The fishing mortality rate (or F-multiplier if FM_relative = TRUE) and the corresponding exploitation rate at which the biomass is at 50% of the virgin state biomass level (B/B_0=0.50). It is a conservative reference point that ensures 50% of the biomass remains available, reducing the risk of overfishing.

YPR_F05: The yield that would be produced if F05 is implemented.

BPR_F05: The biomass that would remain if F05 is implemented.

SPR_F05: The spawning potential ratio, i.e., spawning stock biomass ratio (SSB/SSB$_0$) that would remain if F05 is implemented.

F04 and E04: The fishing mortality rate (or F-multiplier if FM_relative = TRUE) and the corresponding exploitation rate at which the spawning stock biomass (SSB) is at 40% of the virgin SSB level ($SSB/SSB_0=0.40$). It is a precautionary reference point that ensures the availability of 40% spawning stock biomass to reduce the chance of recruitment overfishing.

YPR_F04: The yield that would be produced if F04 is implemented.

BPR_F04: The biomass that would remain if F04 is implemented.

SPR_F04: The spawning potential ratio, i.e., spawning stock biomass ratio (SSB/SSB_0) that would remain if F04 is implemented.

Understanding the stock simulation outputs (current stock status)

Curr.LC	**Current length at capture (LC_{50})**	**Curr.tC**	**Current age at capture (tC_{50})**
Curr.E	Current exploitation ratio (E_{cur})	**Curr.F**	Current fishing mortality rate (F_{cur})
Curr.Catch	Current catch (C_{cur}) obtained at F_{cur}	**Curr.Yield**	Current yield (Y_{cur}) obtained at F_{cur}
Curr.Revenue	Current revenue (R_{cur}) obtained at F_{cur}	**Curr.Biomass**	Current biomass (B_{cur}) present at F_{cur}
Curr.SSB	Current SSB (SSB_{cur}) present at F_{cur}	**SPR**	Current spawning potential ration (SSB_{cur}/SSB_0) present at F_{cur}

References

Journal articles

Alagaraja, K. (1984). Simple methods for estimation of parameters for assessing exploited fish stocks. *Indian Journal of Fisheries, 31*, 177–208. https://epubs.icar.org.in/index.php/IJF/article/view/11722

Alverson, D. L., & Carney, M. J. (1975). A graphic review of the growth and decay of population cohorts. *Journal du Conseil International pour l'Exploration de la Mer, 36*, 133–143. https://doi.org/10.1093/icesjms/36.2.133

Charnov, E. L., Gislason, H., & Pope, J. G. (2012). Evolutionary assembly rules for fish life histories. *Fish and Fisheries*. https://doi.org/10.1111/j.1467-2979.2012.00467.x

Derzhavin, A. N. (1922). The stellate sturgeon (Acipenser stellatus Pallas), a biological sketch. Bulletin of the Baku Ichthyological Station, 1, 1–393. (In Russian)

Fry, F. E. J. (1949). Statistics of a lake trout fishery. *Biometrics, 5*, 27–67.

Gislason, H., Daan, N., Rice, J. C., & Pope, J. G. (2010). Size, growth, temperature and the natural mortality of marine fish. *Fish and Fisheries, 11*, 149–158. https://doi.org/10.1111/j.1467-2979.2009.00350.x

Gunderson, D. R. (1997). Trade-off between reproductive effort and adult survival in oviparous and viviparous fishes. *Canadian Journal of Fisheries and Aquatic Sciences, 54*, 990–998. https://doi.org/10.1139/f97-019

Gunderson, D. R., & Dygert, P. H. (1988). Reproductive effort as a predictor of natural mortality rate. *Journal du Conseil International pour l'Exploration de la Mer, 44*, 200–209. https://doi.org/10.1093/icesjms/44.2.200

Hamel, O. S. (2015). A method for calculating a meta-analytical prior for the natural mortality rate using multiple life history correlates. *ICES Journal of Marine Science, 72*, 62–69. https://doi.org/10.1093/icesjms/fsu131

Hewitt, D., & Hoenig, J. (2005). Comparison of two approaches for estimating natural mortality based on longevity. *Fishery Bulletin, 103*, 433–437. https://spo.nmfs.noaa.gov/sites/default/files/pdf-content/2005/1032/hewitt.pdf

Hilborn, R., & Ovando, D. (2014). Reflections on the success of traditional fisheries management. *ICES Journal of Marine Science, 71*, 1040–1046. https://doi.org/10.1093/icesjms/fsu034

Hoenig, J. M. (1983). Empirical use of longevity data to estimate mortality rates. *Fishery Bulletin, 82*, 898–903. https://spo.nmfs.noaa.gov/sites/default/files/pdf-content/fish-bull/hoenig.pdf

Jensen, A. L. (1996). Beverton and Holt life history invariants result from optimal trade-off of reproduction and survival. *Canadian Journal of Fisheries and Aquatic Sciences, 53*, 820–822. https://doi.org/10.1139/f95-233

Kenchington, T. J. (2014). Natural mortality estimators for information-limited fisheries. *Fish and Fisheries, 15*, 533–562. https://doi.org/10.1111/faf.12027

Lorenzen, K. (1996). The relationship between body weight and natural mortality in juvenile and adult fish: a comparison of natural ecosystems and aquaculture. *Journal of Fish Biology, 49*, 627–647. https://doi.org/10.1111/j.1095-8649.1996.tb00060.x

Maunder, M. N., Hamel, O. S., Lee, H.-H., Piner, K. R., Cope, J. M., Punt, A. E., ... Methot, R. D. (2023). A review of estimation methods for natural mortality and their performance in the context of fishery stock assessment. *Fisheries Research, 25*, 106489. https://doi.org/10.1016/j.fishres.2022.106489

Mildenberger, T. K., Taylor, M. H., & Wolff, M. (2017). TropFishR: An R package for fisheries analysis with length-frequency data. *Methods in Ecology and Evolution, 8*(11), 1520–1527. https://doi.org/10.1111/2041-210X.12791

Pauly, D. (1980). On the interrelationships between natural mortality, growth parameters and mean environmental temperature in 175 fish stocks. *Journal du Conseil International pour l'Exploration de la Mer, 39*(3), 175–192. https://s3-us-west-2.amazonaws.com/legacy.seaaroundus/doc/Researcher+Publications/dpauly/PDF/1980/Journal+Articles/OnInterrelationshipNaturalMortalityGrowthParametersAndMeanEn.pdf

Peterson, I., & Wroblewski, J. S. (1984). Mortality rate of fishes in the pelagic ecosystem. *Canadian Journal of Fisheries and Aquatic Sciences, 41*, 1117–1120. https://doi.org/10.1139/f84-131 26.

Pope, J.G., 1972. An investigation of the accuracy of virtual population analysis using cohort analysis. ICNAF Res. Bull., (9): 65-74.

Then, A. Y., Hoenig, J. M., Hall, N. G., & Hewitt, D. A. (2015). Evaluating the predictive performance of empirical estimators of natural mortality rate using information on over 200 fish species. *ICES Journal of Marine Science, 72*(1), 82–92. https://doi.org/10.1093/icesjms/fsu136

Wang, K., Zhang, C., Xu, B., Xue, Y., & Ren, Y. (2020). Selecting optimal bin size to account for growth variability in Electronic Length Frequency Analysis (ELEFAN). *Fisheries Research, 225*, 105474. https://doi.org/10.1016/j.fishres.2019.105474

Zhang, C. I., & Megrey, B. A. (2006). A revised Alverson and Carney model for estimating the instantaneous rate of natural mortality. *Transactions of the American Fisheries Society, 135*, 620–633. https://doi.org/10.1577/T04-173.1

Books

Longhurst, A. R., & Pauly, D. (1987). Ecology of tropical oceans. Academic Press, San diego, 407 p. https://api.pageplace.de/preview/DT0400.9780323140638_A23647441/preview-9780323140638_A23647441.pdf

Quinn, T. J., & Deriso, R. B. (1999). Quantitative fish dynamics. Oxford University Press, 542 p.

Ricker, W. E. (1958). Hand book of computations for biological statistics of fish populations. Bull. Fish. Res. Bd Canada., 119: 300 p.

Ricker, W. E. (1975). Computation and interpretation of biological statistics of fish populations. Bulletin of the Fisheries Research Board of Canada, Bulletin 191, 382 p. https://waves-vagues.dfo-mpo.gc.ca/Library/1485.pdf

Roff, D. A. (1992). The evolution of life histories. Chapman and Hall New York, NY, 535 p.

Technical Documents

Gulland, J. A. (1965). Estimation of mortality rates. Annex to Arctic Fisheries Working Group Report. ICES C.M. Doc. 3 (mimeo).

Jones, R. (1984). Assessing the effects of changes in exploitation pattern using length composition data. FAO Fisheries Technical Paper, 256, 118 pp.

Pauly, D. (1983). Some simple methods for the assessment of tropical fish stocks. FAO Fisheries Technical Paper, No. 234. FAO, Rome. 52p.

Pauly, D., & Morgan, G. R. (Eds.). (1987). Length-based methods in fisheries research. Conference Proceedings No. 13. International Center for Living Aquatic Resources Management, Manila, Philippines.

Pauly, D., Soriano-Bartz M., Moreau J. and Jarre A.. 1992. A new model accounting for seasonal cessation of growth in fishes. Aust. J. Mar. Freshwat. Res. 43:1151-1156.

Somer, I.F. 1988. On a seasonally oscillating growth function. Fishbyte 6(1):8-11.

Sparre, P., & Venema, S. C. (1998). Introduction to tropical fish stock assessment. Part 1. Manual (FAO Fisheries Technical Paper No. 306.1, Rev. 2). FAO. 407p.

Rikhter, V. A., & Efanov, V. N. (1976). On one of the approaches to estimation of natural mortality of fish populations. ICNAF Research Document, 76/VI/8, 12p.

Online resources

R Core Team. (n.d.). TropFishR: lfqData vignette. https://cran.r-project.org/web/packages/TropFishR/vignettes/lfqData.html

R Core Team. (n.d.). TropFishR: tutorial vignette. https://cran.r-project.org/web/packages/TropFishR/vignettes/tutorial.html

FAO. (n.d.). Fisheries publications and reports. https://openknowledge.fao.org/server/api/core/bitstreams/bc7c37b6-30df-49c0-b5b4-8367a872c97e/content

FAO. (n.d.). Fisheries publications and reports. https://www.fao.org/fishery/en/publications/query/Pseudojuloides%20edwardi,Mombasa%20pencil%20wrasse

2.12. Length-Weight Relationship (LWR)

Introduction

The length-weight relationship (LWR) is essential for converting length-based numerical observations into biomass estimates, which is crucial for biomass modeling and stock simulation in the aquatic ecosystem. The relationship between length and weight of fish can be expressed in the following two commonly used equation forms (Keys, 1928; Clark, 1928; Froese, 2006):

$$Linear\ Model,\ log(W) = log(a) + b \times log(L) + \varepsilon, \varepsilon \approx normal\ (0, \sigma^2) \ldots\ldots (1)$$

$$Nonlinear\ Model,\ W = a \times L^b + \varepsilon, \varepsilon \approx normal\ (0, \sigma^2) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Essentially, the choice of model depends primarily on the variance or residual distribution structure of the model. The log-transformed linear model assumes an underlying multiplicative log-normal variance (error) distribution on an untransformed scale, while the simple nonlinear model assumes an additive normal variance (error) distribution on the original (untransformed) scale (Xiao et al., 2011; and De Giosa and Czerniejewski, 2016). Since fish grow in all the three dimensions, their weight and variance associated with the weight also increases with length (Cawley and Janacek, 2010; and Xiao et al., 2011). Therefore, it is biologically reasonable to assume a log-normal multiplicative error structure, and therefore the log-transformed linear model (shown in equation 1) is frequently used in the LWRs (Ogle, 2015). However, to improve the accuracy of the LWR, it is essential to understand the variance/error distribution structure to correctly apply the appropriate modeling approach (Xiao et al., 2011; and Dash et al., 2023). The variance distribution structure can be found out by fitting the collected length and weight data using both the linear and nonlinear modeling approaches and then performing a multi-model comparison. For the multi-model comparison, a likelihood-based information theoretic criterion such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) can be used (Xiao et al., 2011; De Giosa and Czerniejewski, 2016; and Dash et al., 2023). A derivative of AIC (the small-sample equivalent of AIC), i.e., AICc is also often recommended for model selection when the ratio between the sample size (n) and the number of modeling parameters (k), i.e., n/k is less than 40 (Sugiura, 1978; and Hurvich and Tsai, 1995). The modeling approach with lower AIC or BIC score is considered being better compared to the other competing model(s) and thus explains better support for its underlying assumption about variance distribution structure.

Length-Weight Relationship: R Implementation

2.12.1. Requirement for Length-Weight Relationship (LWR)

Installing and loading dependent R-packages

Install the dependent R-packages (Do not install again if already installed).

```
install.packages("nlstools")
install.packages("propagate")
install.packages("boot")
install.packages("caret")
install.packages("car")
install.packages("ggplot2")
```

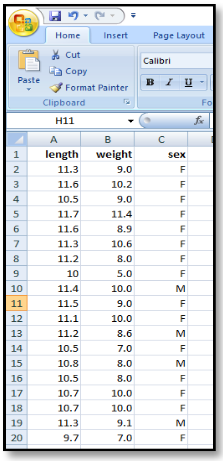
Load the dependent R-packages

```
library(nlstools)
library(propagate)
library(boot)
library(caret)
library(car)
library(ggplot2)
```

Length-weight data

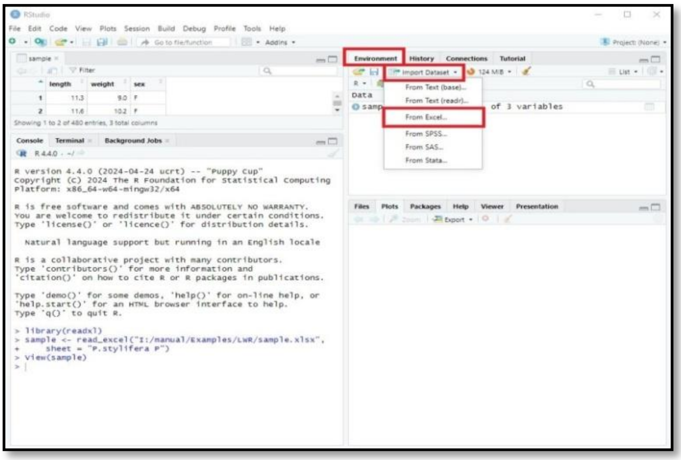
The length-weight data should be available in following format for the analysis.

Length-Weight data (e.g., sample)



	A	B	C
1	length	weight	sex
2	11.3	9.0	F
3	11.6	10.2	F
4	10.5	9.0	F
5	11.7	11.4	F
6	11.6	8.9	F
7	11.3	10.6	F
8	11.2	8.0	F
9	10	5.0	F
10	11.4	10.0	M
11	11.5	9.0	F
12	11.1	10.0	F
13	11.2	8.6	M
14	10.5	7.0	F
15	10.8	8.0	M
16	10.5	8.0	F
17	10.7	10.0	F
18	10.7	10.0	F
19	11.3	9.1	M
20	9.7	7.0	F

Excel Import window in R



	length	weight	sex
1	11.3	9.0	F
2	11.6	10.2	F

Showing 1 to 2 of 480 entries, 3 total columns

```
R version 4.4.0 (2024-04-24 ucrt) -- "Puppy Cup"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(readxl)
> sample <- read_excel("I:/manual/Examples/LWR/sample.xlsx",
+     sheet = "P.stylifera P")
> View(sample)
> 
```

Refer ***'Example data file download link'*** in the last page to download and use the example data.

2.12.2. Importing length-weight data to R Interface

The above length-weight data needs to be imported to the R interface. Click the Import Dataset of the Environment tab (top right side panel) and then select from Excel. Browse the Excel file (ex: 'rawdata' sheet of 'rawdata' excel file) and then import.

Environment> Import Dataset> from Excel and then browse the file on disk and import.

Define the two measurement variables, i.e., x for the independent measurement variable (here, length of the fish, preferably in 'cm') and 'y' for the dependent measurement or response variable (here, weight of the fish, preferably in 'g'). Additionally, 'sex' of fish can be assigned as a categorical variable (as factor) for ANCOVA to observe the difference in weights between the sexes.

```
x<- as.vector(LWR_data$length)
y<- as.vector(LWR_data$weight)
sex<-as.factor(LWR_data$sex)
```

2.12.3. Modeling the relationship between length and weight (Linear Model)

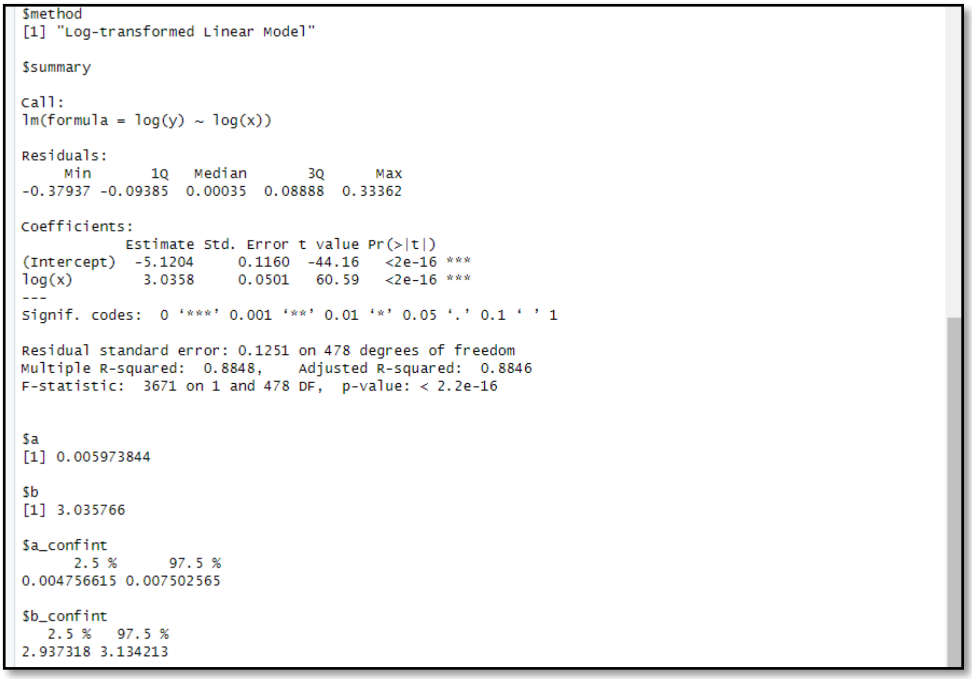
Linear modeling

Most commonly used approach is to log transform the measurement variables and then fit a linear regression assuming a biologically reasonable log-normal multiplicative error structure for the residuals. The model can be fitted using the following codes:

```
linear_model = lm(log(y) ~ log(x))
a_LM = exp(coef(summary(linear_model))[1, 1])
b_LM = coef(summary(linear_model))[2, 1]
sd_LM = sd(log(y) - (log(a_LM) + b_LM * log(x)))
list(method = "Log-transformed Linear Model", summary = summary(linear_model), a = a_LM, b = b_LM, a_confint = exp(confint(linear_model)[1, ]), b_confint = confint(linear_model)[2, ])
```

Understanding the output of linear modeling

The ***summary(linear_model)*** code produces a summary of the model (shown below). Since the data used in the model are in a log-transformed scale, the first coefficient, i.e., intercept (a) produce by the code is in log form (here, -5.1204) which should be exponentiated (exp(coef(summary(linear_model))[1, 1])) to express the value in original untransformed scale (i.e., 0.00597). Apart from this the summary also produces standard error (SE), t-statistics and p-value for each of the model coefficients. The code also produces residual standard error (RSE), degrees of freedom (DF), R-squared, F-statistics, and p-value for the model. The confint(linear_model) code produces a 95% confidence interval of the model parameters (a & b). However, the confidence interval of intercept (a) produce by the code is in log form which should be exponentiated (exp(confint(linear_model)[1,])) to express the value in original untransformed scale. The model assumptions can be checked using model diagnosis tools (Refer to 2.12.6.Model Diagnosis). The log-transformed linear model is adequate enough to establish the LWR in most of the cases. However, to improve the accuracy of the model coefficient, it is better to model the relationship using a nonlinear model and then evaluate the variance distribution structure of both the models to arrive at best modeling approach.



```
$method
[1] "Log-transformed Linear Model"

$summary

Call:
lm(formula = log(y) ~ log(x))

Residuals:
     Min       1Q   Median       3Q      Max 
-0.37937 -0.09385  0.00035  0.08888  0.33362 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -5.1204     0.1160  -44.16   <2e-16 ***
log(x)        3.0358     0.0501   60.59   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1251 on 478 degrees of freedom
Multiple R-squared:  0.8848,	Adjusted R-squared:  0.8846 
F-statistic:  3671 on 1 and 478 DF,  p-value: < 2.2e-16


$a
[1] 0.005973844

$b
[1] 3.035766

$a_confint
      2.5 %      97.5 % 
0.004756615 0.007502565 

$b_confint
   2.5 %   97.5 % 
2.937318 3.134213 
```

2.12.4. Modeling the relationship between length and weight (Nonlinear Model)

Nonlinear modeling

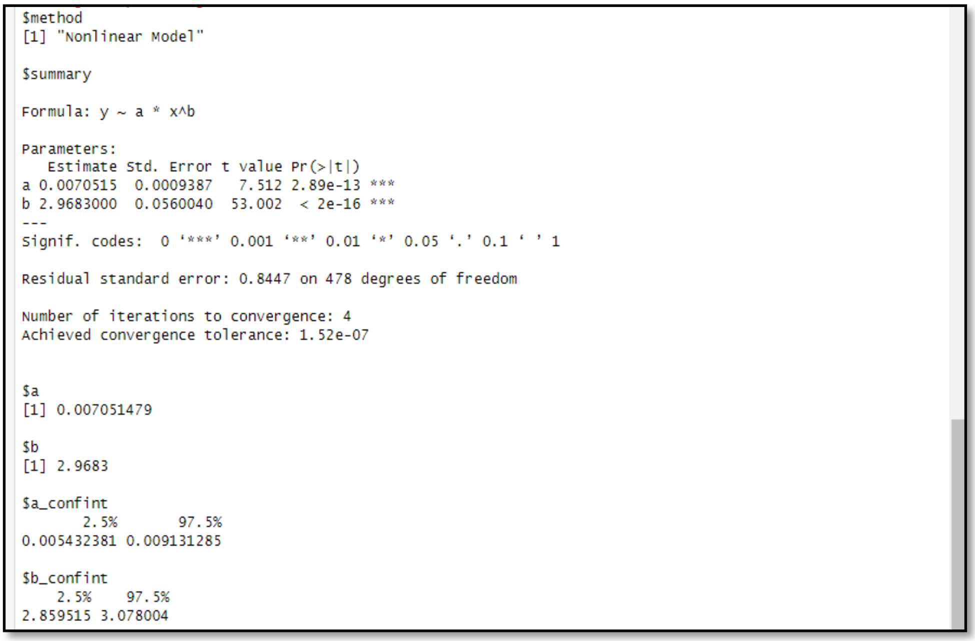
When it is suspected that residuals of the model may follow an additive normal variance (error) distribution structure in the original (untransformed) scale, it is advisable to conduct a nonlinear modeling approach. The support for the correct variance distribution structure can be found out later using an information theoretic criterion like AIC or BIC (Xiao et al., 2011; and Dash et al., 2023). The model can be fitted using the following codes:

```
nonlinear_model = nls(y ~ a* x ^ b, start = list(a = a_LM, b = b_LM), control = nls.control(maxiter = 2000, warnOnly = TRUE))

a_NLM = coef(summary(nonlinear_model))[1, 1]

b_NLM = coef(summary(nonlinear_model))[2, 1]

sd_NLM = sd(y - a_NLM * x ^ b_NLM)
```

```
list(method = "Nonlinear Model", summary = summary(nonlinear_model), a = a_NLM, b = b_NLM, a_confint = confint(nonlinear_model)[1, ], b_confint = confint(nonlinear_model)[2, ])
```

Understanding the output of nonlinear modeling

The ***summary(nonlinear_model)*** code produces a summary of the model (shown below). Since the data used in the model are in an original untransformed scale, both the coefficients (a & b) produced by the code are in the original scale. Apart from this the summary also produces standard error (SE), t-statistics and p-value for each of the model coefficients. Apart from this the summary also produces residual standard error (RSE), and degrees of freedom (DF) for the model. Since it is a nonlinear model, the R-squared values are not produced by the model because of lack of relevance. The confint(nonlinear_model) code produces a 95% confidence interval of the model coefficients (a & b) in the original untransformed scale. The model assumptions can be checked using model diagnosis tools (Refer to 2.12.6.Model Diagnosis).



```
$method
[1] "Nonlinear Model"

$summary

Formula: y ~ a * x^b

Parameters:
   Estimate Std. Error t value Pr(>|t|)    
a 0.0070515  0.0009387   7.512 2.89e-13 ***
b 2.9683000  0.0560040  53.002  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8447 on 478 degrees of freedom

Number of iterations to convergence: 4 
Achieved convergence tolerance: 1.52e-07


$a
[1] 0.007051479

$b
[1] 2.9683

$a_confint
       2.5%       97.5% 
0.005432381 0.009131285 

$b_confint
    2.5%    97.5% 
2.859515 3.078004 
```

2.12.5. Selecting the best modeling approach

In most of the cases, the log-transformed linear model is adequate enough to establish the LWR. However, to improve the accuracy of the model coefficient, it is better to model the relationship using a nonlinear model and then evaluate the variance distribution structure of both the models to arrive at best modeling approach. The support for the correct variance distribution structure can be found out using an information theoretic criterion like AIC or BIC (Xiao et al., 2011; and Dash et al., 2023). In the present example, the small-sample equivalent of AIC, i.e., AICc, has been used to evaluate the variance distribution structure. It has been recommended to use AICc for small sample size (especially, when the ratio between sample size (n) and number of modeling parameters (k), i.e., n/k is less than 40) (Sugiura, 1978; and Hurvich and Tsai, 1995). In has also been noted that the AICc provides stronger penalty compared to the AIC and BIC for small and very small sample size, respectively (Brewer et al., 2016). Based on the AICc following criteria is used to select the best modeling approach.

Criteria for model selection

if delta_AICc < -2*: The assumption of additive normal error is better supported so proceed with nonlinear model results

if delta_AICc > 2*: The assumption of multiplicative log-normal error is better supported, so proceed with linear model results

if – 2* ≥ delta_AICc ≤ 2*: The two error distributions have similar support and so proceed with weighted average mixed model

*the cutoff value of 2 has been recommended by Burnham and Anderson (2002). Despite these rules of thumb, there is considerable ambiguity regarding the treatment (acceptance or rejection) of competing models if the delta_AICc falls within the doubtful zone of 4-7. According to Dash et al. (2023b), it has been recommended to use a cutoff value of 4.2 for model selection. Check with the residual diagnostic plot and residual density plot for visual confirmation with the Shapiro-Wilk test of normality of residuals.

Calculate AICc and their difference (delta_AICc)

```
likelihood_lognormal = sum(log(dlnorm(y, log(a_LM * x ^ b_LM), sd_LM)))
likelihood_normal = sum(log(dnorm(y, a_NLM * x ^ b_NLM, sd_NLM)))
n = length(x)
k=length(coefficients(linear_model))+1
AICc_lognormal = 2 * k - 2 * likelihood_lognormal + 2 * k * (k + 1) / (n - k - 1)
AICc_normal = 2 * k - 2 * likelihood_normal + 2 * k * (k + 1) / (n - k - 1)
delta_AICc = AICc_normal - AICc_lognormal
```

AICc comparison

The below-mentioned model comparison codes have been designed to automatically select the best modeling approach and the model summary based on the above criteria. If the visual diagnostics like residual diagnostic plots and residual density plots of residuals do not provide sufficient visual cues with Shapiro-Wilk test of normality for model selection at **delta_AICc < - 2,** then increase it to **delta_AICc < - 4** at both the below-highlighted places.

```
if (delta_AICc < - 2){
writeLines ("Better support for additive normal error assumption. Go for nonlinear power regression NLR")
list(method = "Nonlinear Model", summary = summary(nonlinear_model), a = a_NLM, b = b_NLM, a_confint = confint(nonlinear_model)[1, ], b_confint = confint(nonlinear_model)[2, ], AICc = AICc_normal, RMSE = RMSE(predict(nonlinear_model), y))
```

```
 } else if (delta_AICc > 2){
writeLines("Better support for multiplicative log-normal error assumption. Go for log-transformed linear regression LR.")
list(method = "Log-transformed Linear Model", summary = summary(linear_model), a = a_LM, b = b_LM, a_confint = exp(confint(linear_model)[1, ]), b_confint = confint(linear_model)[2, ], AICc = AICc_lognormal, RMSE = RMSE(exp(predict(linear_model)), y))
 } else {
writeLines ("Equal support for both the multiplicative log-normal error and additive normal error assumptions. Go for model averaging MA. Now attempting model averaging to get mean and confidence interval of coefficients by boot strapping")
 library(boot)
{
# Attempting model averaging to get mean and confidence interval of coefficients by boot strapping
boot.est=function(dat, indices) {
dat.sub=dat[indices, ]
names(dat.sub) = c("x", "y")
linear_model_bs = lm(log(y) ~ log(x), dat = dat.sub)
a_LM_bs = exp(coef(summary(linear_model_bs))[1, 1])
b_LM_bs = coef(summary(linear_model_bs))[2, 1]
sd_LM_bs = sd(log(dat.sub$y) - (log(a_LM_bs) + b_LM_bs * log(dat.sub$x)))
a_LM_bs.CI = confint(linear_model_bs)[1, ]
b_LM_bs.CI = confint(linear_model_bs)[2, ]
nonlinear_model_bs = nls(y ~ a * x ^ b, start = list(a = a_LM_bs, b = b_LM_bs), dat = dat.sub,
control = nls.control(maxiter = 2000, warnOnly = TRUE))
a_NLM_bs = coef(summary(nonlinear_model_bs))[1, 1]
b_NLM_bs = coef(summary(nonlinear_model_bs))[2, 1]
sd_NLM_bs = sd(dat.sub$y - a_NLM_bs * dat.sub$x ^ b_NLM_bs)
a_NLM_bs.CI = confint(nonlinear_model_bs)[1, ]
b_NLM_bs.CI = confint(nonlinear_model_bs)[2, ]
likelihood_lognormal_bs = sum(log(dlnorm(dat.sub$y, log(a_LM_bs * dat.sub$x ^ b_LM_bs), sd_LM_bs)))
likelihood_normal_bs = sum(log(dnorm(dat.sub$y, a_NLM_bs * dat.sub$x ^ b_NLM_bs, sd_NLM_bs)))
n = length(x)
k = 3
AICc_lognormal_bs = 2 * k - 2 * likelihood_lognormal_bs + 2 * k * (k + 1) / (n - k - 1)
AICc_normal_bs = 2 * k - 2 * likelihood_normal_bs + 2 * k * (k + 1) / (n - k - 1)
AICc.min = min(AICc_lognormal_bs, AICc_normal_bs)
weight_lognormal_bs = exp(-(AICc_lognormal_bs - AICc.min)/2)
weight_normal_bs = exp(-(AICc_normal_bs - AICc.min)/2)
lognormal_weightage_bs = weight_lognormal_bs / (weight_lognormal_bs + weight_normal_bs)
normal_weightage_bs = weight_normal_bs / (weight_lognormal_bs + weight_normal_bs)
a_boot = a_LM_bs * lognormal_weightage_bs + a_NLM_bs * normal_weightage_bs
b_boot  = b_LM_bs * lognormal_weightage_bs + b_NLM_bs * normal_weightage_bs
return(c(a_boot, b_boot ))
}
dat.boot=boot(data = as.data.frame(cbind(x, y)), statistic = boot.est, R = 1000)
a_confint_boot = boot.ci(dat.boot, index = 1, type = "perc")$perc[4:5]
b_confint_boot = boot.ci(dat.boot, index = 2, type = "perc")$perc[4:5]
}
#Calculating AICc for model averaging
a_MA =dat.boot$t0[1]
b_MA=dat.boot$t0[2]
sd_MA = sd(y- a_MA* x ^ b_MA)
n = length(x)
k=3
likelihood_MA = sum(log(dnorm(y, a_MA * x ^ b_MA, sd_MA)))
AICc_MA = 2 * k - 2 * likelihood_MA + 2 * k * (k + 1) / (n - k - 1)
#Calculating RMSE for model averaging
library(caret)
RMSE_MA =RMSE(a_MA *x^ b_MA, y)
list(method = "Model Averaging", a = dat.boot$t0[1], b = dat.boot$t0[2], a_confint = a_confint_boot, b_confint = b_confint_boot, AICc = AICc_MA, RMSE = RMSE_MA)
}
```

Note: *Depending on the model selection criterion, it will recommend the best modeling approach, i.e., log-transformed linear model or nonlinear model and will also produce the results from the appropriate model. If there is a tie or model ambiguity, then it will try for model averaging and estimation of confidence interval by bootstrapping. It must be noted that all the dependent R-package should be installed prior to execution of the codes without which error messages will be flagged.*

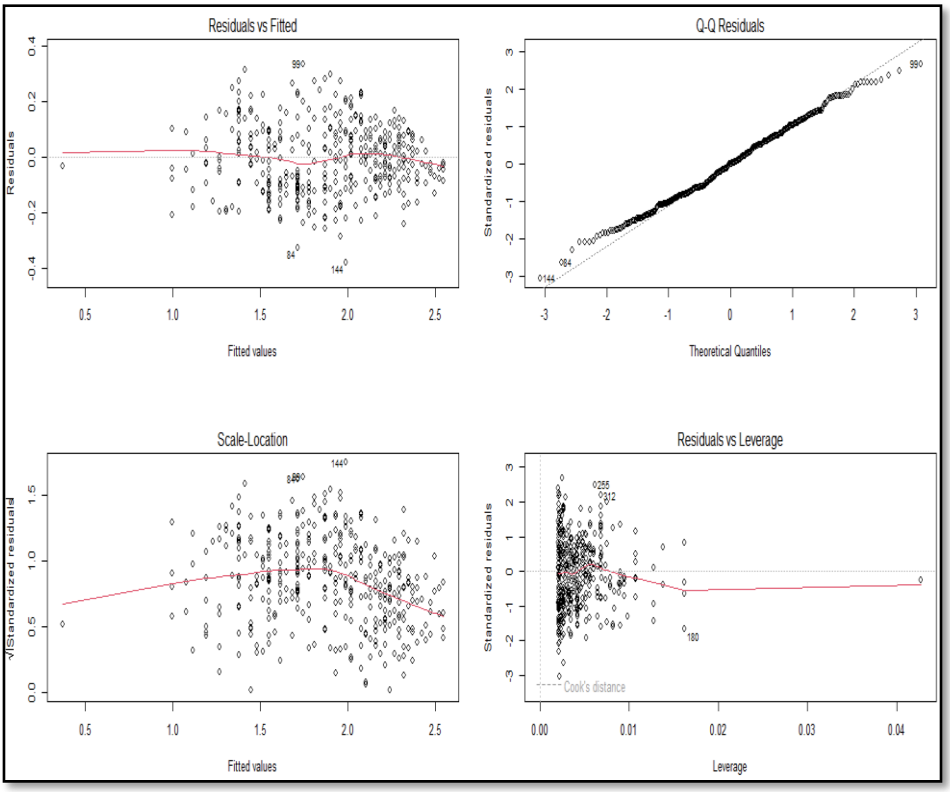
2.12.6. Model Diagnosis

Check for the important model assumptions, i.e., homoscedasticity (homogeneity of variances or residual distribution) and normal distribution of residuals through residual diagnostic plots.

Residual diagnostic plots for linear model

Use the following codes to plot residual diagnostic plots for linear model:

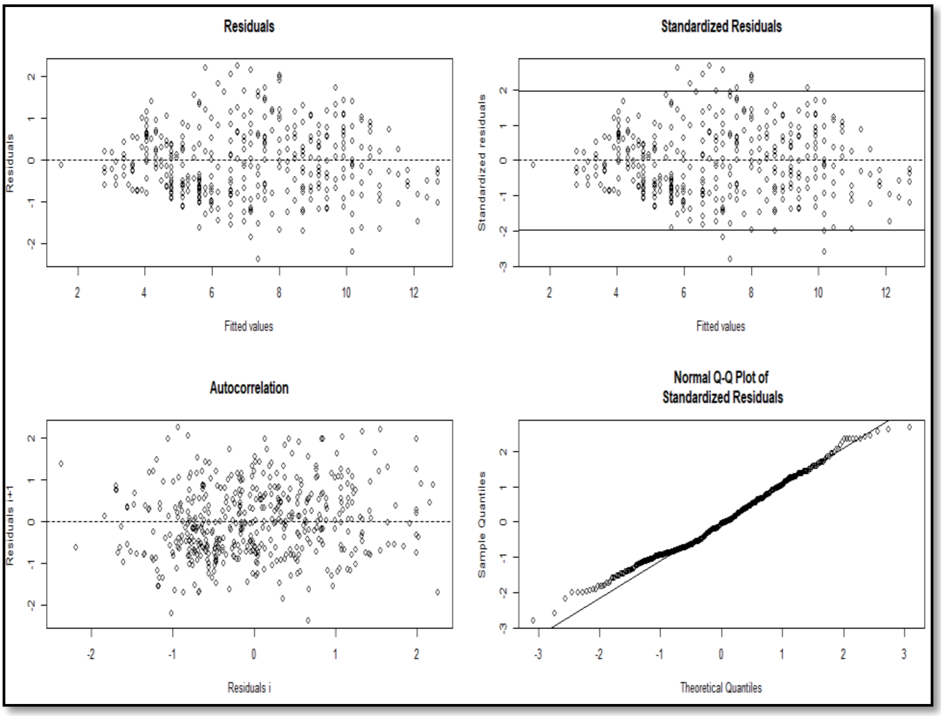
```
par(mfrow = c(2, 2))
plot(linear_model)
par(mfrow = c(1, 1))
```



Residual diagnostic plot for nonlinear model

Use the following codes to plot residual diagnostic plots for nonlinear model:

```
library(nlstools)
residuals_NLM<-nlsResiduals(nonlinear_model)
plot(residuals_NLM)
```



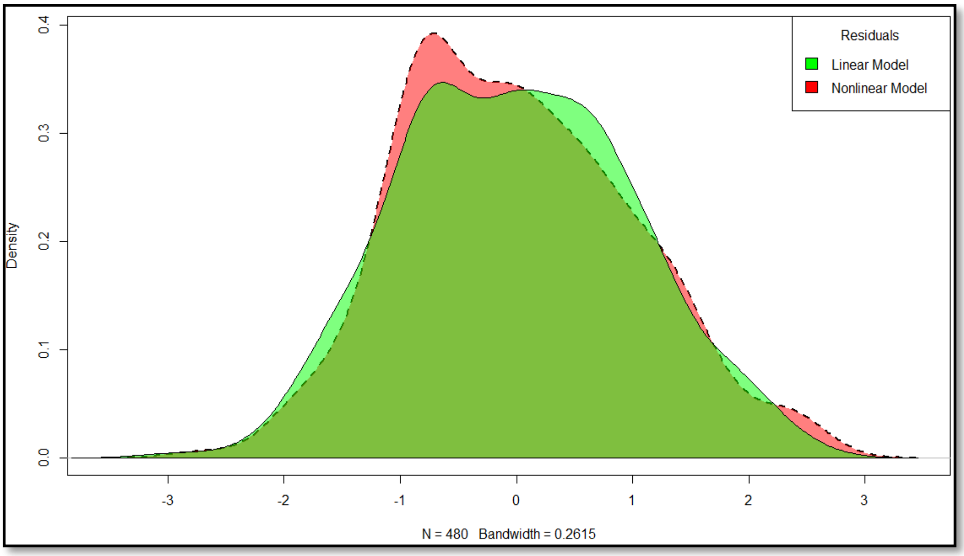
Understanding the output of residual diagnostic plot

Consider the fitted values vs. standardized residuals plot for homoscedasticity of variance and Q-Q plot for normality of variance. For homoscedasticity, the residuals should be randomly and evenly distributed both above and below the horizontal lines passing at zero. Any particular pattern like conical shape shows a gradual increase or decrease of variance (heteroscedasticity). Similarly, for the residuals to be normal, they should be closely aligned with the diagonal line in the Q-Q plot. Departure from the line shows a lack of normal distribution of the residuals.

Comparative residual density plot for normality of residuals

Use the following codes to comparative the residual density plots of the competing models (linear model vs. nonlinear model) for their normality of residuals:

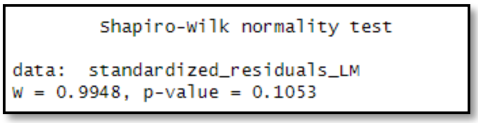
```
standardized_residuals_LM = rstandard(linear_model)
standardized_residuals_NLM = nlsResiduals(nonlinear_model)$resi2[,2]
plot(density(standardized_residuals_NLM), lty="dashed", lwd = 2, col="black")
polygon(density(standardized_residuals_NLM), border=NA, col = rgb(1, 0, 0, alpha = 0.5))
lines(density(standardized_residuals_LM), lwd = 1, col = "black")
polygon(density(standardized_residuals_LM), col = rgb(0, 1, 0, alpha = 0.5))
legend(x = "topright", title="Residuals", legend=c("Linear Model", "Nonlinear Model"),  fill
= c("green","red"))
```



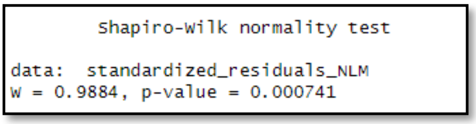
Shapiro-Wilk test of normality of residuals

shapiro.test(standardized_residuals_LM)

shapiro.test(standardized_residuals_NLM)



```
        Shapiro-Wilk normality test

data:  standardized_residuals_LM
W = 0.9948, p-value = 0.1053
```



```
        Shapiro-Wilk normality test

data:  standardized_residuals_NLM
W = 0.9884, p-value = 0.000741
```

Understanding the output of the Shapiro-Wilk test

The normality test will produce W-score and p-value for the residual. As the null hypothesis (H0) of the test is residuals are normally distributed, a p-value of higher than 0.05 (P > 0.05) shows that residuals are normally distributed. A p-value of less than 0.05 (P < 0.05) shows that the residuals are not normally distributed. In the present case for the log-transformed linear model, the p-value is 0.1053, which shows that the residuals are normally distributed, whereas the residuals from the nonlinear model are not normally distributed because of their p-value of 0.0007. Similarly, higher the W-score better is the model in terms of normality of residuals.

2.12.7. Plotting the length-weight relationship

Both the model-predicted mean values and their corresponding confidence intervals (lower and upper bounds) are essential for generating informative plots. These can be prepared using the following approaches:

Model prediction and error propagation for the linear model

The following code can be used to predict mean values along with their confidence intervals through model-based error propagation in a linear regression framework.

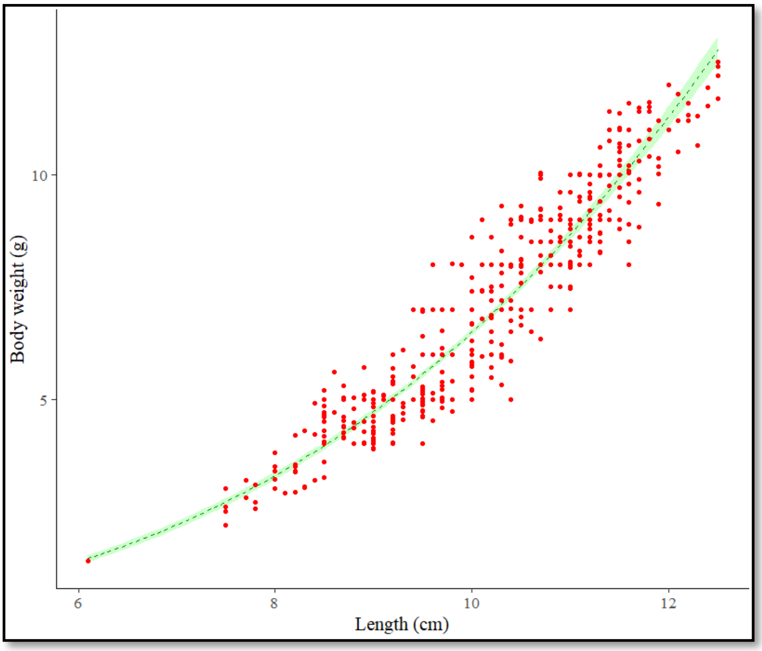
new_x<-data.frame(x=seq(min(x), max(x), (max(x)- min(x))*0.010))

```
prediction_LM<- predict(linear_model, newdata = new_x, interval="confidence")
prediction_LM<-as.data.frame(exp(prediction_LM))
FIT1<- prediction_LM$fit
UL1<- prediction_LM$lwr
LL1<- prediction_LM$upr
prediction_LM_data<-cbind(new_x, FIT1, LL1, UL1)
```

Plotting linear model output

After deriving the predictions and confidence intervals, visualize the linear model output using following code:

```
fig1<- ggplot(data=NULL, aes(x, y)) + geom_line(data=prediction_LM_data, aes(x=x,
y=FIT1), linetype = "dashed", color="darkgreen")+ geom_line(data=prediction_LM_data,
aes(x=x, y=LL1), linetype = "blank", color="darkgreen")+
geom_line(data=prediction_LM_data, aes(x=x, y=UL1), linetype = "blank",
color="darkgreen")+ theme_classic(base_size = 15)+xlab("Length (cm)") + ylab("Body
weight (g)")+ geom_ribbon(data=prediction_LM_data, aes(x = x, y=FIT1, ymin = LL1,
ymax = UL1), fill = "green", alpha=0.2)+theme(text=element_text(size=16,
family="serif"))+geom_point(data=LWR_data, aes(x=x, y=y), color="red")
fig1
```



Model prediction and error propagation for nonlinear model

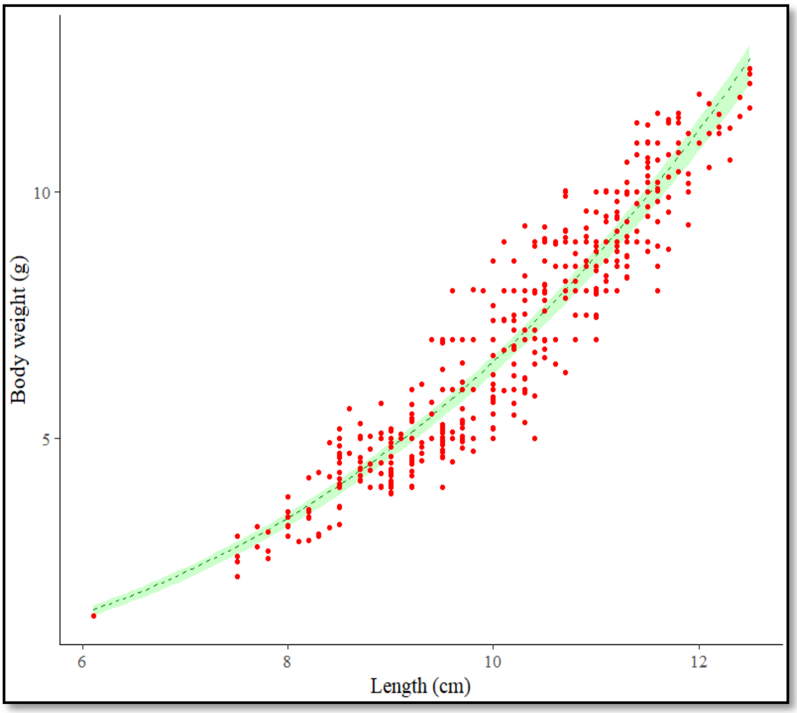
The following code can be used to predict mean values along with their confidence intervals through model-based error propagation in a nonlinear regression framework.

```
new_x<-data.frame(x=seq(min(x), max(x), (max(x)- min(x))*0.010))
library(propagate)
prediction_NLM<-predictNLS(nonlinear_model, newdata = new_x, nsim = 10000)
FIT2<- prediction_NLM$summary$Prop.Mean.1
UL2<- prediction_NLM$summary$"Prop.97.5%"
LL2<- prediction_NLM$summary$"Prop.2.5%"
prediction_NLM_data<-cbind(new_x, FIT2, LL2, UL2)
```

Plotting nonlinear model output

After deriving the predictions and confidence intervals, visualize the nonlinear model output using following code:

```
fig2<- ggplot(data=NULL, aes(x, y)) + geom_line(data=prediction_NLM_data, aes(x=x, y=FIT2), linetype = "dashed", color="darkgreen")+ geom_line(data=prediction_NLM_data, aes(x=x, y=LL2), linetype = "blank", color="darkgreen")+ geom_line(data=prediction_NLM_data, aes(x=x, y=UL2), linetype = "blank", color="darkgreen")+ theme_classic(base_size = 15)+xlab("Length (cm)") + ylab("Body weight (g)")+ geom_ribbon(data=prediction_NLM_data, aes(x = x, y=FIT2, ymin = LL2, ymax = UL2), fill = "green", alpha=0.2)+theme(text=element_text(size=16, family="serif"))+geom_point(data="LWR_data, aes(x=x, y=y), color="red")
fig2
```



2.12.8. Test for isometric growth

Once the LWR is established, it is necessary to understand if the growth of the species is isometric or allometric. If the slopes (b) of regression differs significantly from the theoretical expectation of 3 (i.e., b ≠ 3), then the growth is allometric. On the other hand, if the slopes (b) of regression is not significantly different from the theoretical expectation of 3 (i.e., b = 3), then the growth is considered isometric. Several tests are used to check if the growth is isometric or allometric. The R implementation of some of these tests is mentioned below.

t-test for isometric growth

Use the following code to perform a t-test and determine whether the estimated b value significantly differs from 3.

```
model<-lm(log(y) ~ log(x))
ttest <- function(model, coefnum, val){
 co <- coef(summary(model))
 tstat <- (co[coefnum,1]-val)/co[coefnum,2]
 2 * pt(abs(tstat), model$df.residual, lower.tail = FALSE)
}
ttest(model, 2, 3)
```

Understanding the output of t-test

The above function tests if the 2nd parameter or coefficient (i.e., b) in the regression (i.e., model) is similar to the theoretical value of 3, which is necessary to declare the growth as isometric. Only the P-value is obtained from the above function. As the null hypothesis (H0) expects b=3, a P-value higher than 0.05 (P > 0.05) shows that the growth is isometric. A P-value lower than 0.05 (P < 0.05) results in rejection of the null hypothesis and acceptance of the alternative hypothesis, which states that b≠3 (the growth is allometric.

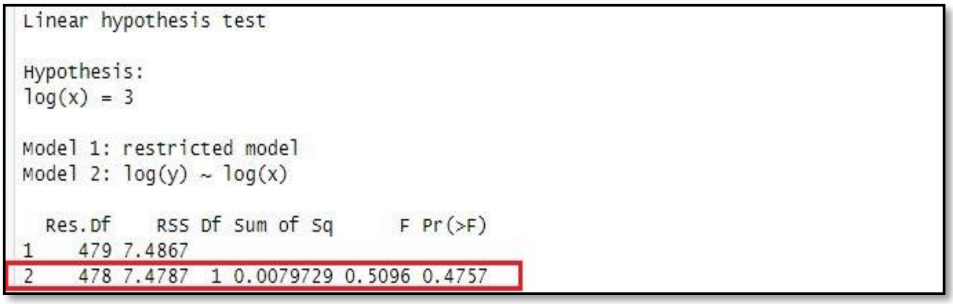
Wald test for isometric growth

Use the following code to perform a Wald test and determine whether the estimated b value significantly differs from 3.

```
library(car)
model<-lm(log(y) ~ log(x))
linearHypothesis(model, hypothesis.matrix= c(0, 1), rhs=3)
```

Understanding the output of Wald test

The argument **hypothesis.matrix** is used to define the coefficient that is tested. Use zero for the first coefficient (here, the first coefficient is intercept, i.e., a) and one for the second coefficient in question (here, the second coefficient is slope, i.e., b). Finally, rhs (i.e., right-hand side) defines the theoretical value for which the hypothesis is being tested (here, b=3). For the above condition, the null hypothesis (H_o) is b=3 and the alternative hypothesis is b≠3. As the null hypothesis (H_0) expects b=3, a P-value higher than 0.05 ($P > 0.05$) shows that the growth is isometric. A P-value lower than 0.05 ($P < 0.05$) results in rejection of the null hypothesis and acceptance of the alternative hypothesis, which states that b≠3 (the growth is allometric.



```
Linear hypothesis test

Hypothesis:
log(x) = 3

Model 1: restricted model
Model 2: log(y) ~ log(x)

  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    479 7.4867
2    478 7.4787  1 0.0079729 0.5096 0.4757
```

2.12.9. Test for difference in body weights between the sexes

Once the LWR is established, it is sometime necessary to understand if the increase in the body weight is same or different between males and females. This is done using an analysis of the covariance (ANCOVA) test. The Analysis of covariance (ANCOVA) is a statistical procedure that is followed to compare multiple regression lines. ANCOVA will test if the regression lines significantly differ in terms of slope or intercept. ANCOVA is used when there are two measurement variables (here, length and weight of fish) and one more nominal or categorical variable (here, sex of the fish) that categorize the entire dataset into two or more groups (here, males and females). The procedure for its implementation using an R interface is given below.

Analysis of Covariance (ANCOVA)

Importing length-weight data to R Interface

It is an optional step if the analysis is freshly starting as a new R session. If the analysis is continuing after establishing the LWRs from the previous steps, then there is no need to import the data again. To freshly import length-weight data, refer to '*2.12.2.Importing length-weight data to R Interface*'.

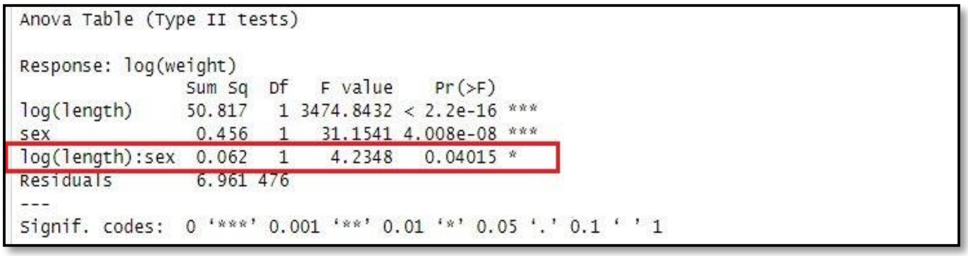
Defining the variables

Define the two measurement variables, i.e., x for the independent measurement variable (here, length of the fish, preferably in 'cm') and 'y' for the dependent measurement or response variable (here, weight of the fish, preferably in 'g'). Additionally, 'sex' of fish can be assigned as a categorical variable (as factor) for ANCOVA to observe the difference in weights between the sexes.

```
x<- as.vector(LWR_data$length)

y<- as.vector(LWR_data$weight)

sex<-as.factor(LWR_data$sex)
```

Building the model for testing the difference in slopes (b) of the regressions

```
library(car)
b_test_model<-lm(log(y)~log(x)+sex+log(x):sex)
Anova(b_test_model, type="II")
```



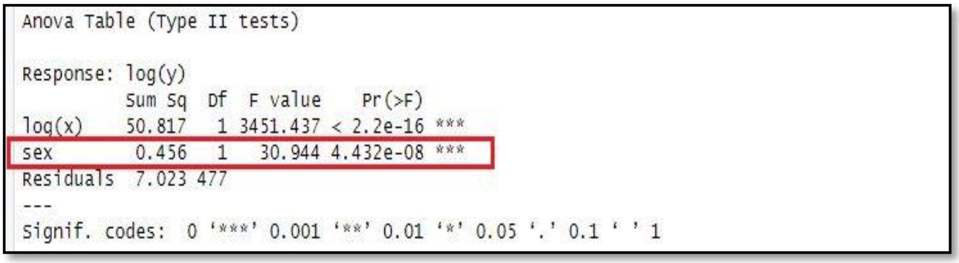
```
Anova Table (Type II tests)

Response: log(weight)
                Sum Sq  Df  F value    Pr(>F)
log(length)     50.817   1 3474.8432 < 2.2e-16 ***
sex              0.456   1   31.1541 4.008e-08 ***
log(length):sex  0.062   1    4.2348   0.04015 *
Residuals        6.961 476
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Understanding the output of the slope test

Check the interaction effect of length and sex (Here, log(x): sex) for significance. If it is significant (P<0.05), then there is a significant difference in the slope of LWRs between the categorical variable, i.e., sex (here, between males and females). In the present example, the P-value of 0.04015 clearly shows that there is a significant difference in the slope (b) of LWRs between males and females. If the interaction is not significant, then the intercept should be checked for any significant difference using the following steps.

Building the model for testing the difference in intercepts (a) of the regressions

```
library(car)
a_test_model <-lm(log(y)~log(x)+sex)
Anova(a_test_model, type="II")
```



```
Anova Table (Type II tests)

Response: log(y)
          Sum Sq  Df  F value    Pr(>F)
log(x)    50.817   1 3451.437 < 2.2e-16 ***
sex        0.456   1   30.944 4.432e-08 ***
Residuals  7.023 477
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Understanding the output of the intercept test

Check the categorical variable (here, sex) for significance. If it is significant (P<0.05), then there is a significant difference between the intercept (a) of LWRs between the categorical variable, i.e., sex (here, between males and females). In the present example, the P-value of 4.432 × 10-8 clearly shows that there is a significant difference in the intercept (a) of LWRs between males and females.

***Note:** if either of the coefficients (either a or b) is significantly different, then the LWRs are significantly different. This is an illustration to test the difference of LWRs between the sexes. The* *same principle can be followed to test the difference in LWRs between maturity stages (juveniles vs. adults) or between the species (species 1 vs. species 2) etc.*

References

Journal articles

Brewer, M.J., Butler, A., Cooksley, S.L., & Freckleton, R. (2016). The relative performance of AIC, AICc and BIC in the presence of unobserved heterogeneity. *Methods in Ecology and Evolution*, 7, 679–692. https://doi.org/10.1111/2041-210X.12541

Cawley, G.C., & Janacek, G.J. (2010). On allometric equations for predicting body mass of dinosaurs. *Journal of Zoology*, 280, 355–361. https://doi.org/10.1111/j.1469-7998.2009.00665.x

Clark, F.N. (1928). The weight-length relationship of the California sardine (*Sardina caerulea*) at San Pedro. *Fish Bulletin*, No. 12, 59 pp. https://escholarship.org/uc/item/5r71r464

Dash, G., Sen, S., Pradhan, R.K., Ghosh, S., Jose, J., & Jayasankar, J. (2023a). Modeling framework for establishing the power law between length and weight of fishes and a meta-analysis for validation of LWRs for six commercially important marine fishes from the northwestern Bay of Bengal. *Fisheries Research*, 257, 106496, 1–13. https://doi.org/10.1016/j.fishres.2022.106496

Dash, G., Sen, S., Pradhan, R.K., Ghosh, S., & Jose, J. (2023b). Application of information theory-based decision support system for high precision modeling of the length-weight relationship (LWR) for five marine shrimps from the northwestern Bay of Bengal. *Regional Studies in Marine Science*, 66, 103140, 1–10. https://doi.org/10.1016/j.rsma.2023.103140

De Giosa, M., & Czerniejewski, P. (2016). A generalized, nonlinear regression approach to the length-weight relationship of European perch (*Perca fluviatilis* L.) from the Polish coast of the southern Baltic Sea. *Archives of Polish Fisheries*, 24, 169–175. https://doi.org/10.1515/aopf-2016-0014

Froese, R. (2006). Cube law, condition factor and weight–length relationships: history, meta-analysis and recommendations. *Journal of Applied Ichthyology*, 22, 241–253. https://doi.org/10.1111/j.1439-0426.2006.00805.x

Hurvich, C.M., & Tsai, C.-L. (1995). Model selection for extended quasi-likelihood models in small samples. *Biometrics*, 51, 1077–1084. https://doi.org/10.2307/2533006

Keys, A.B. (1928). The weight-length relationship in fishes. *Proceedings of the National Academy of Sciences of the United States of America*, 14(12), 922–925. https://doi.org/10.1073/pnas.14.12.922

Sugiura, N. (1978). Further analysts of the data by Akaike's information criterion and the finite corrections. *Communications in Statistics - Theory and Methods*, 7(1), 13–26. https://doi.org/10.1080/03610927808827599

Xiao, X., White, E.P., Hooten, M.B., & Durham, S.L. (2011). On the use of log-transformation vs. nonlinear regression for analyzing biological power laws. *Ecology*, 92(10), 1887–1894. https://doi.org/10.1890/11-0538.1

Books

Burnham, K.P., & Anderson, D.R. (2002). Model selection and multimodel inference: A practical information-theoretic approach (2nd ed.). Springer-Verlag, New York, USA. https://doi.org/10.1007/b97636

Ogle, D.H. (2015). Introductory fisheries analysis with R. Chapman and Hall/CRC, Boca Raton, USA, p.317. https://doi.org/10.1201/9781315371986

Online resources

McDonald, J.H. Analysis of covariance. In: Handbook of Biological Statistics. https://www.biostathandbook.com/ancova.html

Mangiafico, S.S. Analysis of covariance. In: An R Companion for the Handbook of Biological Statistics. https://rcompanion.org/rcompanion/e_04.html

2.13. Length at Maturity

Introduction

The simple binomial (logistic) regression can be used to determine the length at maturity (LM_{50}), i.e., the length at which 50% of the population attains maturity. This can be applied with data that have only two variables, i.e., (1) independent measurement variable (here, length of the animal), and (2) dependent nominal variable with two possible outcomes (here, maturity condition, i.e., Immature or Mature). Simple logistic regression resembles linear regression, except for the fact that the dependent variable is nominal, rather than a measurement variable. For instance, when examining the relationship between length and weight (LWR), linear regression is used because the dependent variable (weight of the animal) is a measurable quantity expressed in units such as grams or kilograms, unlike the nominal or categorical dependent variables, which are expressed as either immature or mature, male or female, pass or fail, etc. The main aim of the simple binomial (logistic) regression is to predict the probability of obtaining a specific value of the nominal variable (here, mature individual) based on the measurement variable (here, length of the animal). In simple word, the goal of the exercise is to determine the probabilities of maturity for the different sizes of the animal. The aim is achieved by fitting the following equation with the data:

$$\ln\left(\frac{Y}{1-Y}\right) = a + bX$$

The above equation can be rearranged as:

$$Y = \frac{1}{1 + exp^{-(a+b*X)}} = \frac{exp^{a+b*X}}{1 + exp^{a+b*X}}$$

Where, X is the independent measurement variable (here, length of the fish), Y is the dependent categorical variable (here, probability of maturity, i.e., proportion of the matured specimens compared to the total specimen at a given length); a & b are the intercept and slope of the equation, respectively. Since the probability of maturity (Y) falls in a narrow range from 0 to 1, it creates difficulty while fitting the regression. Therefore, to overcome the situation, the odds, i.e., Y/(1-Y), which is the ratio between probability of mature/probability of immature, are used for the regression. Finally, taking the natural log of the odds makes the variable more appropriate for regression analysis. The coefficients derived from the regression are subsequently used to estimate the length at which different levels of maturity happen using the rearranged form of the above equation as follows:

$$X = \frac{\ln\left(\frac{Y}{1-Y}\right) - a}{\mathrm{b}}$$

For example to derive the X, i.e., the length (LM_{25}) at which 25% maturity happens (Y=0.25) populate the values in the above equation as mentioned below:

$$X = \frac{\ln\left(\frac{Y}{1-Y}\right) - a}{\mathrm{b}} = \mathrm{LM}_{25} = \frac{\ln\left(\frac{0.25}{1-0.25}\right) - a}{\mathrm{b}} = \frac{\ln\left(\frac{0.25}{0.75}\right) - a}{\mathrm{b}} = \frac{\ln\left(\frac{1}{3}\right) - a}{\mathrm{b}}$$

For example to derive the X, i.e., the length (LM_{50}) at which 50% maturity happens (Y=0.50) populate the values in the above equation as mentioned below:

$$X = \frac{\ln\left(\frac{Y}{1-Y}\right) - a}{\mathrm{b}} = \mathrm{LM}_{50} = \frac{\ln\left(\frac{0.50}{1-0.50}\right) - a}{\mathrm{b}} = \frac{\ln\left(\frac{0.50}{0.50}\right) - a}{\mathrm{b}} = \frac{\ln(1) - a}{\mathrm{b}}$$

For example to derive the X, i.e., the length (LM_{75}) at which 75% maturity happens (Y=0.75) populate the values in the above equation as mentioned below:

$$X = \frac{\ln\left(\frac{Y}{1-Y}\right) - a}{\mathrm{b}} = \mathrm{LM}_{75} = \frac{\ln\left(\frac{0.75}{1-0.75}\right) - a}{\mathrm{b}} = \frac{\ln\left(\frac{0.75}{0.25}\right) - a}{\mathrm{b}} = \frac{\ln(3) - a}{\mathrm{b}}$$

For example to derive the X, i.e., the length (LM_{95}) at which 95% maturity happens (Y=0.95) populate the values in the above equation as mentioned below:

$$X = \frac{\ln\left(\frac{Y}{1-Y}\right) - a}{\mathrm{b}} = \mathrm{LM}_{95} = \frac{\ln\left(\frac{0.95}{1-0.95}\right) - a}{\mathrm{b}} = \frac{\ln\left(\frac{0.95}{0.05}\right) - a}{\mathrm{b}} = \frac{\ln(19) - a}{\mathrm{b}}$$

The length and age at which different levels of maturity happen can be summarised as follows:

The length (LM_{25}) and age (tm_{25}) at which 25% of the fish in the population matures	$LM_{25} = \left[\frac{\ln\left(\frac{1}{3}\right) - a}{b}\right]$	$tm_{25} = \frac{\log\left(1 - \frac{LM_{25}}{L_\infty}\right)}{-\mathrm{K}}$
The length (LM_{50}) and age (tm_{50}) at which 50% of the fish in the population matures	$LM_{50} = \left[\frac{-a}{b}\right]$	$tm_{50} = \frac{\log\left(1 - \frac{LM_{50}}{L_\infty}\right)}{-\mathrm{K}}$

The length (LM_{75}) and age (tm_{75}) at which 75% of the fish in the population matures	$LM_{75} = \left[\frac{\ln(3) - a}{b}\right]$	$tm_{75} = \frac{\log\left(1 - \frac{LM_{75}}{L_\infty}\right)}{-K}$
The length (LM_{95}) and age (tm_{95}) at which 95% of the fish in the population matures	$LM_{95} = \left[\frac{\ln(19) - a}{b}\right]$	$tm_{95} = \frac{\log\left(1 - \frac{LM_{95}}{L_\infty}\right)}{-K}$

Length at Maturity: R Implementation

2.13.1. Requirement for Length at Maturity

Installing and loading dependent R-packages

Install the dependent R-packages dplyr for better data handling and ggplot2 for better data visualization.

```
install.packages("dplyr") #Do not install again if already installed
install.packages("ggplot2") #Do not install again if already installed
```

Load the dependent R-packages using the following codes:

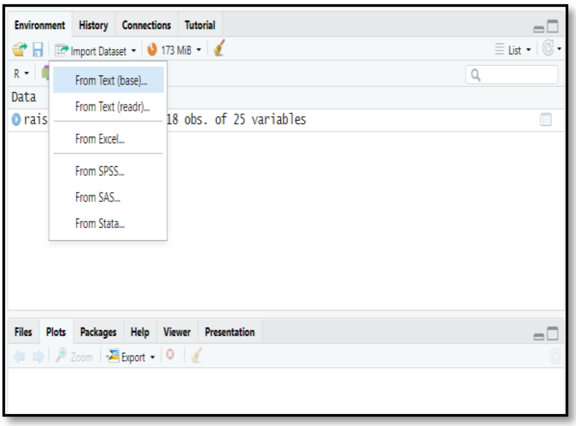
```
library(dplyr)
library(ggplot2)
```

Length-at-maturity data

The Length-at-maturity data should be available in the following format for the analysis. Refer ***'Example data file download link'*** in the last page to download and use the example data.

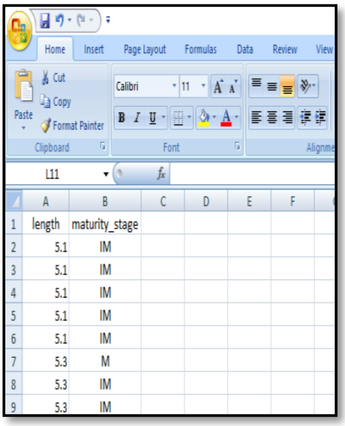
Environment> Import Dataset> from Excel and then browse the file on disk and import.

Excel Import Window



rais 18 obs. of 25 variables

Lengths and Maturity Stage (maturity_data)



	A	B	C	D	E	F
1	length	maturity_stage				
2	5.1	IM				
3	5.1	IM				
4	5.1	IM				
5	5.1	IM				
6	5.1	IM				
7	5.3	M				
8	5.3	IM				
9	5.3	IM				

***Note:** Use the lengths in ‘cm’ to maintain uniformity during subsequent calculations. IM is the ‘immature’ and M is the ‘mature’.*

2.13.2. Importing length-at-maturity data to R Interface

Click the Import Dataset of the Environment tab (top right side panel) and then select from Excel. Browse the Excel file (e.g., maturity_data) and then import.

The maturity stages information are in character form (‘IM’ and ‘M’) which need to be converted into factors (0 or 1). Use the following code to factories the maturity stages:

```
maturity_data$maturity_stage <-as.factor(maturity_data$ maturity_stage)
```

2.13.3. Modeling the relationship between length and maturity stage

Generalized linear modeling (GLM)

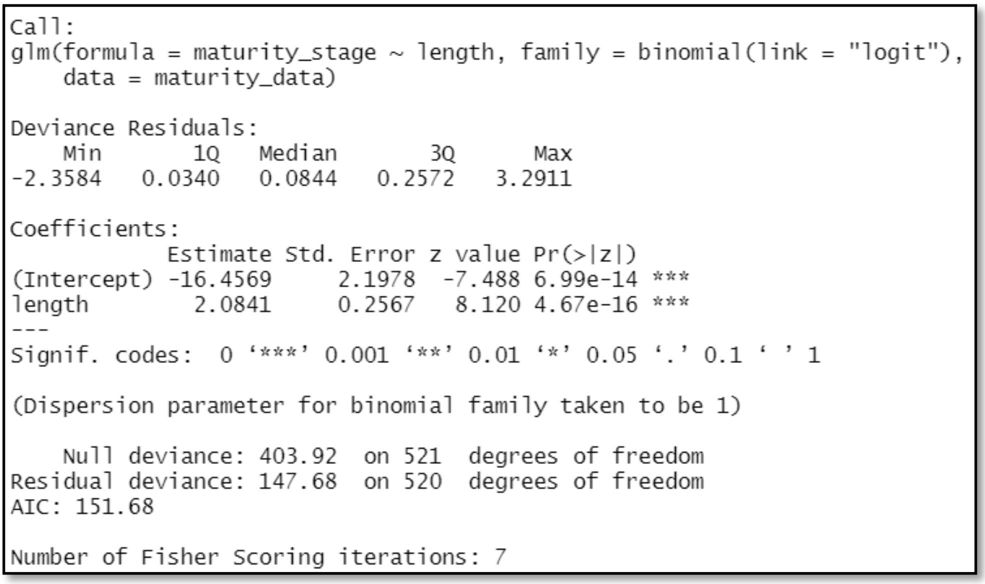
To fit a simple logistic regression using the generalized linear model (GLM), use the following code:

```
model<-glm(maturity_stage ~length, data = maturity_data, family = binomial(link = "logit"))
```

To get the detailed information on model fitting (model coefficients: intercept and slope, p-value, goodness of fit: Deviance & AIC, use the following code

```
summary(model)
```

Understanding the output of generalized linear modeling



```
Call:
glm(formula = maturity_stage ~ length, family = binomial(link = "logit"),
    data = maturity_data)

Deviance Residuals:
    Min       1Q    Median       3Q       Max
-2.3584   0.0340   0.0844   0.2572   3.2911

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -16.4569     2.1978  -7.488 6.99e-14 ***
length        2.0841     0.2567   8.120 4.67e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 403.92  on 521  degrees of freedom
Residual deviance: 147.68  on 520  degrees of freedom
AIC: 151.68

Number of Fisher Scoring iterations: 7
```

The summary(model) code produces a summary of the model. The summary code gives the model coefficients, i.e., intercept (a) and slope (b). Apart from this, the summary also produces standard error (SE), and p-value for model coefficients. Since it is a logistic regression model, it does not produce the R-squared values because of a lack of relevance. Alternatively, Akaike Information Criterion (AIC) score is generated, which can evaluate model performance for the multi-model comparison. The summary code also produces deviance and degrees of freedom of the model. The confint(model) code produces a 95% confidence interval of the model parameters (a & b).

2.13.4. Plotting the length at maturity

Model prediction and error propagation for logistic regression

Create a new X-axis variable (independent variable, i.e., lengths) for model prediction using the following code. In this example, new lengths ranging from minimum to maximum lengths in the example file (e.g., maturity_data) have been created with a gradual increment (step size) of 0.001.

```
XV<-seq(min(maturity_data$length), max(maturity_data$length), 0.001)
```

Predict the Y-axis variable (dependent variable, i.e., maturity probability) and its confidence interval (lower and upper limits) using the model parameters. In this example, type = ‘link’ has been used to produce the values of maturity (in the Y-axis) for each length (in the X-axis) as the log of odds. Alternatively, to produce the values of maturity (in Y-axis) for each length (in X-axis) as the probability of maturity, use type = ‘response’.

```
YV<-predict(model, list(length=XV), type = "link", se.fit=TRUE)
```

As the values of maturity (in Y-axis) for each length (in X-axis) has been extracted as the log of odds using type = ‘link’ function, they need to be changed to probabilities using below-mentioned plogis code. Create a new data frame (e.g., propdata) containing mean predicted value (i.e., FIT) and its confidence interval (lower, i.e., LL and upper limits, i.e., UL).

```
FIT<-plogis(YV$fit)
LL<-plogis(YV$fit-1.96*YV$se.fit)
UL<-plogis(YV$fit+1.96*YV$se.fit)
propdata<-data.frame(XV, FIT, LL, UL)
```

Deriving means and confidence intervals for length at maturity

#Use the following codes to calculate means of LM_{25}, LM_{50}, LM_{75} and LM_{95}

```
LM25<- as.numeric((log(1/3)-coef(model)[1])/coef(model)[2])
LM50<- as.numeric((log(1)-coef(model)[1])/coef(model)[2])
LM75<-as.numeric((log(3)-coef(model)[1])/coef(model)[2])
LM95<-as.numeric((log(19)-coef(model)[1])/coef(model)[2])
```

#Use the following codes to calculate confidence intervals of LM_{25}, LM_{50}, LM_{75} and LM_{95}

```
lower_LM25<-as.numeric((tail(propdata %>% filter(UL<= 0.25), n=1))[1])
upper_LM25<-as.numeric((tail(propdata %>% filter(LL<= 0.25), n=1))[1])
lower_LM50<-as.numeric((tail(propdata %>% filter(UL<= 0.5), n=1))[1])
upper_LM50<-as.numeric((tail(propdata %>% filter(LL<= 0.5), n=1))[1])
```

```
lower_LM75<-as.numeric((tail(propdata %>% filter(UL<= 0.75), n=1))[1])
upper_LM75<-as.numeric((tail(propdata %>% filter(LL<= 0.75), n=1))[1])
lower_LM95<-as.numeric((tail(propdata %>% filter(UL<= 0.95), n=1))[1])
upper_LM95<-as.numeric((tail(propdata %>% filter(LL<= 0.95), n=1))[1])
```

#Use the following codes to prepare a data frame on means and confidence intervals of lengths at maturities

```
LM_confidence <- data.frame (
  Parameter = c("LM25", "LM50", "LM75", "LM95"),
  Mean = c(LM25, LM50, LM75, LM95),
  Lower_95_CI = c(lower_LM25, lower_LM50, lower_LM75, lower_LM95),
  Upper_95_CI = c(upper_LM25, upper_LM50, upper_LM75, upper_LM95))
LM_confidence
```

Deriving means and confidence intervals of age at maturity

#Define the L_∞ and K (ex: L_∞ = 13.95 cm and K = 1.71 yr^{-1}) from the previous analysis

```
Linf<-13.95
K<-1.71
```

#Back calculate the means and confidence intervals of age at different levels of maturity from the above calculated lengths using inverse Von Bertalanfy's equation

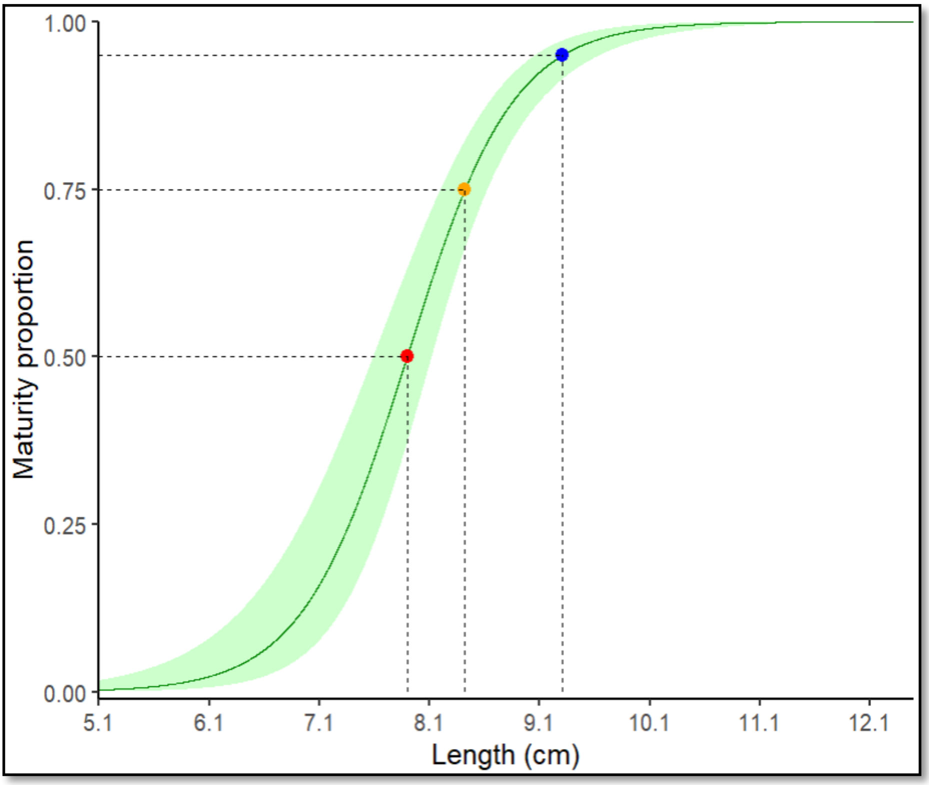
```
tLM50<-(log(1-(LM50/Linf)))/-K
lower_tLM50<-(log(1-(lower_LM50/Linf)))/-K
upper_tLM50<-(log(1-(upper_LM50/Linf)))/-K
tLM75<-(log(1-(LM75/Linf)))/-K
lower_tLM75<-(log(1-(lower_LM75/Linf)))/-K
upper_tLM75<-(log(1-(upper_LM75/Linf)))/-K
tLM95<-(log(1-(LM95/Linf)))/-K
lower_tLM95<-(log(1-(lower_LM95/Linf)))/-K
upper_tLM95<-(log(1-(upper_LM95/Linf)))/-K
```

#Use the following codes to prepare a data frame on means and confidence intervals of ages at maturities

```
tLM_confidence <- data.frame (
  Parameter = c("tLM25", "tLM50", "tLM75", "tLM95"),
  Mean = c(tLM25, tLM50, tLM75, tLM95),
  Lower_95_CI = c(lower_tLM25, lower_tLM50, lower_tLM75, lower_tLM95),
  Upper_95_CI = c(upper_tLM25, upper_tLM50, upper_tLM75, upper_tLM95))
tLM_confidence
```

Plotting the length at maturity

```
library(ggplot2)

ggplot(data=NULL, aes(x, y)) + geom_line(data=propdata, aes(x=XV, y=FIT), linetype =
"solid", color="darkgreen")+ geom_line(data=propdata, aes(x=XV, y=LL), linetype =
"blank", color="darkgreen")+ geom_line(data=propdata, aes(x=XV, y=UL), linetype =
"blank", color="darkgreen")+ geom_ribbon(data=propdata, aes(x = XV, y=FIT, ymin = LL,
ymax = UL), fill = "green", alpha=0.2)+ geom_point(aes(x=LM50, y=0.5), size=3,
colour="red")+geom_point(aes(x=LM75, y=0.75), size=3, colour="orange") +
geom_point(aes(x=LM95, y=0.95), size=3, colour="blue")+
geom_segment(aes(x=min(XV), y=0.5), xend=LM50, yend=0.5, linetype="dashed") +
geom_segment(aes(x=LM50, y=0.0), xend=LM50, yend=0.5, linetype="dashed")+
geom_segment(aes(x=min(XV), y=0.75), xend=LM75, yend=0.75, linetype="dashed") +
geom_segment(aes(x=LM75, y=0.0), xend=LM75, yend=0.75, linetype="dashed")+
geom_segment(aes(x=min(XV), y=0.95), xend=LM95, yend=0.95, linetype="dashed") +
geom_segment(aes(x=LM95, y=0.0), xend=LM95, yend=0.95, linetype="dashed")+
theme_classic(base_size = 15)+xlab("Length (cm)") + ylab("Maturity
proportion")+scale_x_continuous(expand = c(0,0), limits = c(min(XV),max(XV)), breaks =
seq(min(XV), max(XV),((max(XV)- min(XV))/10)))+ scale_y_continuous(expand =
c(0,0),limits = c(-0.01,1.0))
```



References

Online resources/e-Books

McDonald John H., Simple logistic regression, In: Handbook of Biological Statistics. https://www.biostathandbook.com/simplelogistic.html

Salvatore S. Mangiafico. Simple Logistic Regression, In: An R Companion for the Handbook of Biological Statistics. https://rcompanion.org/rcompanion/e_06.html

2.14. Length-Based Spawning Potential Ratio (LBSPR)

Introduction

Length-based spawning potential ratio (LB-SPR) has been developed for data-limited fisheries by Hordyk et al. (2016) for the estimation of SPR as a key management reference point. Spawning Potential Ratio (SPR) is a well adopted Biological Reference Point (BRP) to inform management decision. The method assumes that the length composition data represents the exploited population, which is at an equilibrium and steady state. The initial age-structured SPR model developed by Hordyk et al. (2015a and 2015b) assumes that selectivity is age-based. The model assumes that younger fish grow quickly to reach larger sizes (a "regeneration" of large-sized fish despite high mortality) and therefore, large fish are expected to exist in the population even under high fishing mortality. This assumption overestimates fishing mortality for a given size structure, as it does not fully account for the effects of size-dependent fishing. The age-structured SPR model also under-estimates the SPR level of the stock. On the contrary, the length-structured SPR (LBSPR) model incorporates size-dependent selectivity and accounts for Lee's Phenomenon (Lee, 1912), which suggests that larger individuals within an age group are disproportionately removed under size-selective fishing. The 'Lee's Phenomenon' assumes that the faster growing fish in an age group attain the length at which they are vulnerable to the fishing gear before the slower growing individuals, and thus are exposed to a higher cumulative fishing mortality throughout their lifetime. As a result, when subject to fishing mortality, the size-at-age distribution of older age classes is no longer normally distributed (truncated), as the larger individuals in each age class are reduced in number relative to the smaller individuals in the same age class. Therefore, the model segregates the population or stock to several growth-type-groups (GTGs) and tracks the cumulative impact of size-dependent selectivity and fishing mortality (Hordyk et al., 2016). The LBSPR model underestimates fishing mortality rate for a given length class compared to the age-based SPR and also overestimates SPR, especially at lower level of SPR and higher value of M/K ratio.

The model can include variable M at size, though M is assumed to be constant as a default setting. The method requires at least one year of length structured data representing the vulnerable portion of the population under investigation, and can process a time series of length frequency data as well. The method requires limited input data on life history, such as asymptotic length (L_{inf}), ratio of natural mortality and growth coefficient (M/K) and the parameters of maturity ogive (Lm_{50}& Lm_{95}). One interesting advantage with LBSPR is it does not require separate estimation of natural mortality rate (M), which is notoriously difficult to estimate for an exploited population, rather uses the ratio of M/K, which is readily available for most of the species and believed to vary less across stocks and species compared to the M (Prince et al., 2015). The analysis produce estimates on the SPR, relative fishing mortality (F/M) and selectivity parameters (SL_{50} & SL_{95}) as the model outputs of the analysis. Various steps involved in the model are given below:

Modeling the number at length

The number of individuals (number-per-recruit) at a specific length can be derived recursively using the following size distribution equation:

$$N_{L+dL} = N_L \left(\frac{L_\infty - L - dL}{L_\infty - L}\right)^{\frac{Z_L}{K}}$$

Where N_L: Number of individuals at length L

N_{L+dL}: Number of individuals at length, L + dL (where dL is a small increment in length)

Z_L = Total mortality rate at length L, calculated as M_L+F_L.

Modeling the size dependent mortality

Contrary to the conventional belief of a constant total mortality (Z) for all size groups, the approach uses size dependent total mortalities (Z_L) for the analysis. The Z is separated into two components: (1) natural mortality (M), and (2) fishing mortality (F). Fish at smaller length (age) is assumed to suffer higher natural mortality (M) which gradually decreases as fish grow bigger. The size-dependent natural mortality (M_L) is calculated as:

$$M_L = M_{L_\infty} \left(\frac{L_\infty}{L}\right)^C$$

Contrary to this, fish at smaller size faces lower fishing mortality (F), which increases with size as the fish become increasingly vulnerable to gear. As both M and F constitute total mortality (Z), it is assumed to remain constant for all sizes.

The size-dependent fishing mortality (F_L) is calculated as:

$$F_L = F \times S_L$$

Where S_L is the selectivity (also mentioned as vulnerability) of the fish at length L. This is calculated assuming a logistic selectivity as follows:

$$S_L = \frac{1}{1 + exp^{\left(-\ln(19)\left[\frac{L - S_{L50}}{S_{L95} - S_{L50}}\right]\right)}}$$

Where S_{L50} and S_{L95} are the lengths at which 50% and 95% of the fish are vulnerable to fishing gear.

Finally, the size dependent total mortalities (Z_L) is calculated as:

$$Z_L = M_L + F_L$$

Modeling the cumulative number (density) between lengths

Assuming dL is small enough so that the mortality rate (Z_L) is constant within the length class, the cumulative density (D_{L+dL}) between the length L and L + dL is calculated as:

$$D_{L+dL} = \frac{1}{Z_L}(N_L - N_{L+dL})$$

As it is a per-recruit model, the above equation is standardized to sum to one across the length classes as follows:

$$D_{L+dL} = \frac{\frac{1}{Z_L}(N_L - N_{L+dL})}{\sum_L \frac{1}{Z_L}(N_L - N_{L+dL})}$$

Instead of using specific values of Z_L, the Z_L/K ratio (depicted as θ_L) can be used in the above equation by dividing K with numerator and denominator as:

$$\widetilde{D}_{L+dL} = \frac{\frac{1}{\frac{Z_L}{K}}(N_L - N_{L+dL})}{\sum_L \frac{1}{\frac{Z_L}{K}}(N_L - N_{L+dL})} = \frac{\frac{1}{\theta_L}(N_L - N_{L+dL})}{\sum_L \frac{1}{\theta_L}(N_L - N_{L+dL})}$$

Modeling the maturity-at-size

The maturity-at-size (Mat_L) is modeled assuming logistic maturity as follows:

$$Mat_L = \frac{1}{1 + exp^{\left(-\ln(19)\left[\frac{L - Mat_{L50}}{Mat_{L95} - Mat_{L50}}\right]\right)}}$$

Where Mat_{L50} and Mat_{L95} are the lengths at which 50% and 95% of the fish are mature.

Modeling the fecundity-at-size

Assuming that egg production is proportional to the size of mature fish, fecundity-at-size is calculated as:

$$Fec_L = Mat_L L^{\beta}$$

Where β reflects the size fecundity relationship. When β is zero, it depicts the reproductive output of mature individuals is constant and independent of size, which is appropriate for some sharks and other elasmobranchs.

Modeling the spawning potential ratio (SPR)

The spawning potential between the lengths (SP_{L+dL}) is calculated as:

$$SP_{L+dL} = \frac{1}{M_L + F_L} \times \left(\widetilde{D}_L - \widetilde{D}_{L+dL}\right) \times Fec_L$$

The total spawning potential is calculated as:

$$SP = \sum_L \frac{1}{M_L + F_L} \times \left(\widetilde{D}_L - \widetilde{D}_{L+dL}\right) \times Fec_L$$

Finally, the spawning potential ratio is calculated as the proportion of reproduction in the fished state relative to the unfished state as:

$$SPR \;=\frac{\sum_L \frac{1}{M_L+F_L}\times\left(\widetilde{D}_L-\widetilde{D}_{L+dL}\right)\times Fec_L}{\sum_L \frac{1}{M_L}\times\left(\widetilde{D}_L-\widetilde{D}_{L+dL}\right)\times Fec_L}$$

Dividing both the numerator and denominator with K and, again, dividing F_L/K in the numerator with M/M will change the equation as follows:

$$SPR \;=\frac{\sum_L \frac{1}{\left[\frac{M}{K}\right]_L+\left[\frac{F}{M}\times\frac{M}{K}\right]_L}\times\left(\widetilde{D}_L-\widetilde{D}_{L+dL}\right)\times Fec_L}{\sum_L \frac{1}{\left[\frac{M}{K}\right]_L}\times\left(\widetilde{D}_L-\widetilde{D}_{L+dL}\right)\times Fec_L}$$

Modeling the in terms of Growth-Type-Group (GTG)

Contrary to the simplistic assumption of a single growth trajectory for all the individuals in the population, a more biologically oriented differential growth assumption for the different groups of individuals in a cohort is assumed in the GTG model to adequately account for the cumulative effect of size-based fishing mortality on the size structure of the stock. To account for variability in individual growth patterns, the stock or main cohort is assumed to have several sub-cohorts or growth-type-groups (g) that have a different L_∞ but a common K parameter. Therefore, all the above mentioned calculations such as Number at Length (N_{L+dL}), Size dependent mortalities (M_L, F_L and Z_L), Cumulative number (density) between lengths (D_{L+dL}), Maturity-at-size (Mat_L), Fecundity-at-size (Fec_L) and spawning potential ratio (SPR) can be calculated for the 'g' numbers of growth-type-groups (GTGs). The final SPR equation accounting for the growth variability in GTGs can be expressed as:

$$SPR \;=\frac{\sum_g\sum_L \frac{1}{\left[\frac{M}{K}\right]_{L,g}+\left[\frac{F}{M}\times\frac{M}{K}\right]_{L,g}}\times\left(\widetilde{D}_{L,g}-\widetilde{D}_{L+dL,g}\right)\times Fec_{L,g}}{\sum_g\sum_L \frac{1}{\left[\frac{M}{K}\right]_{L,g}}\times\left(\widetilde{D}_{L,g}-\widetilde{D}_{L+dL,g}\right)\times Fec_{L,g}}$$

The above formulations (equations) illustrates that the SPR, F/M, S_{L50}, and S_{L95} can be derived using the reasonable estimates of M/K ratio, L_∞, and $\sigma^2_{L\infty}$ (or $CV_{L\infty}$), size-at-maturity and a representative sample of length structure of the catch. The following multinomial negative log likelihood function (NLL) is used to fit the model:

$$NLL = \underset{\frac{F}{M}S_{L50},\,S_{L95}}{\operatorname{argmin}} \sum_i O_i \, ln\frac{\widetilde{P}_i}{\widetilde{O}_i}$$

Where O_i and $\widetilde{O}_i$ are the observed number and proportion in length class i respectively, and $\widetilde{P}_i$ is the model estimate of the probability in length class i. $\widetilde{P}_i$ can be calculated by summing all the number of individuals (D_{L+dL}) in each length class of the 'g' growth-type-groups and then multiplying by the estimated selectivity, and standardized to sum to one. The spawning potential ratio is calculated from the above mentioned SPR equation using the model estimates of F/M and selectivity-at-length parameters, and the input parameters M/K, L_∞, and $\sigma^2_{L\infty}$ (or $CV_{L\infty}$), and size-at-maturity parameters.

LBSPR: R-implementation

2.14.1. Requirement for LBSPR

Installing and loading LBSPR R-package

The LBSPR package can be installed from CRAN and loaded for running analysis using the following code:

```
install.packages("LBSPR") # for installation of the package
library(LBSPR) # for loading the package for analysis
```

Alternatively, the development version of the package can be downloaded from GitHub using the devtools package:

```
install.packages("devtools")
devtools::install_github("AdrianHordyk/LBSPR")
```

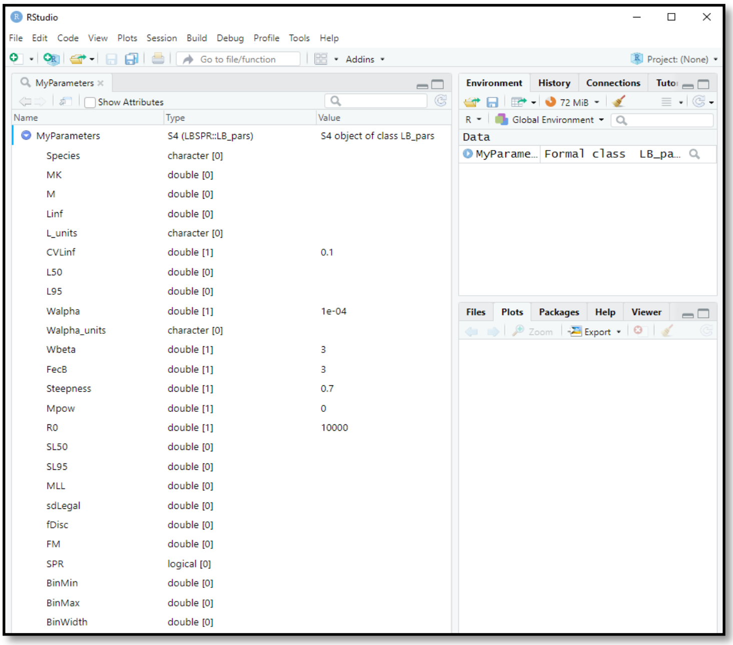
Supplying LFQ data and essential input parameters

The essential parameters for the species such as species name, asymptotic length (L_{inf}), MK ratio (M/K), parameters of maturity ogive (Lm_{50} & Lm_{95}) and the unit of length (cm) can be supplied by creating an S4 object "**MyParameters**".

Initially, a blank "**MyParameters**" is created, which contains all the parameter fields required for running LB-SPR analysis.

```
MyParameters <- new("LB_pars")#create a blank S4 object of class LB_pars for input parameters
```

Freshly created MyParameters (S4 object)



Name	Type	Value
MyParameters	S4 (LBSPR::LB_pars)	S4 object of class LB_pars
Species	character [0]	
MK	double [0]	
M	double [0]	
Linf	double [0]	
L_units	character [0]	
CVLinf	double [1]	0.1
L50	double [0]	
L95	double [0]	
Walpha	double [1]	1e-04
Walpha_units	character [0]	
Wbeta	double [1]	3
FecB	double [1]	3
Steepness	double [1]	0.7
Mpow	double [1]	0
R0	double [1]	10000
SL50	double [0]	
SL95	double [0]	
MLL	double [0]	
sdLegal	double [0]	
fDisc	double [0]	
FM	double [0]	
SPR	logical [0]	
BinMin	double [0]	
BinMax	double [0]	
BinWidth	double [0]	

```
Data
MyParame…  Formal class  LB_pa…
```

Subsequently, the following series of codes can be used to populate the different parameter fields in the S4 object "**MyParameters**".

```
MyParameters@Species <- "P stylifera"# species name
MyParameters@Linf <- 175 # Asymptotic length input
MyParameters@L50 <- 89.6 # length at which 50% individual attain maturity
MyParameters@L95 <- 137 # length at which 95% individual attain maturity
MyParameters@MK <- 1.64 # ratio of natural mortality and growth coefficient (M/K)
MyParameters@L_units <- "mm" # units of input data (LF, Linf, Lm50 and Lm95)
```

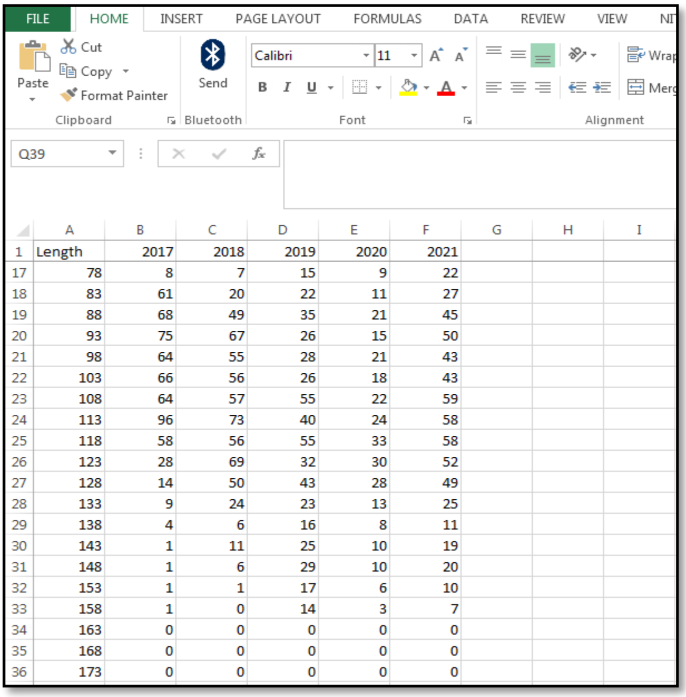
> ***Note:*** *This example illustrates the supply of essentially required input parameters for the analysis. The user can supply any additional parameters depending on the availability.*

Importing annual length frequency data (LFQ)

Annual length frequency data (LFQ)

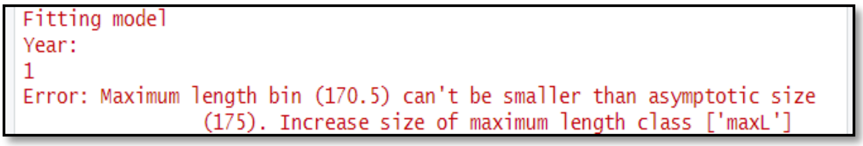
The data sheet is to be preferably prepared in CSV file format (**e.g., Pstylifera.csv**) and has the first column "Length", which refers to the mid-value of the class interval (mid length). The subsequent columns are the frequency (observed number) against each class interval and the headers of the column reflect the year of observation (e.g. 2017, 2018, … , 2021). Refer ***'Example data file download link'*** in the last page to download and use the example data.

Pstylifera.csv



	A	B	C	D	E	F
1	Length	2017	2018	2019	2020	2021
17	78	8	7	15	9	22
18	83	61	20	22	11	27
19	88	68	49	35	21	45
20	93	75	67	26	15	50
21	98	64	55	28	21	43
22	103	66	56	26	18	43
23	108	64	57	55	22	59
24	113	96	73	40	24	58
25	118	58	56	55	33	58
26	123	28	69	32	30	52
27	128	14	50	43	28	49
28	133	9	24	23	13	25
29	138	4	6	16	8	11
30	143	1	11	25	10	19
31	148	1	6	29	10	20
32	153	1	1	17	6	10
33	158	1	0	14	3	7
34	163	0	0	0	0	0
35	168	0	0	0	0	0
36	173	0	0	0	0	0

Note: *It must be noted that the maximum length in the LFQ must be larger than the input value of the asymptotic length (L_{inf}). In this example, as the L_{inf} used for the species is 175 mm, there has to be a bare minimum size group with mid-length of 173 mm, so that the maximum length for the mid-length 173 mm with 5 mm class interval (170.5-175.5 mm) will be 175.5 mm, higher than the provided L_{inf} of 175 mm. Use of size group with a maximum length of less than 175 mm will produce the following error during model fitting.*



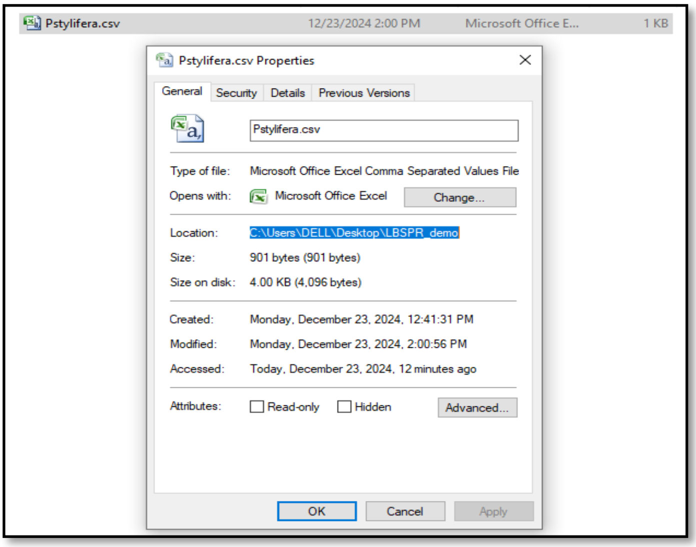
```
Fitting model
Year:
1
Error: Maximum length bin (170.5) can't be smaller than asymptotic size
              (175). Increase size of maximum length class ['maxL']
```

Solution: Check the mid-length and class interval. If the sum of mid-length + half of the class interval is less than the supplied L_{inf}, create the next new mid-length, even if no observation are made under this length class.

The externally created **Pstylifera.csv** can be imported to R-interface and saved along with the previously supplied parameters by creating a new file “**MyLengths**” using the following code:

```
MyLengths<-new("LB_lengths", LB_pars=MyParameters,
file=paste0("C:/Users/DELL/Desktop/LBSPR_demo/Pstylifera.csv"),
dataType="freq", header=TRUE)
```

To determine the path of the CSV file (e.g., Pstylifera.csv) right-click on it, then select ‘**Properties**’ by left clicking on it, and copy the file location under the ‘**General**’ tab (e.g., C:\Users\cmfri\Desktop\LBSPR_demo). Replace the bold portion of the code with this copied file location, and then add the file name with its extension (e.g., /**Pstylifera.csv**). Make sure to use double backslashes (\\) or a single forward slash (/) between each segment of the path and enclose the entire path in quotation marks like (“.../..../..../....”) or (“...\\....\\....\\....”).



Pstylifera.csv 12/23/2024 2:00 PM Microsoft Office E... 1 KB

Pstylifera.csv Properties

Pstylifera.csv

Type of file: Microsoft Office Excel Comma Separated Values File

Opens with: Microsoft Office Excel

Location: C:\Users\DELL\Desktop\LBSPR_demo

Size: 901 bytes (901 bytes)

Size on disk: 4.00 KB (4,096 bytes)

Created: Monday, December 23, 2024, 12:41:31 PM

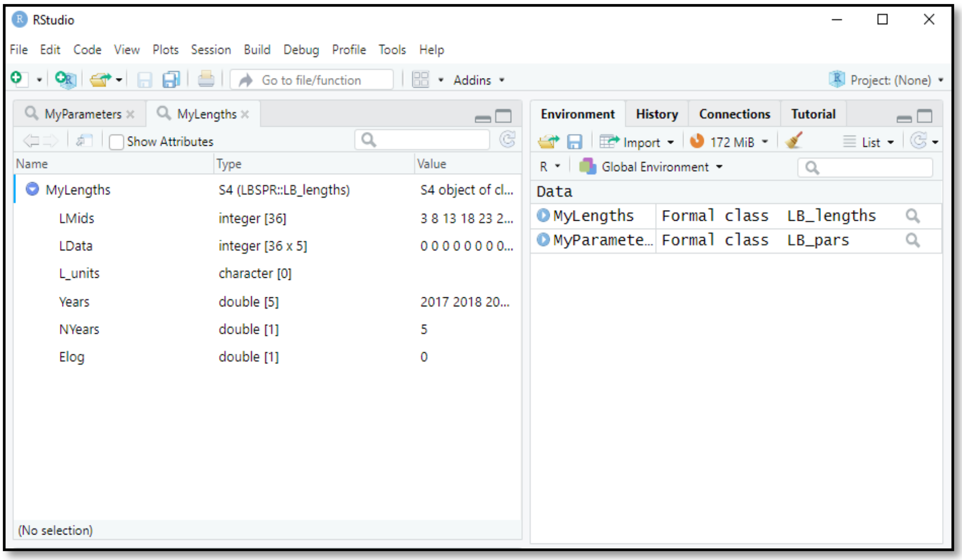
Modified: Monday, December 23, 2024, 2:00:56 PM

Accessed: Today, December 23, 2024, 12 minutes ago

Attributes: □ Read-only □ Hidden

The user can get more insight into the imported data using the code:

View(MyLengths) # to see the data in R studio



Name	Type	Value
MyLengths	S4 (LBSPR::LB_lengths)	S4 object of cl...
LMids	integer [36]	3 8 13 18 23 2...
LData	integer [36 x 5]	0 0 0 0 0 0 0 0...
L_units	character [0]	
Years	double [5]	2017 2018 20...
NYears	double [1]	5
Elog	double [1]	0

```
Data
MyLengths    Formal class  LB_lengths
MyParamete…  Formal class  LB_pars
```

The data structure (**e.g., MyLengths**) shows that there are 36 length class (LMids, all rows need not be populated) and the number of years of data (NYears) is 5 and the years are 2017 to 2021.

2.14.2. Running the LBSPR analysis

The "LBSPRfit" function is used for fitting the LBSPR model to the imported LF data (**e.g. MyLengths**) using supplied parameters (**e.g., MyParameters**). The code for running of the model is:

LBSPR_model<- LBSPRfit(MyParameters, MyLengths, **yrs = NA**) #to fit the model for all the available years

LBSPR_model<- LBSPRfit(MyParameters, MyLengths, **yrs = 5**) #to fit the model for any specific year (**e.g., fifth year** by specifying **yrs = 5**)

LBSPR text output

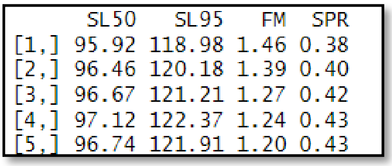
The model estimates can be visualized in the R console using the following code:

LBSPR_model@Ests #viewing the model output estimates in R console

The outputs are selectivity parameters (SL50 & SL95), FM ratio (F/M) and SPR for the five years. However, when multiple years of data are used, the estimates are presented as smoothed figures following the Kalman filter and the Rauch-Tung-Striebel smoother function. The raw estimates of each year can be visualized in the R console using the following code:

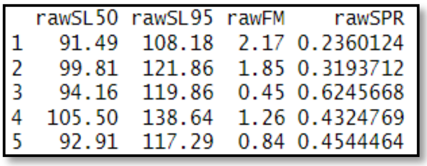
data.frame(rawSL50=LBSPR_model@SL50, rawSL95= LBSPR_model@SL95, rawFM= LBSPR_model@FM, rawSPR= LBSPR_model@SPR) # to see raw estimates

Smoothed Estimates



```
      SL50   SL95   FM  SPR
[1,] 95.92 118.98 1.46 0.38
[2,] 96.46 120.18 1.39 0.40
[3,] 96.67 121.21 1.27 0.42
[4,] 97.12 122.37 1.24 0.43
[5,] 96.74 121.91 1.20 0.43
```

Raw Estimates

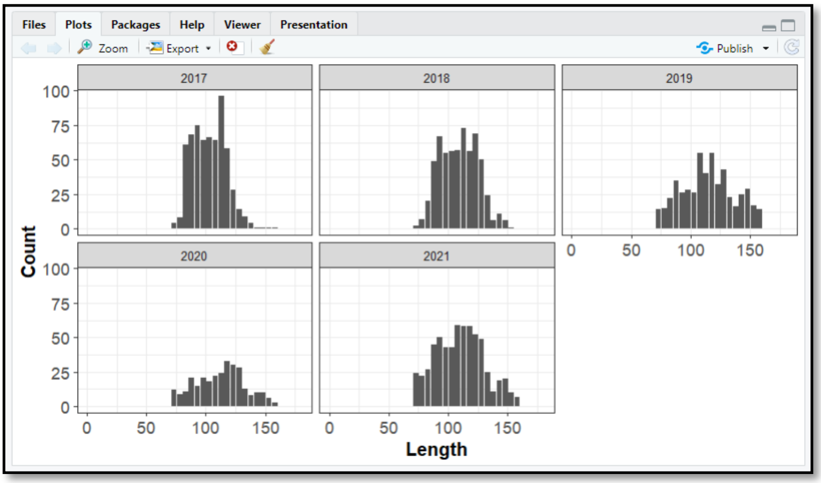


```
  rawSL50 rawSL95 rawFM    rawSPR
1   91.49  108.18  2.17 0.2360124
2   99.81  121.86  1.85 0.3193712
3   94.16  119.86  0.45 0.6245668
4  105.50  138.64  1.26 0.4324769
5   92.91  117.29  0.84 0.4544464
```

2.14.3. Plotting LBSPR analysis outputs

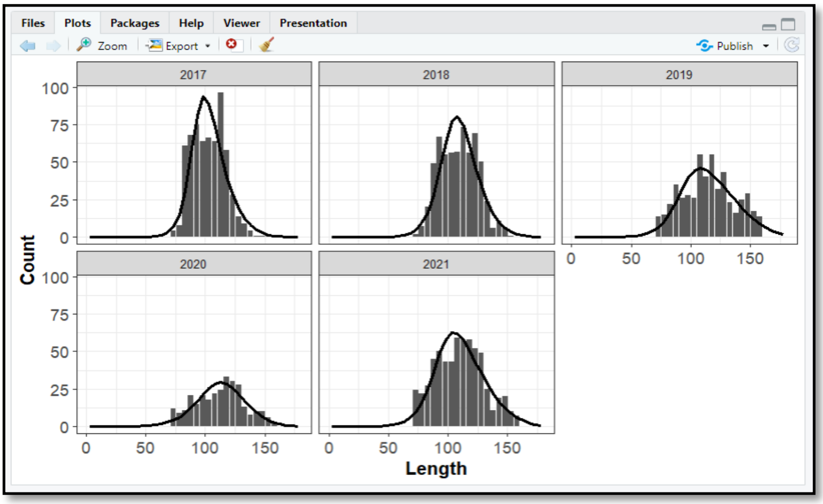
User can plot the imported length frequency data for visualization using "plotSize" function and the following code:

plotSize(MyLengths) # LF data visualization; **MyLengths** refers to the imported data file.



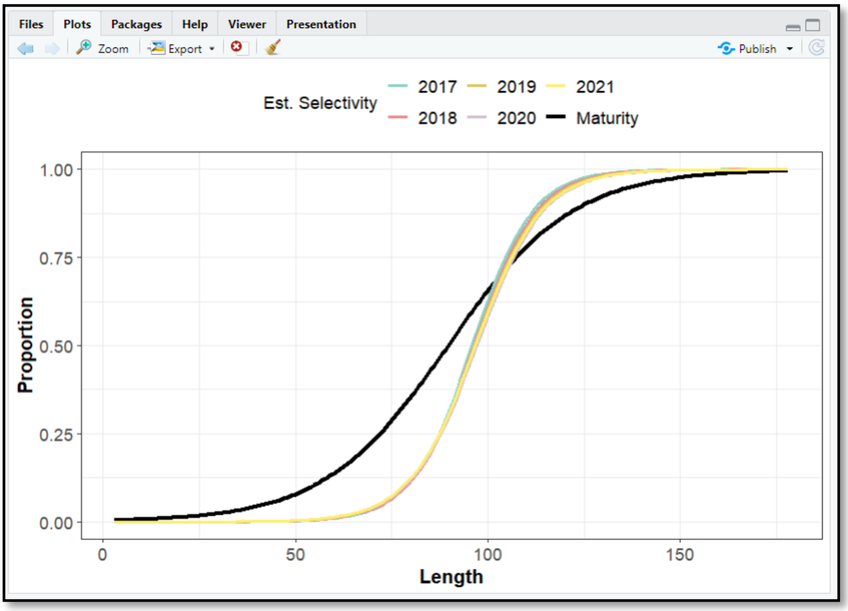
The fitted model to the input LF data can be visualized using the following code:

plotSize(LBSPR_model) # plot the model fit to the input LF data



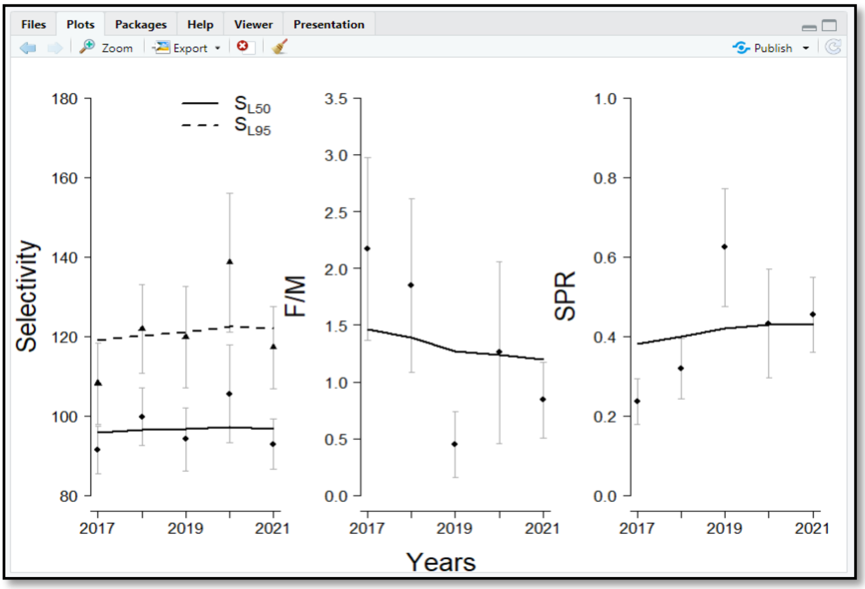
The graphical outputs, like maturity curve and selectivity curves for every available year, can also be plotted using the given codes:

plotMat(LBSPR_model)# plots year-wise selectivity curve along with maturity ogive



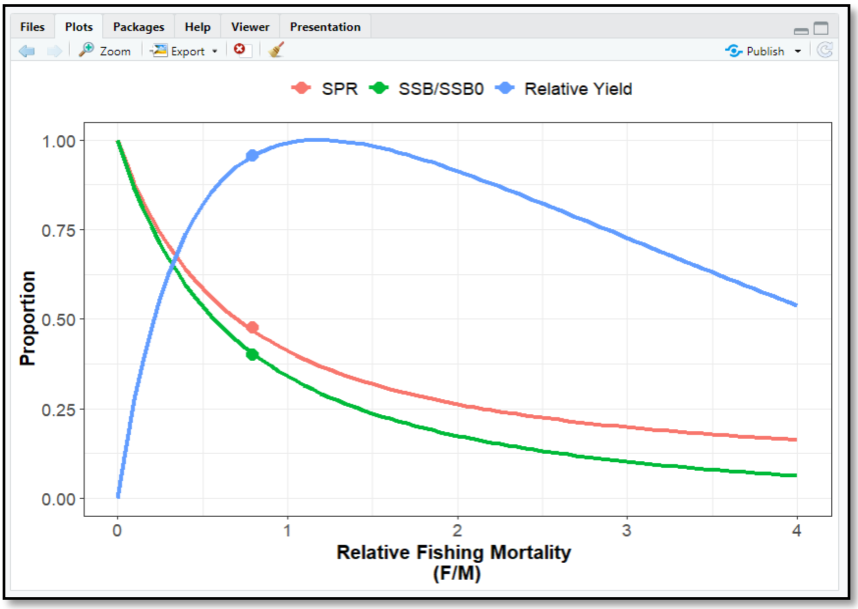
The code given below plots all the output of the model (SL50, SL95, F/M & SPR) in a single graph

plotEsts(LBSPR_model)# plotting all the graphs in a single graph



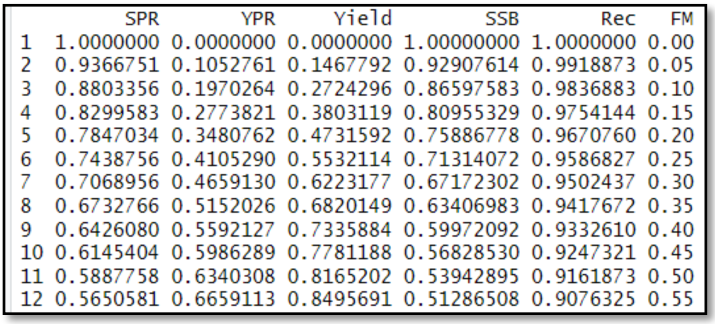
The code given below plots the SPR, SSB and relative yield trajectories as a response to change in relative F (F/M) for different years. For example, to generate the SPR, SSB and relative yield trajectories for the **fifth year**, the user can fit the model for fifth-year data only by specifying **yrs = 5** in the **LBSPRfit code** and then plotting the trajectories using **plotCurves** code as mentioned bellow:

```
LBSPR_model<- LBSPRfit(MyParameters, MyLengths, yrs = 5)
plotCurves(LBSPR_model)
```



The values used for the above plotting of the SPR, SSB and relative yield trajectories as a response to change in relative F (F/M) can be retrieved using the following code:

```
calcCurves(LBSPR_model)
```

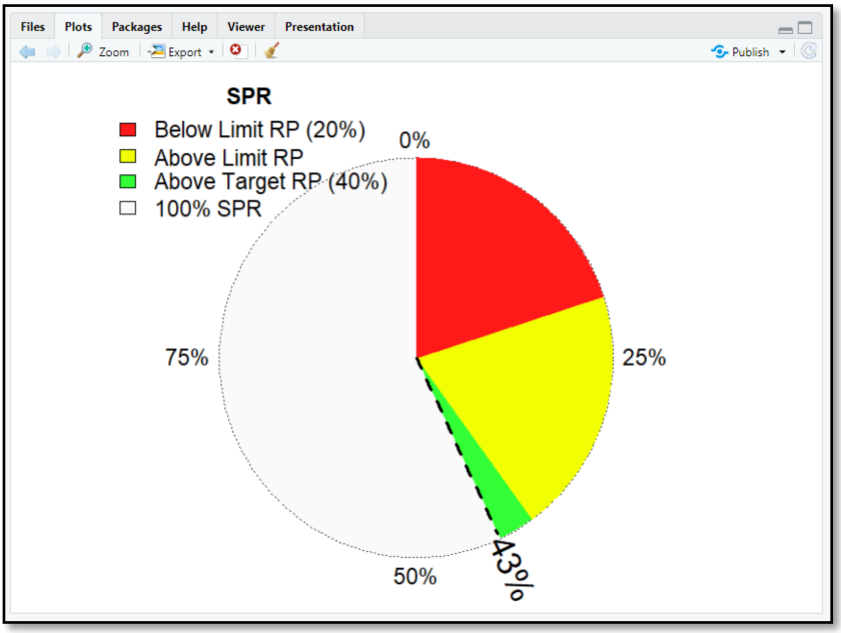


```
         SPR       YPR     Yield        SSB       Rec   FM
1  1.0000000 0.0000000 0.0000000 1.00000000 1.0000000 0.00
2  0.9366751 0.1052761 0.1467792 0.92907614 0.9918873 0.05
3  0.8803356 0.1970264 0.2724296 0.86597583 0.9836883 0.10
4  0.8299583 0.2773821 0.3803119 0.80955329 0.9754144 0.15
5  0.7847034 0.3480762 0.4731592 0.75886778 0.9670760 0.20
6  0.7438756 0.4105290 0.5532114 0.71314072 0.9586827 0.25
7  0.7068956 0.4659130 0.6223177 0.67172302 0.9502437 0.30
8  0.6732766 0.5152026 0.6820149 0.63406983 0.9417672 0.35
9  0.6426080 0.5592127 0.7335884 0.59972092 0.9332610 0.40
10 0.6145404 0.5986289 0.7781188 0.56828530 0.9247321 0.45
11 0.5887758 0.6340308 0.8165202 0.53942895 0.9161873 0.50
12 0.5650581 0.6659113 0.8495691 0.51286508 0.9076325 0.55
```

The code given below plots the SPR status of the required year with reference to the prescribed SPR limit and SPR target. For example, to generate the SPR status for the **fifth year**, the user can fit the model for fifth-year data only by specifying **yrs = 5** in the

LBSPRfit code and then plotting the trajectories using **plotSPRCirc** code as mentioned below:

```
LBSPR_model<- LBSPRfit(MyParameters, MyLengths, yrs = 5)
plotSPRCirc(LBSPR_model)
```



2.14.4. Plotting observed LFQ vs. target LFQ

The model also allows us to graphically compare the input LF data with the ideal length structure (Target LF) corresponding to the given spawning potential ratio. To do so, we have to rerun the above analysis with some modification.

Delete the previous analysis and outputs (not required) for a smooth run and avoid confusion using the following code:

```
rm(list=ls()) # erases all the existing analysis and data
```

Now create a new blank "**MyParameters**" S4 object as done previously and populate it with the same input values for the parameters. An additional field "MyPars@SPR"needs to be populated, corresponding to the targeted SPR depending on the resilience of the species (Ideally taken as 40%).

```
MyParameters <- new("LB_pars")#create a blank S4 object of class LB_pars for input parameters
MyParameters@Species <- "P stylifera"# species name
MyParameters@Linf <- 175 # asymptotic length input
MyParameters@L50 <- 89.6 # length at which 50% individual attain maturity
MyParameters@L95 <- 137 # length at which 95% individual attain maturity
MyParameters@MK <- 1.64 # ratio of natural mortality and growth coefficient (M/K)
```

MyParameters@SPR <- 0.40 # Target SPR for the reference LF

MyParameters@L_units <- "mm" # units of input data (LF, L_{inf}, Lm_{50} and Lm_{95})

The observed length frequency data needs to be imported in the R environment, as done previously

MyLengths<-new("LB_lengths", LB_pars=MyParameters,
file=paste0("**C:/Users/DELL/Desktop/LBSPR_demo**/Pstylifera.csv"),
dataType="freq", header=TRUE)

The LBSPR model will be fitted to the LF data using LBSPRfit

LBSPR_model <- LBSPRfit(MyParameters, MyLengths, yr = NA)# runs the LBSPR model

For simulating LF data with desired SPR, we need additional input related to selectivity parameters, which can be provided from the estimates of fitted model. As the data is of multi-year type, we need to specify the year for which the comparison is to be made. Following codes can be used:

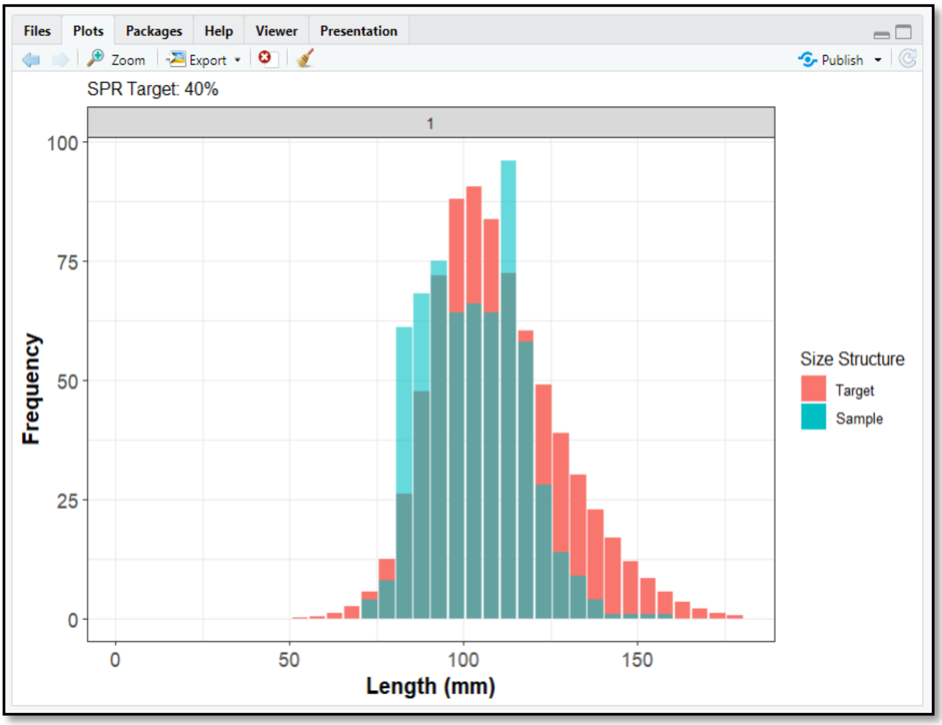
yr <- 5 # selection of the year, **here 5th year was selected as an example**

MyParameters@SL50 <- LBSPR_model@SL50[yr]# SL50 value from model output

MyParameters@SL95 <- LBSPR_model@SL95[yr]# SL95 value from model output

The observed LF and the target LF (SPR = 0.40 in the present case) can be graphically seen using plotTarg function

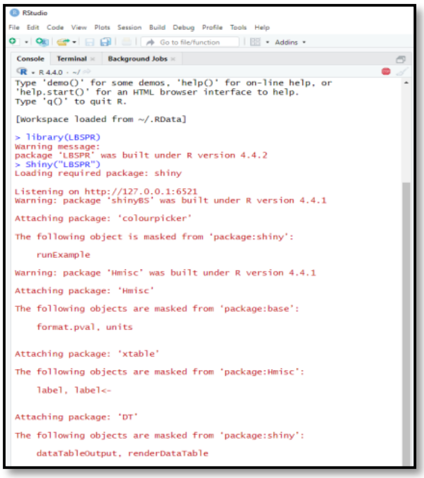
plotTarg(MyParameters, MyLengths, yr=yr)



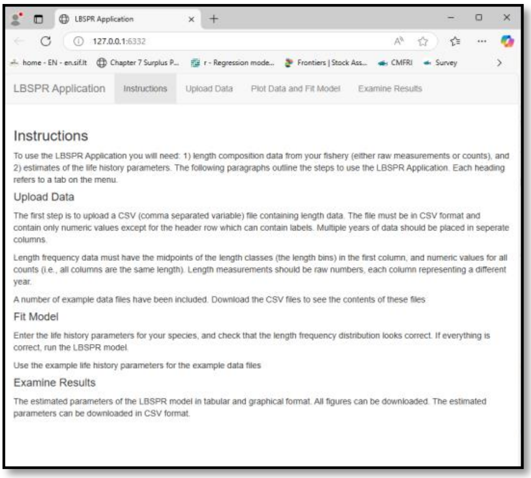
2.14.5. Implementing R Shiny package for LBSPR analysis

The Shiny package enables the creation of user-friendly applications that combine R's computational power with the interactivity of modern web browsers. This approach simplifies access to R-based models through intuitive interfaces, eliminating the need for direct interaction with R software. The Shiny application for LBSPR can be launched using the following Shiny function code in the R-Studio interface, which will open the interface in the web browser.

```
library(LBSPR)
Shiny("LBSPR")
```



```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> library(LBSPR)
Warning message:
package 'LBSPR' was built under R version 4.4.2
> Shiny("LBSPR")
Loading required package: shiny

Listening on http://127.0.0.1:6521
Warning: package 'shinyBS' was built under R version 4.4.1

Attaching package: 'colourpicker'

The following object is masked from 'package:shiny':

    runExample

Warning: package 'Hmisc' was built under R version 4.4.1

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units


Attaching package: 'xtable'

The following objects are masked from 'package:Hmisc':

    label, label<-


Attaching package: 'DT'

The following objects are masked from 'package:shiny':

    dataTableOutput, renderDataTable
```



Instructions

To use the LBSPR Application you will need: 1) length composition data from your fishery (either raw measurements or counts), and 2) estimates of the life history parameters. The following paragraphs outline the steps to use the LBSPR Application. Each heading refers to a tab on the menu.

Upload Data

The first step is to upload a CSV (comma separated variable) file containing length data. The file must be in CSV format and contain only numeric values except for the header row which can contain labels. Multiple years of data should be placed in seperate columns.

Length frequency data must have the midpoints of the length classes (the length bins) in the first column, and numeric values for all counts (i.e., all columns are the same length). Length measurements should be raw numbers, each column representing a different year.

A number of example data files have been included. Download the CSV files to see the contents of these files

Fit Model

Enter the life history parameters for your species, and check that the length frequency distribution looks correct. If everything is correct, run the LBSPR model.

Use the example life history parameters for the example data files

Examine Results

The estimated parameters of the LBSPR model in tabular and graphical format. All figures can be downloaded. The estimated parameters can be downloaded in CSV format.

References

Journal articles

Hordyk, A. R., Ono, K., Sainsbury, K. J., Loneragan, N., & Prince, J. D. (2015a). Some explorations of the life history ratios to describe length composition, spawning-per-recruit, and the spawning potential ratio. *ICES Journal of Marine Science, 72*(1), 204–216. https://doi.org/10.1093/icesjms/fst235

Hordyk, A. R., Ono, K., Valencia, S. R., Loneragan, N. R., & Prince, J. D. (2015b). A novel length-based empirical estimation method of spawning potential ratio (SPR), and tests of its performance, for small-scale, data-poor fisheries. *ICES Journal of Marine Science, 72*(1), 217–231. https://doi.org/10.1093/icesjms/fsu004

Hordyk, A. R., Ono, K., Prince, J. D., & Walters, C. J. (2016). A simple length-structured model based on life history ratios and incorporating size-dependent selectivity: Application to spawning potential ratios for data-poor stocks. *Canadian Journal of Fisheries and Aquatic Sciences, 73*(1), 1–13. https://doi.org/10.1139/cjfas-2015-0422

Prince, J. D., Hordyk, A. R., Valencia, S. R., Loneragan, N. R., & Sainsbury, K. J. (2015).

Revisiting the concept of Beverton–Holt life-history invariants with the aim of informing data-poor fisheries assessment. *ICES Journal of Marine Science, 72*(1), 194–203. https://doi.org/10.1093/icesjms/fsu011

Lee, R. M. (1912). An investigation into the methods of growth determination in fishes by means of scales. *Journal du Conseil International pour l'Exploration de la Mer, s1*(63), 3–34. https://doi.org/10.1093/icesjms/s1.63.3

Online resources

Hordyk, A. R. (n.d.). *LBSPR: Length-Based Spawning Potential Ratio* [Article]. https://adrianhordyk.github.io/LBSPR/articles/LBSPR.html

The Barefoot Ecologist. (n.d.). *LBSPR: Length-Based Spawning Potential Ratio*. http://barefootecologist.com.au/lbspr

2.15. The Length-based Bayesian (LBB) estimation method

Introduction

The Length-based Bayesian (LBB) estimation method has been developed for data-limited fisheries by Froese et al. (2018) for the estimation of key management reference points such as depletion level (B/B_0), relative fishing mortality (F/M), length at first capture corresponding to maximum catch and biomass (L_{c_opt}) and the relative biomass producing maximum sustainable yields (B_{MSY}/B_0). Being a data limited method, it requires only annual length frequency (LFQ) data as mandatory data input for analysis. The method expects that the sampled length frequency (LFQ) data truly represent the exploited phase of the population. The method assumes that the organism grows throughout the life span and growth follows the Von Bertalanffy's (VBGF) growth pattern (von Bertalanffy, 1938). Because of which, the increase in length can be used as a proxy for elapsed time. The analysis also assumes a synthetic cohort under equilibrium condition with stable growth, mortality, and recruitment during the study period. The analysis expects a ratio of natural mortality (M) to somatic growth (K) with a prior mean value around 1.5 which is a typical value for adults of species that grow throughout their life, reaching maximum size at maximum age (Hordyk et al., 2015 and Froese et al., 2016).

The analysis requires prior information on crucial input parameters like asymptotic length (L_∞), length at first capture (L_C), but relative values of natural mortality (M/K) and fishing mortality (F/M) which minimizes the parameter requirements. Nevertheless, the default values are either pre-fixed or estimated from the input LFQ and hence the user need not provide the information for the same (Froese et al., 2018). However, if robust or local estimates of these priors are available, the user can use the same replacing the default priors. These inputs were used to estimates of exploited biomass relative to unexploited biomass (B/B_0), length at first capture corresponding to maximum catch and biomass (L_{c_opt}) and the relative biomass producing maximum sustainable yields as a fraction of unexploited biomass (B_{MSY}/B_0). Several additional outputs of management importance of the model are relative fishing mortality (F/M), yield per recruit (Y/R'), and length based indicators like L_C/L_{C_opt}, L_{mean}/L_{opt}, and L/L_∞. The depletion (B/B_0) got from the LBB can fix the depletion-prior in the catch-based methods like CMSY and SRA, which often requires and independent estimates of these priors. Various steps involved in the model are given below:

Modeling the growth of fish

In order to use the increase in length as a proxy for elapsed time, the method assumes that the fish follows the Von Bertalanffy's (VBGF) growth pattern, which can be expressed as: $L_t = L_\infty \times \left(1 - exp^{-K(t-t_0)}\right)$

Where L_t is the length at age t, L_∞ is the asymptotic length, K is the growth coefficient, and t_0 is the theoretical age at which the fish's length would be zero.

Estimating the survivors-at-length in the fully exploited phase

In an exploited population, the number of survivors (N_L) at length L after full selection (N_{LStart}) is expressed using the following equation of Quinn and Deriso (1999):

$$N_L = N_{L_{start}} \times \left(\frac{L_\infty - L}{L_\infty - L_{Start}}\right)^{\frac{Z}{K}}$$

Where N_{LStart} is the number at length L_{Start} with full selection (the length at which all individuals entering the gear are retained by the gear), and Z/K is the ratio of the total mortality rate Z to the somatic growth rate K.

Estimating the proportion (probability) of survivors-at-length in the fully selected phase

As, the length frequency (LFQ) data collected from the population only reflect relative abundance, the length class-wise abundances (N_L) in the above equation are divided with the sum of all N_L values across all length classes (ΣN_L) to convert the absolute abundance to relative abundance or proportion of fish in each length class as follows:

$$\frac{N_L}{\sum N_L} = \frac{N_{L_{start}} \times \left(\frac{L_\infty - L}{L_\infty - L_{Start}}\right)^{\frac{Z}{K}}}{\sum\left(N_{L_{start}} \times \left(\frac{L_\infty - L}{L_\infty - L_{Start}}\right)^{\frac{Z}{K}}\right)}$$

This relative abundance ($N_L/\Sigma N_L$) at L represents the probability (p□$_L$) of fish to survive to a relative length L under fully exploited condition (at and above L_{C95}). By cancelling the N_{LStart} in both the numerator and denominator, the equation can be rewritten as:

$$\hat{p}_L = \frac{N_L}{\sum N_L} = \frac{\left(\frac{L_\infty - L}{L_\infty - L_{Start}}\right)^{\frac{Z}{K}}}{\sum\left(\frac{L_\infty - L}{L_\infty - L_{Start}}\right)^{\frac{Z}{K}}}$$

Estimating the proportion (probability) of survivors-at-length in the unexploited phase

In an unexploited population, as Z = M, the Z/K becomes M/K, L_{Start} becomes zero. The probability to survive to a relative length L compared to L_∞ (L/L_∞) can be derived by setting the N_{LStart} to 1 and using the restructured version of the above equation by Quinn and Deriso (1999) as follows:

$$\hat{p}_{L/L_\infty} = \left(\frac{L_\infty - L}{L_\infty}\right)^{\frac{M}{K}} = \left(1 - \frac{L}{L_\infty}\right)^{\frac{M}{K}}$$

The above equation shows that all populations with the same M/K ratio, whether small or large size, short or long-lived, herbivore or carnivore, occurring in warm or cold waters, will have the same probability of reaching a given fraction of their asymptotic length in an unexploited stage, independently of the absolute values of M, K, and L_∞. The same is also true for the fully exploited part of the population, where the probability of reaching a length beyond the fully selected length L_{start} (p□$_L$) is a function of Z/K.

Estimating the proportion (probability) of survivors-at-length in the partially exploited phase

However, for the portion of the population that is not selected fully (partially selected), the number of survivors (N_{Li}) at length L_i is calculated from the equation of Quinn and Deriso (1999) by multiplying the number of survivors in previous length class (N_{Li-1}) with the realised fishing mortality rate for the length class L (F_{Li}) (derived by multiplying the F with selectivity (S_{Li}) of the exploited gear for the species at length L_i) as follows:

$$N_{L_i} = N_{L_{i-1}} \times S_{L_i} \times F \times \left(\frac{L_\infty - L_i}{L_\infty - L_{i-1}}\right)^{\frac{M}{K} + \frac{F}{K} \times S_{L_i}}$$

Where S_{Li} is the selectivity of the exploited gear for the species at length L_i which is modeled using a logistic function, as follows:

$$S_{L_i} = \frac{1}{1 + exp^{-\alpha\,(L_i - L_c)}}$$

Where L_c is the length at which 50% of the individuals encountering the gear is captured (the length at capture, also referred as L_{c50}) and α is the steepness of the selection ogive.

Finally, the relative abundance for length Li (also the probability or proportion of fish (p□$_{Li}$) to survive to a length Li) can be derived by dividing with the length class-wise abundance (N_{Li}) with the sum of all N_{Li} values across the length classes (ΣN_{Li}) as follows:

$$\hat{p}_{L_i} = \frac{N_{L_i}}{\sum N_{L_i}} = \frac{N_{L_{i-1}} \times S_{L_i} \times F \times \left(\frac{L_\infty - L_i}{L_\infty - L_{i-1}}\right)^{\frac{M}{K} + \frac{F}{K} \times S_{L_i}}}{\sum\left(N_{L_{i-1}} \times S_{L_i} \times F \times \left(\frac{L_\infty - L_i}{L_\infty - L_{i-1}}\right)^{\frac{M}{K} + \frac{F}{K} \times S_{L_i}}\right)}$$

Cancelling the constant F in both the numerator and denominator will produce the final equation as follows:

$$\hat{p}_{L_i} = \frac{N_{L_i}}{\sum N_{L_i}} = \frac{N_{L_{i-1}} \times S_{L_i} \times \left(\frac{L_\infty - L_i}{L_\infty - L_{i-1}}\right)^{\frac{M}{K} + \frac{F}{K} \times S_{L_i}}}{\sum\left(N_{L_{i-1}} \times S_{L_i} \times \left(\frac{L_\infty - L_i}{L_\infty - L_{i-1}}\right)^{\frac{M}{K} + \frac{F}{K} \times S_{L_i}}\right)}$$

LBB implementation to fit observed proportion with expected proportion

The Length-based Bayesian (LBB) estimation method derives the observed proportion of survival at length (p_{Li}) from the available length frequency (LFQ) data and fit it the expected or modeled proportion of survival at length (p□Li) derived using the above equation with priors to estimate L∞, M/K, F/K, selectivity parameter (Lc) and steepness of selection ogive (α). The LBB is implemented within the Bayesian Gibbs sampler software JAGS (Plummer, 2003) and executed using the R-interface. The method uses a Bayesian framework (Markov Chain Monte Carlo (MCMC) methods) to estimate L∞, M/K, F/K, selectivity parameter (Lc) and steepness of selection ogive (α). It also uses a Dirichlet-Multinomial Distribution to accounts for over-dispersion in the length data.

Later, relative fishing mortality (F/M) is derived by dividing F/K with M/K and cancelling M in numerator and denominator as follows:

$$\frac{F}{M} = \frac{F}{K} \div \frac{M}{K} = \frac{F}{K} \times \frac{K}{M}$$

The length at which the biomass of the unexploited population is at maximum (L_{opt}) is derived as:

$$L_{opt} = L_\infty \times \left(\frac{3}{3+\frac{M}{K}}\right)$$

The length at capture that maximizes catch and biomass for a given fishing pressure and leads to Lopt as mean length in the catch ($Lc\__{opt}$) is got as:

$$L_{c\_opt} = \frac{L_\infty \times \left(2 + 3 \times \frac{F}{M}\right)}{\left(1+\frac{F}{M}\right) \times \left(3+\frac{M}{K}\right)}$$

The relative yield-per-recruit (Y'/R) is calculated following the formulation of Beverton and Holt (1966) as follows:

$$\frac{Y'}{R} = \frac{\frac{F}{M}}{1+\frac{F}{M}} \times \left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}} \times \left(1 - \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)}{1+\frac{1}{\frac{M}{K}+\frac{F}{K}}} + \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)^2}{1+\frac{2}{\frac{M}{K}+\frac{F}{K}}} - \frac{\left(1-\frac{L_c}{L_\infty}\right)^3}{1+\frac{3}{\frac{M}{K}+\frac{F}{K}}}\right)$$

The relative yield-per-recruit (Y'/R) is divided with fishing mortality (or more precisely with relative fishing mortality, F/M), assuming F/M is directly proportional to fishing effort, to derive CPUE'/R, which can be used as a proxy for the exploited relative biomass per recruit (B'/R) as follows:

$$\frac{B'}{R} = \frac{CPUE'}{R} = \frac{\frac{Y'}{R}}{\frac{F}{M}} = \frac{1}{1+\frac{F}{M}} \times \left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}} \times \left(1 - \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)}{1+\frac{1}{\frac{M}{K}+\frac{F}{K}}} + \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)^2}{1+\frac{2}{\frac{M}{K}+\frac{F}{K}}} - \frac{\left(1-\frac{L_c}{L_\infty}\right)^3}{1+\frac{3}{\frac{M}{K}+\frac{F}{K}}}\right)$$

The relative biomass per recruit (B_0'>L_c/R) in the exploitable phase (all the fish above L_c) of the population in the absence of any fishing (F/M = 0) is derived as:

$$\frac{B_0' > L_c}{R} = \left(1-\frac{L_c}{L_\infty}\right)^{\frac{M}{K}} \times \left(1 - \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)}{1+\frac{1}{\frac{M}{K}}} + \frac{3\times\left(1-\frac{L_c}{L_\infty}\right)^2}{1+\frac{2}{\frac{M}{K}}} - \frac{\left(1-\frac{L_c}{L_\infty}\right)^3}{1+\frac{3}{\frac{M}{K}}}\right)$$

Where B_o'>L_c/R is actually not the entire unexploited biomass, rather an exploitable fraction (all the fish above L_c) of the entire unexploited biomass (B_o)

Finally, the B'/R (or CPUE'/R) can be divided by B_o'>L_c/R to derive B/B_o'>L_c as a proxy for the depletion (B/B_o) as follows:

$$\frac{B}{B_0} = \frac{\frac{B'}{R}}{\frac{B_0' > L_c}{R}} = \frac{\frac{CPUE'}{R}}{\frac{B_0' > L_c}{R}}$$

A proxy for the relative biomass that can produce MSY (B_{msy}/B_o) is calculated by re-running equations for Y/R, B/R and B/B_o using F/M = 1 and L_c = L_{c_opt}.

LBB: R-implementation

2.15.1. Requirement for LBB

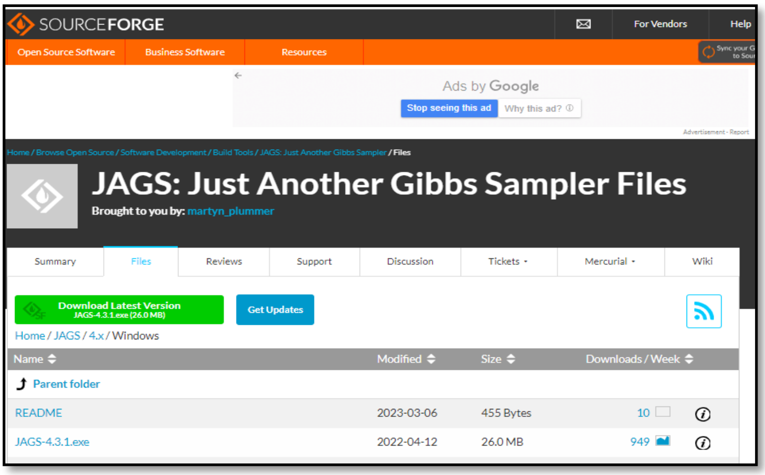
Installing dependent R-packages

LBB is not available as an R package, and therefore it can not be downloaded and installed directly using the command 'install.packages("....")' or indirectly (remotely) from the github using the command 'remotes::install_github("....")'. Therefore, the LBB approach requires three different files (mentioned below), which should be externally downloaded. The LBB analysis requires several dependent R packages **("R2jags", "Hmisc", "lattice", "survival", "Formula", "ggplot2", "crayon"**), which will be prompted for installation when the R-script (e.g., **LBB_33a.R**) is loaded for the first time on RStudio. All these dependent packages should be installed as and when they are prompted for installation.

Install the correct JAGS (Just Another Gibbs Sampler)

JAGS is Just Another Gibbs Sampler. It is a program for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation. The correct file for the Windows Operating System can be downloaded from the following website:

https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Windows/



Home / Browse Open Source / Software Development / Build Tools / JAGS: Just Another Gibbs Sampler / Files

JAGS: Just Another Gibbs Sampler Files

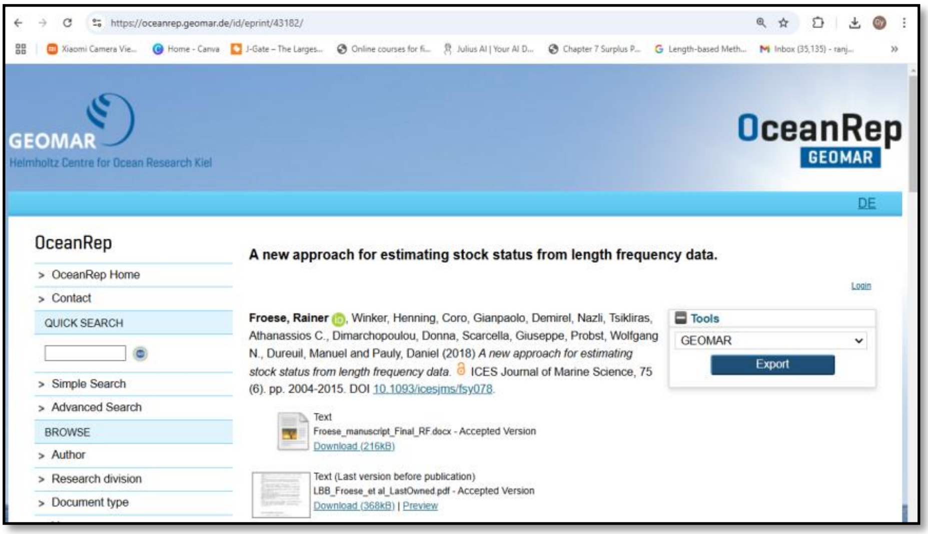
Brought to you by: martyn_plummer

Home / JAGS / 4.x / Windows

Name	Modified	Size	Downloads / Week
Parent folder			
README	2023-03-06	455 Bytes	10
JAGS-4.3.1.exe	2022-04-12	26.0 MB	949

Downloading the LBB files

LBB implementation module contains 3 essential files i.e., **(1) Example_Dat.csv** (A data file having information on annual length frequency data), **(2) Example_ID.csv** (A data file having information on various other dependent attributes), and **(3) LBB_33a.R** (An R-script file containing all the R codes for implementing LBB), which can be downloaded from the following website: http://oceanrep.geomar.de/43182/. Refer ***'Example data file download link'*** in the last page to download and use the example data.

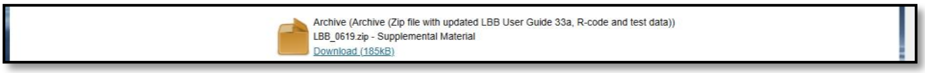


A new approach for estimating stock status from length frequency data.

Froese, Rainer, Winker, Henning, Coro, Gianpaolo, Demirel, Nazli, Tsikliras, Athanassios C., Dimarchopoulou, Donna, Scarcella, Giuseppe, Probst, Wolfgang N., Dureuil, Manuel and Pauly, Daniel (2018) *A new approach for estimating stock status from length frequency data.* ICES Journal of Marine Science, 75 (6). pp. 2004-2015. DOI 10.1093/icesjms/fsy078.

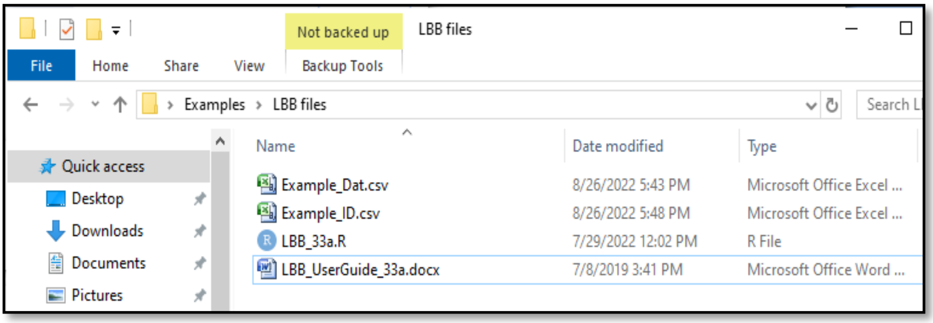
Text
Froese_manuscript_Final_RF.docx - Accepted Version
Download (216kB)

Text (Last version before publication)
LBB_Froese_et al_LastOwned.pdf - Accepted Version
Download (368kB) | Preview



Archive (Archive (Zip file with updated LBB User Guide 33a, R-code and test data))
LBB_0619.zip - Supplemental Material
Download (185kB)

Download the '**LBB_0619.zip**' and unzip (extract) the above-mentioned three files. All the three files must be stored in a single common folder. An additional user guide '**LBB_UserGuide_33a.docx**' for implementing LBB can also be found after extraction.

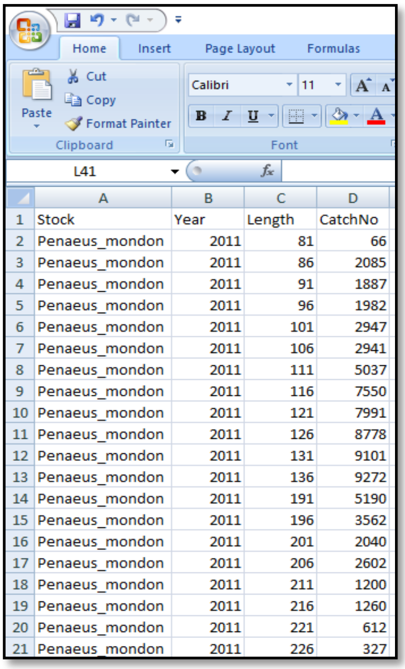


Name	Date modified	Type
Example_Dat.csv	8/26/2022 5:43 PM	Microsoft Office Excel ...
Example_ID.csv	8/26/2022 5:48 PM	Microsoft Office Excel ...
LBB_33a.R	7/29/2022 12:02 PM	R File
LBB_UserGuide_33a.docx	7/8/2019 3:41 PM	Microsoft Office Word ...

Supplying essential length frequency data and input parameters

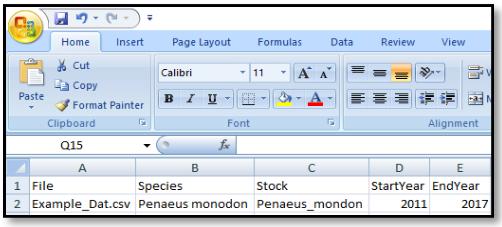
As the R-script is both spelling and case sensitive, utmost care should be taken while preparing own input data files (e.g., **Example_Dat.csv** and **Example_ID.csv**). While working with own data, it is recommended to copy and paste (overwrite) the own input data (both the 'annual length frequency' and 'ID') on the original example data supplied in the example files, i.e., '**Example_Dat.csv**' and '**Example_ID.csv**' without changing the file names of these two CSVs. The copies of the original '**Example_Dat.csv**' and '**Example_ID.csv**' can be prepared as per requirement and populated with desired information for different species.

Annual Length Frequency data (Example_Dat.csv)

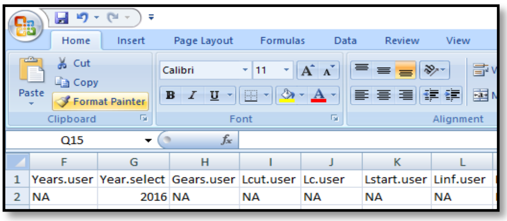


	A	B	C	D
1	Stock	Year	Length	CatchNo
2	Penaeus_mondon	2011	81	66
3	Penaeus_mondon	2011	86	2085
4	Penaeus_mondon	2011	91	1887
5	Penaeus_mondon	2011	96	1982
6	Penaeus_mondon	2011	101	2947
7	Penaeus_mondon	2011	106	2941
8	Penaeus_mondon	2011	111	5037
9	Penaeus_mondon	2011	116	7550
10	Penaeus_mondon	2011	121	7991
11	Penaeus_mondon	2011	126	8778
12	Penaeus_mondon	2011	131	9101
13	Penaeus_mondon	2011	136	9272
14	Penaeus_mondon	2011	191	5190
15	Penaeus_mondon	2011	196	3562
16	Penaeus_mondon	2011	201	2040
17	Penaeus_mondon	2011	206	2602
18	Penaeus_mondon	2011	211	1200
19	Penaeus_mondon	2011	216	1260
20	Penaeus_mondon	2011	221	612
21	Penaeus_mondon	2011	226	327

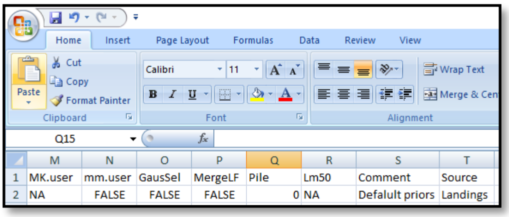
ID data (Example_ID.csv)



	A	B	C	D	E
1	File	Species	Stock	StartYear	EndYear
2	Example_Dat.csv	Penaeus monodon	Penaeus_mondon	2011	2017



	F	G	H	I	J	K	L
1	Years.user	Year.select	Gears.user	Lcut.user	Lc.user	Lstart.user	Linf.user
2	NA	2016	NA	NA	NA	NA	NA



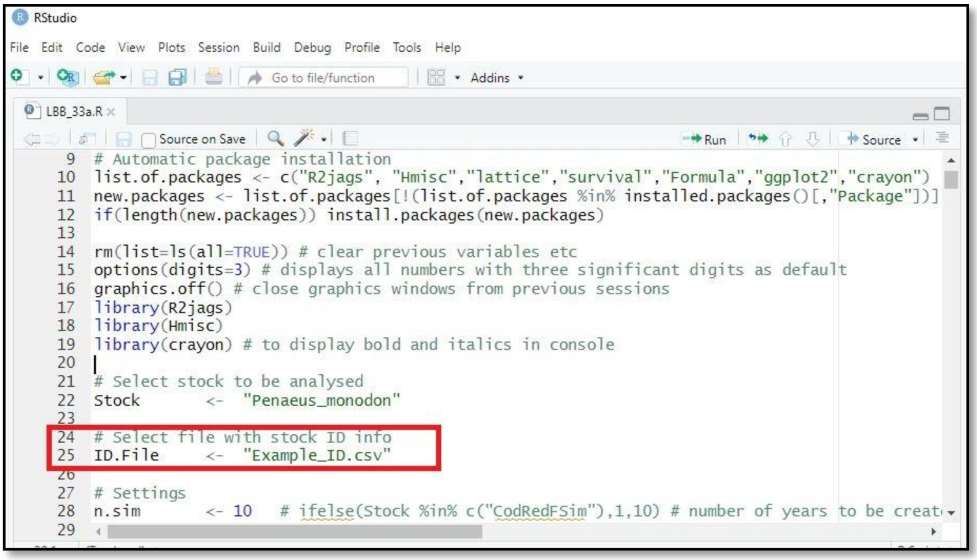
	M	N	O	P	Q	R	S	T
1	MK.user	mm.user	GausSel	MergeLF	Pile	Lm50	Comment	Source
2	NA	FALSE	FALSE	FALSE	0	NA	Defalult priors	Landings

Preparing own length frequency data file

'Example_Dat.csv' contains four columns *viz*. Stock, Year, Length and CatchNo. The first column **(Stock)** has the name of the stock (**Penaeus_monodon,** in the present example). Precaution must be taken to avoid spaces and other separators while writing the name of the stock. Underscore (_) can be used as a separator in the name of the stock. The second column **(Year)** refers to the data year (single 7 multiple year data can be used). Third column **(Length)** represents the mid-length of the class interval. It must be ensured that the mid-lengths are entered in "mm" only. The fourth column **(CatchNo)** refers to the frequency of observation made in each class interval. It may be noted that the length class with no observation must be omitted in the data file. It is recommended only to change the data inside the LFQ data file **'Example_Dat.csv'** as per the requirement but not to change the name of the '**Example_Dat.csv'.** However, if the user wants to change the name of the LFQ data file (say, for example, to '**Prawn_Dat.csv**'), then the same change must also be made to the filename in the **column A** of the ID file '**Example_ID.csv'. This is a critical point without which the analysis will fail.**

Preparing own input parameter file

The second ID data file (e.g., **Example_ID.csv**) allows the user to control the analysis. It also allows the user to provide additional information on priors (if default has to be replaced) and the other customization if required in the analysis. It is recommended only to change the parameters inside the ID data file **'Example_ID.csv'** as per the requirement but not to change the name of the '**Example_ID.csv'.** However, if the user wishes to change the name of the ID data file (say, for example, to '**Prawn_ID.csv**'), then the same change must also be made to the ID File name in the **Line no.25** of the **R-script 'LBB_33a.R'. This is a critical point without which the analysis will fail.**



```
 9  # Automatic package installation
10  list.of.packages <- c("R2jags", "Hmisc","lattice","survival","Formula","ggplot2","crayon")
11  new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
12  if(length(new.packages)) install.packages(new.packages)
13
14  rm(list=ls(all=TRUE)) # clear previous variables etc
15  options(digits=3) # displays all numbers with three significant digits as default
16  graphics.off() # close graphics windows from previous sessions
17  library(R2jags)
18  library(Hmisc)
19  library(crayon) # to display bold and italics in console
20
21  # Select stock to be analysed
22  Stock       <-  "Penaeus_monodon"
23
24  # Select file with stock ID info
25  ID.File     <-  "Example_ID.csv"
26
27  # Settings
28  n.sim       <- 10   # ifelse(Stock %in% c("CodRedFSim"),1,10) # number of years to be creat
29
```

The details of each attribute in the ID file are described below.

A. **File:** This corresponds to the name of the csv file where LFQ data is stored **(it must be exactly the same as the name of the LFQ file, e.g., Example_Dat.csv)**

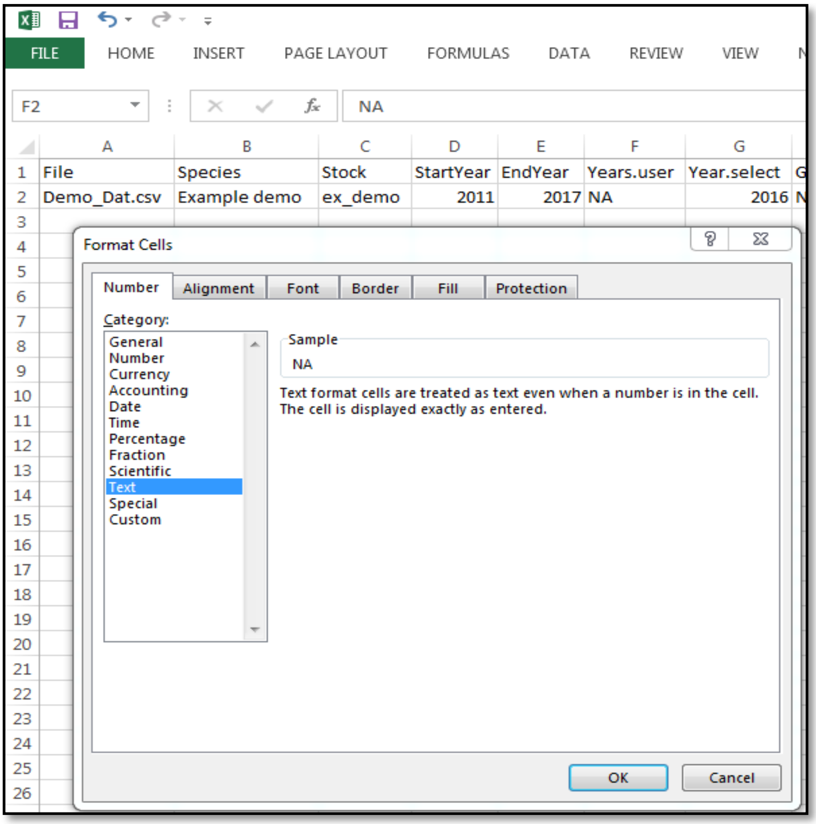
B. **Species:** The name of the species should be filled in this column.

C. **Stock:** The name of the stock should be filled (**it must be exactly the same as the name of the stock in Example_Dat.csv, e.g., Penaeus_monodon)**

D. **StartYear:** The first year in the data series (e.g., 2011 in present example). It is not a mandatory field, default is NA.

E. **EndYear:** The last year in the data series (e.g. 2017 in present example). It is not a mandatory field, default is NA.

F. **Years.user:** a list of year separated by comma (,) which are to be included in the analysis. However, first the cell needs to be formatted to have data type as "text". This can be done by right clicking on the cell and using Format Cell options to select text.



F2: NA

	A	B	C	D	E	F	G
1	File	Species	Stock	StartYear	EndYear	Years.user	Year.select
2	Demo_Dat.csv	Example demo	ex_demo	2011	2017	NA	2016

Category: Text

Sample
NA

Text format cells are treated as text even when a number is in the cell. The cell is displayed exactly as entered.

G. **Year.select:** A single year (anyone in time series) can be selected to display the depletion (B/B₀) along with its confidence interval in R console once the analysis is complete (e.g., 2016 has been selected in the present case). It is not a mandatory field, default is NA.

H. **Gears.user:** Here, the list of gears from which the data were sourced can be filled. Multiple gears can be listed with comma (,) as separator. It is not a mandatory field, default is NA.

I. **Lcut.user:** Here, the user can fix the lower limit of the length data (unit =cm) to be used in the analysis. All the mid length class below this value will not be used in the analysis. It is not a mandatory field, default is NA.

J. **Lc.user:** Here, the user can supply any prior information on length at capture, i.e., Lc50 (unit = cm). It is the length at which 50% of fish encountering the gear are caught. It is not a mandatory field, default is NA.

K. **Lstart.user:** Here, the user can supply any prior information on length at capture, i.e., Lc95 (unit = cm). It is the length at which 95% of fish encountering the gear are caught. It is not a mandatory field, default is NA.

L. **Linf.user:** Here, the user can supply any prior information on asymptotic length, i.e., L_∞ (unit = cm). It is not a mandatory field, default is NA.

M. **MK.user:** Here, the user can supply any prior information on M/K. It is not a mandatory field, default is NA.

N. **mm.user:** If we need the output to be presented in mm, use "**TRUE**" and for cm, use "**FALSE**" (which is set as Default). Irrespective of the output, the mid-length in data file **(e.g., Example_Dat.csv)** must always be in mm.

O. **GausSel:** If the selectivity is trawl type, use "FALSE" (default) and if it is gillnet type, use "TRUE".

P. **MergeLF:** If changed to "**TRUE**", it will merge the length frequencies of all the available years. Default is "**FALSE**" which means no merging applies to the data.

Q. **Pile:** information regarding correction for pile-up effect. Provide '**0**' for no pile up effect correction. Provide '**1**' for Pile up effect correction. Provide '**999**' for model to decide if pile up effect correction is required (Refer Froese et al., 2019 and Hordyk et al., 2019 for more details).

R. **Lm50:** Here, the user can supply any prior information on length at first maturity, i.e., Lm50 (unit =cm).

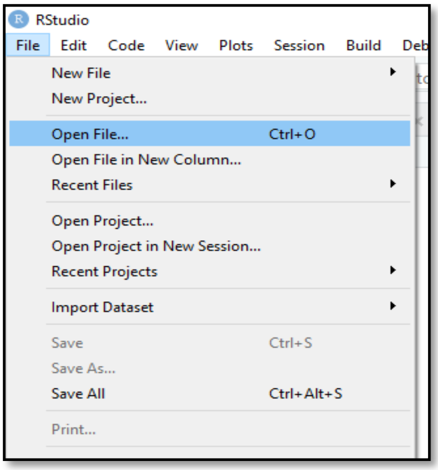
S. **Comment:** Any other info which is to be displayed along with the output (e.g., Default priors in the present example)

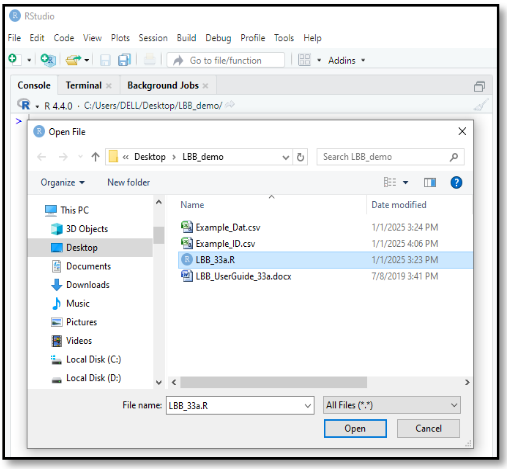
T. **Source:** The source of data can be mentioned (e.g. landings in present example)

2.15.2. Running the LBB analysis

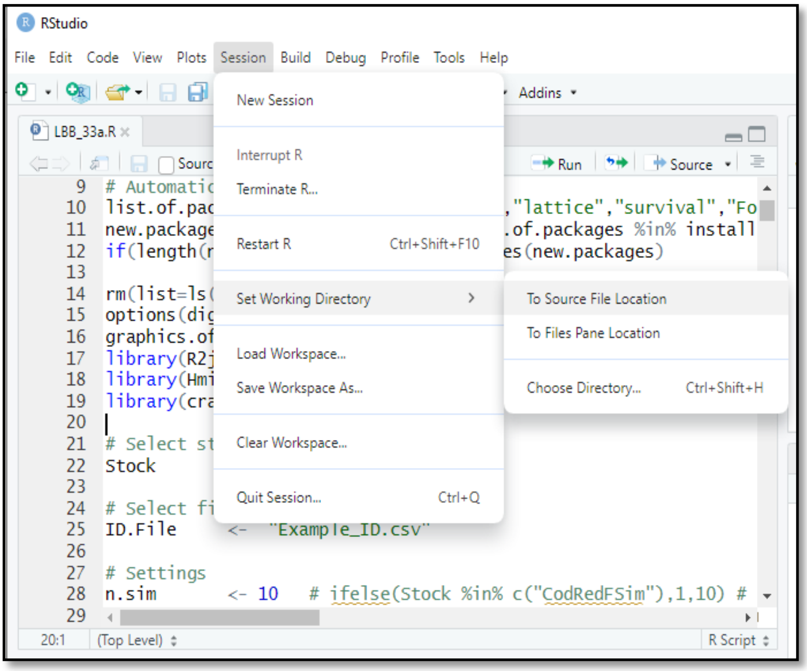
Open the R-script file in RStudio and set the working directory

Open the RStudio and click the '**File**' in RStudio and then '**Open File**...' (or simply **CTRL+O**). This will open up a browsing window to search and load the R-script file. Browse the folder where all the four files (i.e., Example_Dat.csv, Example_ID.csv, and LBB_33a.R) are previously saved and load (open) only the R-script file, i.e.**, LBB_33a.R**.





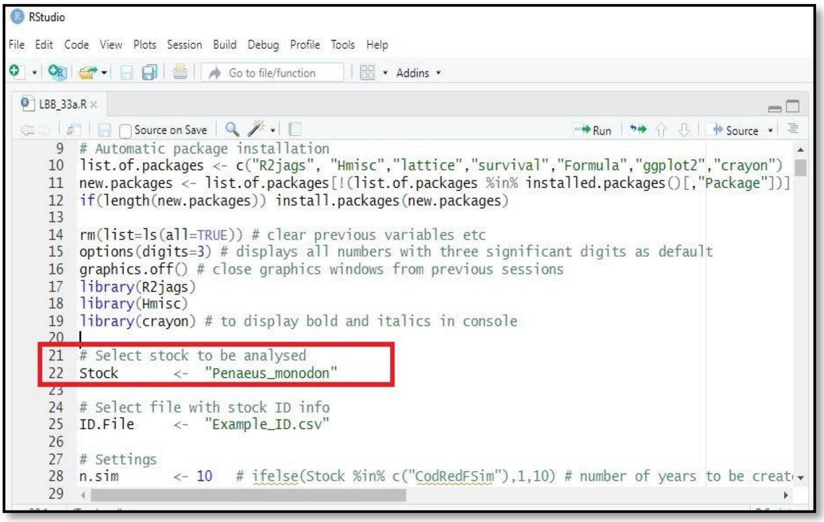
Set the working directory by clicking '**Session**' in RStudio and then select "**Set Working Directory**" to "**To Source File Location**". Setting of working directory is a crucial step which facilitates the R-script (**LBB_33a.R)** to find the remaining three files (i.e., Example_Dat.csv, Example_ID.csv) without which analysis can not be done. It also helps in storing the outputs from the analysis in the same working directory (folder).



```
 9  # Automatic
10  list.of.pac                                        ,"lattice","survival","Fo
11  new.package                                        .of.packages %in% install
12  if(length(r                                        es(new.packages)
13
14  rm(list=ls(
15  options(dic
16  graphics.of
17  library(R2j
18  library(Hmi
19  library(cra
20
21  # Select st
22  Stock
23
24  # Select fi
25  ID.File      <-   "Example_ID.csv"
26
27  # Settings
28  n.sim        <- 10   # ifelse(Stock %in% c("CodRedFSim"),1,10) #
29
```

Define the stock to be analyzed in the opened R-script

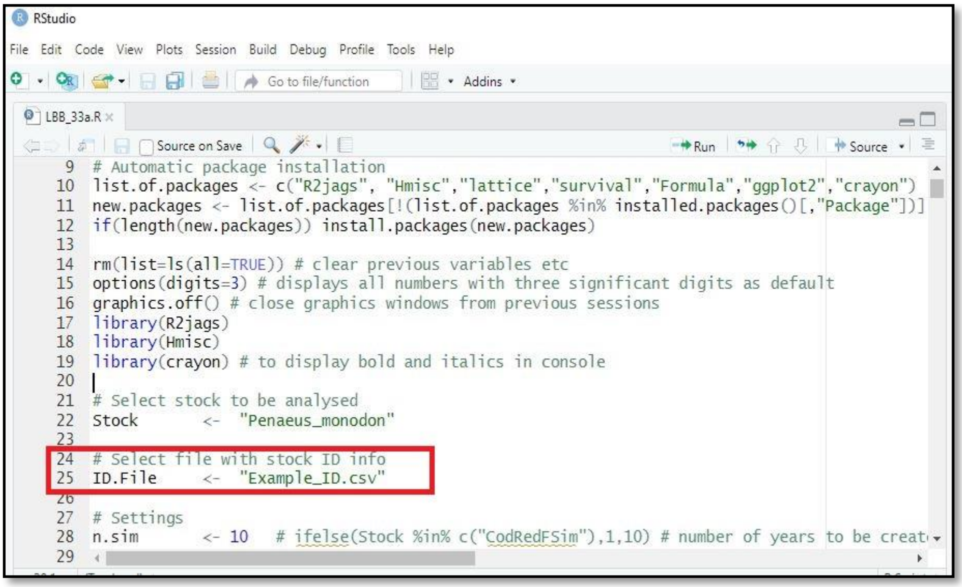
To analyze any stock in the data, go to the "**Select stock to be analyzed**" section of the R-script and enter the desired stock name (e.g., **Penaeus_monodon**) **in line 22** (**Stock <- "Penaeus_monodon"**) by replacing the example stock names or any other existing stock name. **This is a critical step** where the stock name should correctly match with the stock name mentioned in Example_Dat.csv (A column) and Example_ID.csv (C column). **Any mismatch will generate error warnings.**


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
LBB_33a.R
Source on Save
Run
Source

```
 9  # Automatic package installation
10  list.of.packages <- c("R2jags", "Hmisc","lattice","survival","Formula","ggplot2","crayon")
11  new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
12  if(length(new.packages)) install.packages(new.packages)
13
14  rm(list=ls(all=TRUE)) # clear previous variables etc
15  options(digits=3) # displays all numbers with three significant digits as default
16  graphics.off() # close graphics windows from previous sessions
17  library(R2jags)
18  library(Hmisc)
19  library(crayon) # to display bold and italics in console
20
21  # Select stock to be analysed
22  Stock       <-  "Penaeus_monodon"
23
24  # Select file with stock ID info
25  ID.File     <-  "Example_ID.csv"
26
27  # Settings
28  n.sim       <- 10   # ifelse(Stock %in% c("CodRedFSim"),1,10) # number of years to be creat
29
```

Define the ID file to be used in the opened R-script

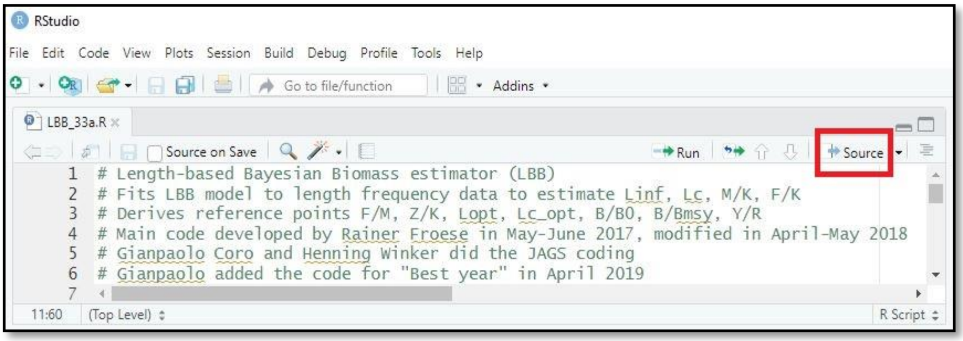
It is recommended only to change the attributes and parameters inside the ID data file **'Example_ID.csv'** without changing the name of the ID file, i.e., **'Example_ID.csv'.** If no change has been made in the name of the **'Example_ID.csv',** then there is no need to change anything in the R-script Line no.25. However, if the user has changed the name of the ID data file (say, for example, to '**Prawn_ID.csv**'), then the same change must also be made to the ID File name in the **Line no.25 of the R-script 'LBB_33a.R'. This is a critical point without which the analysis will fail.**


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
LBB_33a.R
Source on Save
Run
Source

```
 9  # Automatic package installation
10  list.of.packages <- c("R2jags", "Hmisc","lattice","survival","Formula","ggplot2","crayon")
11  new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
12  if(length(new.packages)) install.packages(new.packages)
13
14  rm(list=ls(all=TRUE)) # clear previous variables etc
15  options(digits=3) # displays all numbers with three significant digits as default
16  graphics.off() # close graphics windows from previous sessions
17  library(R2jags)
18  library(Hmisc)
19  library(crayon) # to display bold and italics in console
20
21  # Select stock to be analysed
22  Stock          <-   "Penaeus_monodon"
23
24  # Select file with stock ID info
25  ID.File        <-   "Example_ID.csv"
26
27  # Settings
28  n.sim          <- 10    # ifelse(Stock %in% c("CodRedFSim"),1,10) # number of years to be creat
29
```

Run LBB codes

In RStudio, click on "**Source**" (shown in red colour box) or press **Ctrl + Shift + S** to execute the LBB codes.

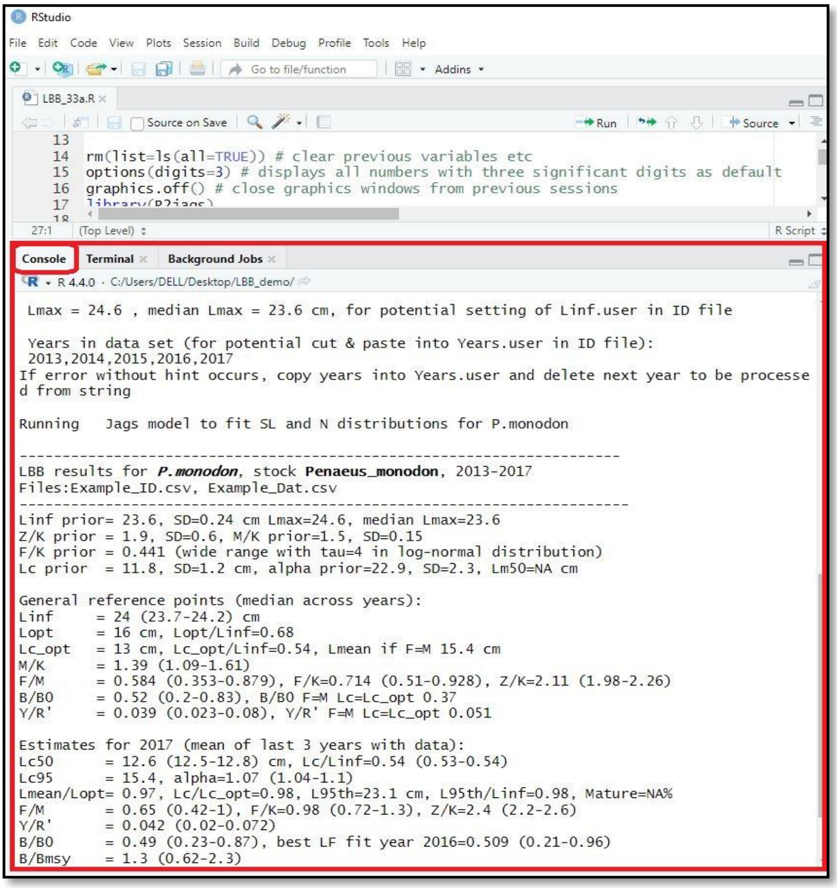


```
# Length-based Bayesian Biomass estimator (LBB)
# Fits LBB model to length frequency data to estimate Linf, Lc, M/K, F/K
# Derives reference points F/M, Z/K, Lopt, Lc_opt, B/B0, B/Bmsy, Y/R
# Main code developed by Rainer Froese in May-June 2017, modified in April-May 2018
# Gianpaolo Coro and Henning Winker did the JAGS coding
# Gianpaolo added the code for "Best year" in April 2019
```

The analysis will take some time to produce text and graphical outputs. The text output will be displayed in the R console (as depicted below) at the end once the analysis is successfully completed.

LBB text output

The analysis, though, produces the text output on the console, but it does not save them.

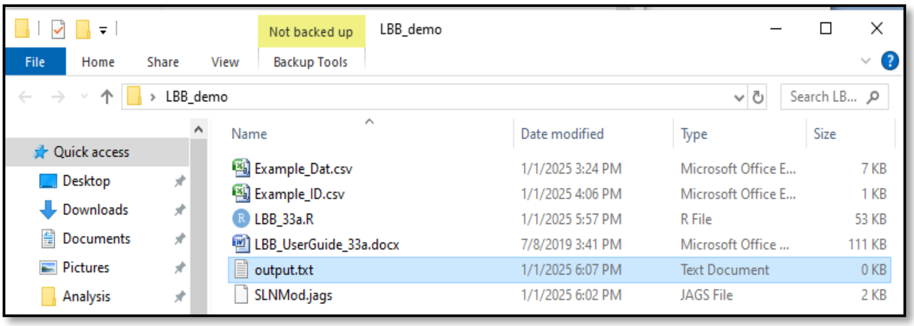


```
rm(list=ls(all=TRUE)) # clear previous variables etc
options(digits=3) # displays all numbers with three significant digits as default
graphics.off() # close graphics windows from previous sessions
library(R2jags)
```

```
 Lmax = 24.6 , median Lmax = 23.6 cm, for potential setting of Linf.user in ID file

 Years in data set (for potential cut & paste into Years.user in ID file):
 2013,2014,2015,2016,2017
If error without hint occurs, copy years into Years.user and delete next year to be processe
d from string

Running  Jags model to fit SL and N distributions for P.monodon

---------------------------------------------------------------------
LBB results for P.monodon, stock Penaeus_monodon, 2013-2017
Files:Example_ID.csv, Example_Dat.csv
---------------------------------------------------------------------
Linf prior= 23.6, SD=0.24 cm Lmax=24.6, median Lmax=23.6
Z/K prior = 1.9, SD=0.6, M/K prior=1.5, SD=0.15
F/K prior = 0.441 (wide range with tau=4 in log-normal distribution)
Lc prior  = 11.8, SD=1.2 cm, alpha prior=22.9, SD=2.3, Lm50=NA cm

General reference points (median across years):
Linf     = 24 (23.7-24.2) cm
Lopt     = 16 cm, Lopt/Linf=0.68
Lc_opt   = 13 cm, Lc_opt/Linf=0.54, Lmean if F=M 15.4 cm
M/K      = 1.39 (1.09-1.61)
F/M      = 0.584 (0.353-0.879), F/K=0.714 (0.51-0.928), Z/K=2.11 (1.98-2.26)
B/B0     = 0.52 (0.2-0.83), B/B0 F=M Lc=Lc_opt 0.37
Y/R'     = 0.039 (0.023-0.08), Y/R' F=M Lc=Lc_opt 0.051

Estimates for 2017 (mean of last 3 years with data):
Lc50       = 12.6 (12.5-12.8) cm, Lc/Linf=0.54 (0.53-0.54)
Lc95       = 15.4, alpha=1.07 (1.04-1.1)
Lmean/Lopt= 0.97, Lc/Lc_opt=0.98, L95th=23.1 cm, L95th/Linf=0.98, Mature=NA%
F/M        = 0.65 (0.42-1), F/K=0.98 (0.72-1.3), Z/K=2.4 (2.2-2.6)
Y/R'       = 0.042 (0.02-0.072)
B/B0       = 0.49 (0.23-0.87), best LF fit year 2016=0.509 (0.21-0.96)
B/Bmsy     = 1.3 (0.62-2.3)
```

To save the text outputs, paste the following code in the console and press enter. This will save the text output in the same analysis folder where the data, ID, and R-script files are present.

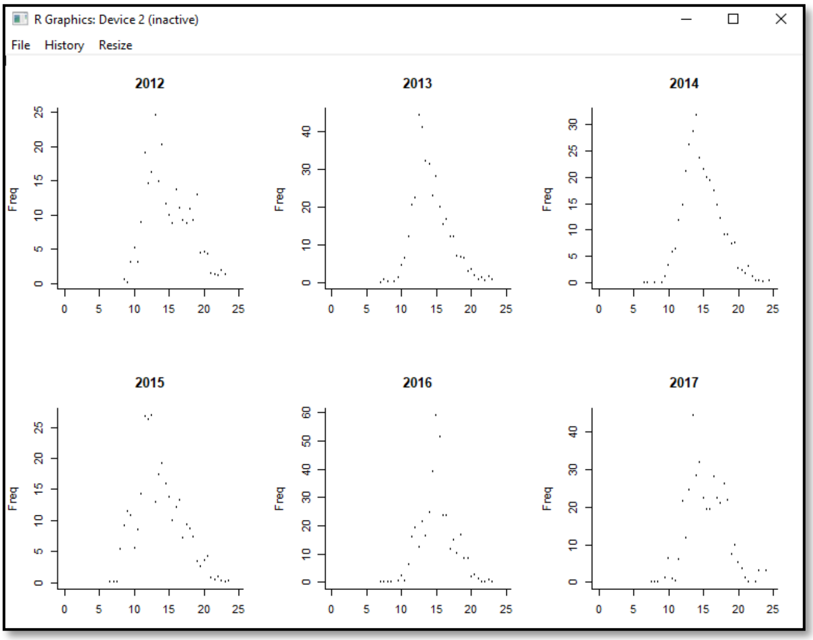
sink("./output.txt", append = T) # for saving the result in external .txt file



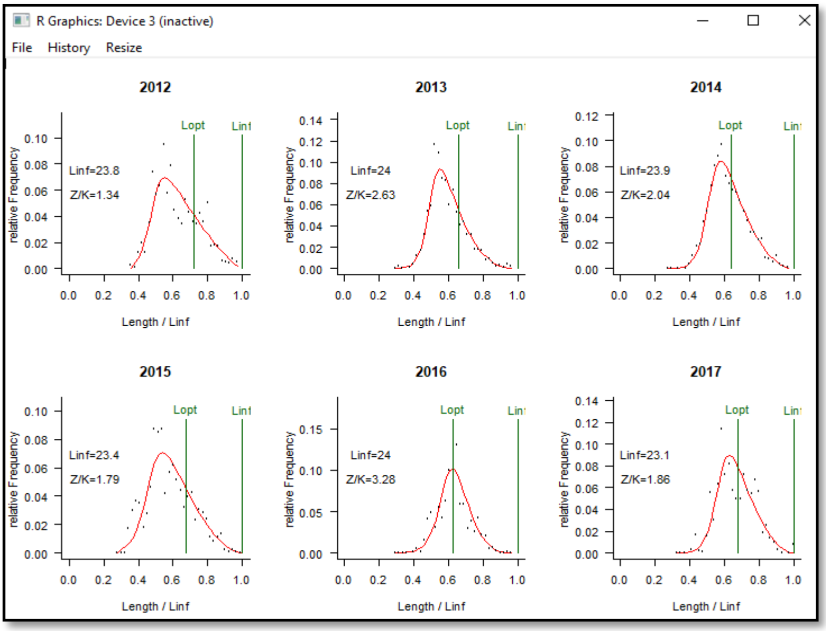
Name	Date modified	Type	Size
Example_Dat.csv	1/1/2025 3:24 PM	Microsoft Office E...	7 KB
Example_ID.csv	1/1/2025 4:06 PM	Microsoft Office E...	1 KB
LBB_33a.R	1/1/2025 5:57 PM	R File	53 KB
LBB_UserGuide_33a.docx	7/8/2019 3:41 PM	Microsoft Office ...	111 KB
output.txt	1/1/2025 6:07 PM	Text Document	0 KB
SLNMod.jags	1/1/2025 6:02 PM	JAGS File	2 KB

LBB Graphical output

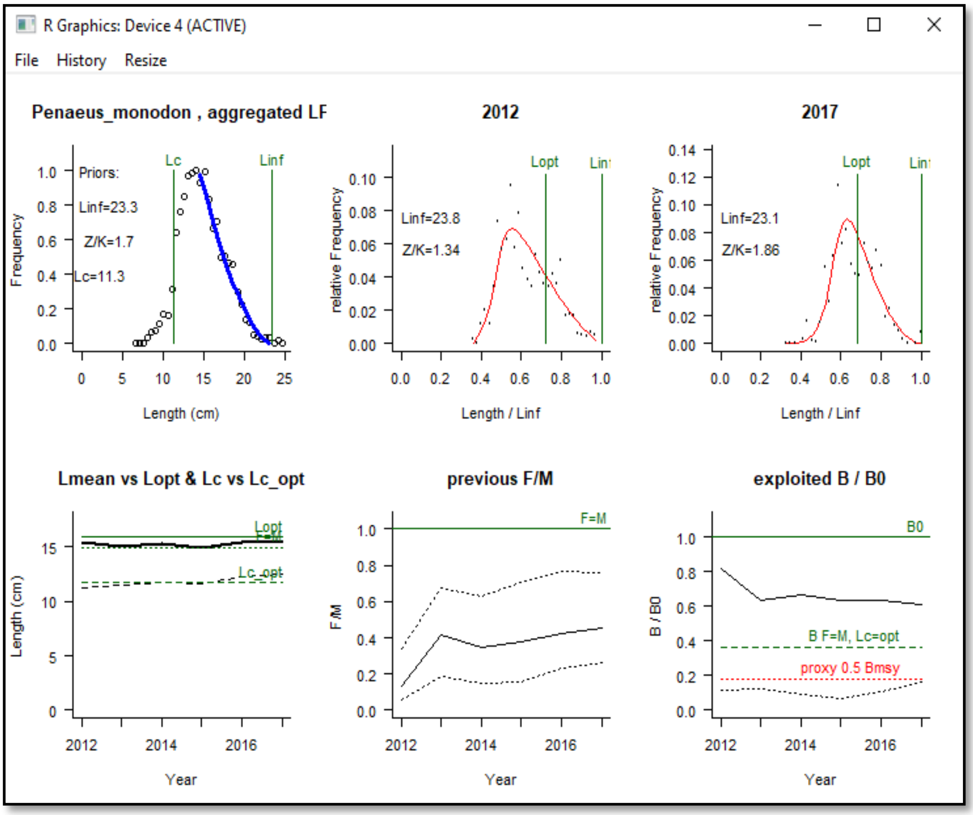
The graphical outputs are automatically displayed in separate multiple windows. The first graph shows the length frequency (LFQ) plots for the analysed years.



The second graph shows the LBB fit to the input LFQ data for the analysed years.



The third graph shows the aggregated LFQ plotted with the values of priors (L_∞, Z/K and Lc) used in the analysis. The Z/K prior was estimated based on the M/K prior and the input LFQ data. The graph also has LBB fit and estimated L_∞ and Z/K for the first and the last year is the series. The time series of L_c and L_{mean} is plotted in comparison with L_{c_opt} and L_{opt}. The other two graphical displays are the time series of F/M and B/B_0, along with their confidence intervals.



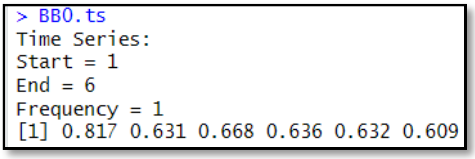
The time series of depletion, i.e., B/B_0, can be viewed in the R console using following code:

BB0.ts

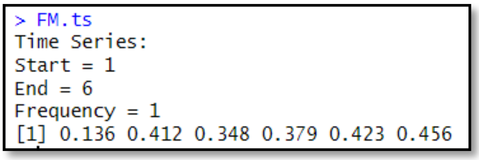
The time series of relative fishing pressure, i.e., F/M, can also be viewed in the R console using the following code:

FM.ts

B/B_0 **F/M**



```
> BB0.ts
Time Series:
Start = 1
End = 6
Frequency = 1
[1] 0.817 0.631 0.668 0.636 0.632 0.609
```

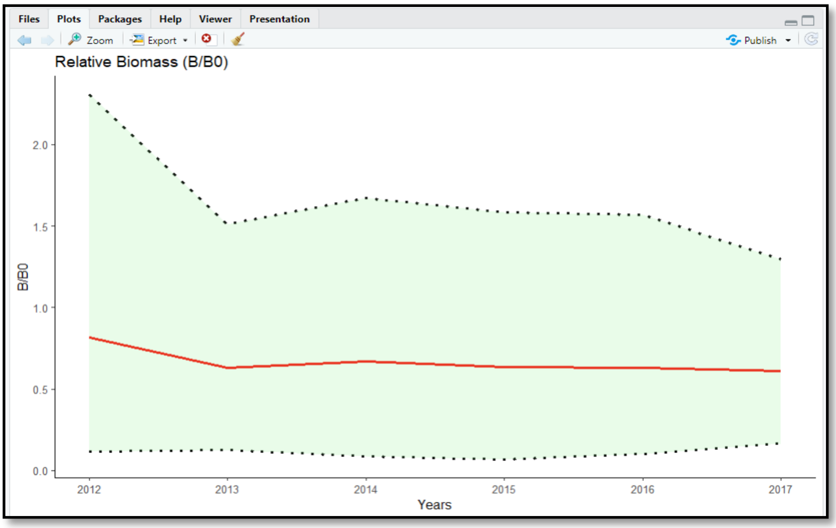


```
> FM.ts
Time Series:
Start = 1
End = 6
Frequency = 1
[1] 0.136 0.412 0.348 0.379 0.423 0.456
```

2.15.3. Plotting crucial outputs of LBB analysis (B/B_0 and F/M)

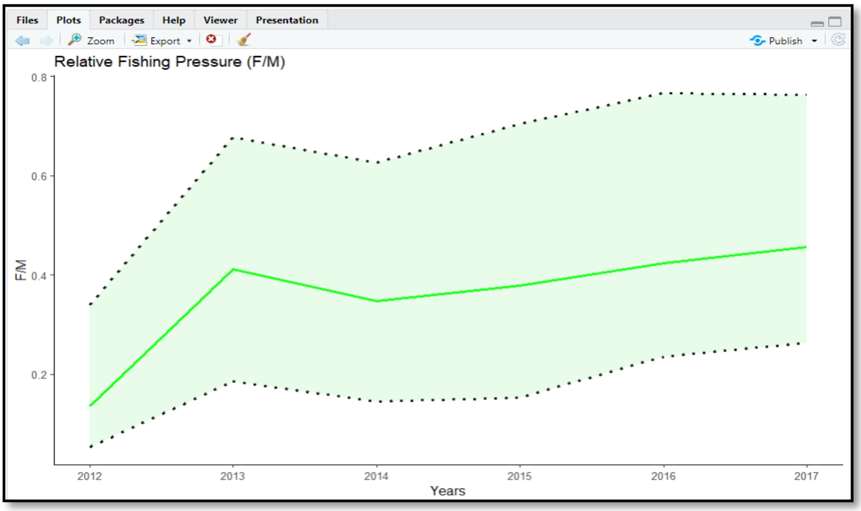
Time series of relative biomass (B/B_0) and relative fishing pressure are the two most important outputs from fisheries management aspect. These can be plotted separately for the better visualization. Following codes are required to generate the additional time series graphs. All the previous graphic windows must be closed to display these graphs.

```
#prepare a data frame for the depletion and relative fishing pressure using following codes:
results <- numeric(0)
results$x <-c(2011:2017) # the start and end year must be changed based on data
results$B_B0 <- BB0.ts
results$B_B0_lcl <- BB0.lcl.ts
results$B_B0_ucl <-BB0.ucl.ts
results$FM <- FM.ts
results$FM_lcl <- FM.lcl.ts
results$FM_ucl <-FM.ucl.ts
dat <- as.data.frame(results)
#Load the ggplot2 package for better plotting
library(ggplot2)
#Plot time series of relative depletion (B/B0) using the following code:
ggplot(dat, aes(x)) + geom_line(aes(y = B_B0_lcl), color = "black", linetype = 3, linewidth=1) + geom_line(aes(y = B_B0), color = "red", linewidth =1) + geom_line(aes(y = B_B0_ucl), color = "black", linetype = 3, linewidth =1) + geom_ribbon(aes(ymin = B_B0_lcl, ymax = B_B0_ucl), fill = "lightgreen", alpha = 0.2) + labs(title="Relative Biomass (B/B0)", x="Years", y = "B/B0")+scale_x_continuous("Years", labels = as.character(Years), breaks = Years)+ theme_classic()
```



#Plot time series of relative fishing pressure (F/M) using the following code:

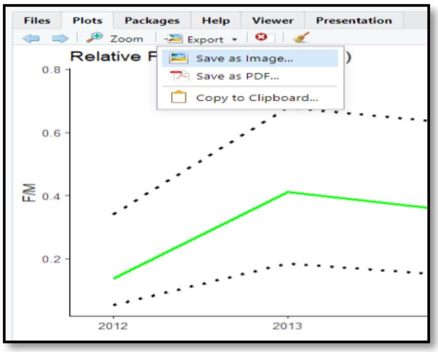
```
ggplot(dat, aes(x)) + geom_line(aes(y = FM_lcl), color = "black", linetype = 3, linewidth =1) + geom_line(aes(y = FM), color = "green", linewidth =1) + geom_line(aes(y = FM_ucl), color = "black", linetype = 3, linewidth =1) + geom_ribbon(aes(ymin = FM_lcl, ymax = FM_ucl), fill = "lightgreen", alpha = 0.2) + labs(title="Relative Fishing Pressure (F/M)", x="Years", y = "F/M") +scale_x_continuous("Years", labels = as.character(Years), breaks = Years)+ theme_classic()
```

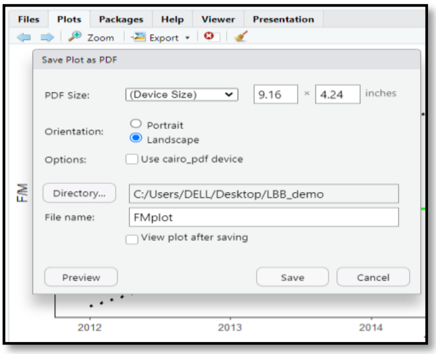


Note: *In the code marked in bold blue (results$x <-* ***c(2011:2017)****), the start and end year must be changed based on the years in the data series (e.g. 2012 to 2017 in the present example). The red and green solid lines are the values for depletion (B/B0) and relative fishing pressure (F/M) for the given year and the dotted lines are the upper and lower confidence intervals.*

Saving the graphs

The graph can be saved in the desired format by clicking on file followed by save as then choose the file format in the drop-down menu (e.g. TIFF was selected in the present example). Once the desired file format is clipped, a window to choose folder will pop-up and a graph can be saved in the desired folder.





Save Plot as PDF

PDF Size: (Device Size) 9.16 × 4.24 inches

Orientation: ○ Portrait ● Landscape

Options: □ Use cairo_pdf device

Directory... C:/Users/DELL/Desktop/LBB_demo

File name: FMplot

□ View plot after saving

Preview Save Cancel

References

Journal articles

Froese, R., Winker, H., Coro, G., Demirel, N., Tsikliras, A. C., Dimarchopoulou, D., Scarcella, G., Probst, W. N., Dureuil, M., & Pauly, D. (2018). A new approach for estimating stock status from length frequency data. *ICES Journal of Marine Science, 75*(6), 2004–2015. https://doi.org/10.1093/icesjms/fsy078

Froese, R., Winker, H., Coro, G., Demirel, N., Tsikliras, A. C., Dimarchopoulou, D., Scarcella, G., Probst, W. N., Dureuil, M., & Pauly, D. (2019). On the pile-up effect and priors for Linf and M/K: Response to a comment by Hordyk et al. on "A new approach for estimating stock status from length frequency data". *ICES Journal of Marine Science, 76*(2), 461–465. https://doi.org/10.1093/icesjms/fsy199

Froese, R., Winker, H., Gascuel, D., Sumaila, U. R., & Pauly, D. (2016). Minimizing the impact of fishing. *Fish and Fisheries, 17*, 785–802. https://doi.org/10.1111/faf.12146

Hordyk, A. R., Ono, K., Valencia, S., Loneragan, N. R., & Prince, J. D. (2015). A novel length-based empirical estimation method of spawning potential ratio (SPR), and tests of its performance, for small-scale, data-poor fisheries. *ICES Journal of Marine Science, 72*, 217–231. https://doi.org/10.1093/icesjms/fsu004

Hordyk, A. R., Prince, J. D., Carruthers, T. R., & Walters, C. J. (2019). Comment on "A new approach for estimating stock status from length frequency data" by Froese et al. (2018). *ICES Journal of Marine Science, 76*(2), 457–460. https://doi.org/10.1093/icesjms/fsy168

von Bertalanffy, L. (1938). A quantitative theory of organic growth (inquiries on growth laws. II). *Human Biology, 10*, 181–213. http://www.jstor.org/stable/41447359

Books

Quinn, T. J., & Deriso, R. B. (1999). *Quantitative fish dynamics*. Oxford University Press. p.542.

Technical papers

Beverton, R. J. H., & Holt, S. J. (1966). *Manual of methods for fish stock assessment, Part II – Tables of yield functions* (FAO Fisheries Technical Paper No. 38, Rev. 1, 10 pp.). Food and Agriculture Organization.

Conference proceedings

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22, Vienna* (pp. 20–22). Vienna Technical University.

Online resources

GEOMAR. (n.d.). *OceanRep: Repository for Ocean Publications*. http://oceanrep.geomar.de/43182/

3. Catch-based Methods

Catch-based methods play a crucial role in fisheries' stock assessment, particularly in data-limited scenarios where detailed biological or abundance data may be unavailable. These methods leverage available information on catch and effort (if available) to generate valuable insights into stock health and sustainable harvest levels. Catch-only methods (COMs) have become increasingly popular for assessing fish stocks in data poor situations where detailed biological data (e.g., age or length-based data) or detailed abundance indices are absent. These methods estimate stock status and reference points using only time-series data on catch, combined with assumptions about resilience, productivity, and exploitation history. One such approach is the Stock Reduction Analysis (SRA), which reconstructs historical stock trajectories by integrating observed catches with assumptions about biological productivity and management targets. SRA models provide insights into historical depletion levels and sustainable exploitation thresholds, making them useful tools for assessing data-poor fisheries. Under data moderate condition, when both catch and effort data are available, surplus production models, also known as dynamic pool models, are used for assessing stock status. These models, such as the Schaefer or Fox surplus production models, integrate catch and effort data to estimate population productivity, biomass trends, and key management reference points. These data-moderated models lie between purely data-limited and data-rich approaches and depend heavily on the reliable and uninfluenced abundance data to model the productivity and yield. Under data moderate condition, the commercial catch rates (catch-per-unit-effort, CPUE) are often used as an indicator of abundance. However, the CPUE is often found to have been influenced by many factors (fishing methods, technological improvements, spatio-temporal factors and environmental factors) other than abundance, which warrants some kind of effort standardization to increase the accuracy and reliability of SPMs.

3.1. Effort Standardisation

Introduction

Catch-based method uses various forms of surplus production models (Schaefer SPM, Fox SPM, Pella and Tomlinson SPM, Pella-Tomlinson-Fletcher SPM (PTF SPM) and hybrid Schaefer-Pella-Tomlinson-Fletcher SPM (Schaefer-PTF SPM)) to derive the maximum sustainable yield and associated management reference points. These SPMs, however, require information of biomass or a reliable proxy or index for abundance to construct the surplus production for the stock. Biomass or any other index of abundance (e.g., CPUE) got from fishery independent surveys (experimental surveys using swept area method or acoustic surveys) are usually recommended to derive reliable estimates from the SPM. However, such fishery independent data are very expensive and cumbersome to get because of which are not available for most of the fisheries around the world. Because of these limitations, the catch per unit effort (CPUE) derived from the commercial fisheries is commonly used as an index of abundance for the SPM. It is believed that CPUE is proportional to the biomass and increases when abundance (biomass) increases or vice versa. This relationship between CPUE and biomass (B) is expressed as:

$$CPUE = q \times B$$

Where, q is called catchability coefficients and represents the fraction of biomass that is caught by deploying a unit amount of effort (CPUE/B).

Ideally, the change in CPUE should only reflect the change in abundance for its reliability in SPM application. However, in the real-world, the CPUE is affected by many other factors other than abundance. Some common factors that can be considered as the potential influencers of the CPUE other than the abundance are given below:

(a) ***Fishing method:*** In a multi-gear and multi-species fishery, different fishing methods have different catching efficiency for different species, which is expressed as catchability coefficients (q). For example, the efficiency with which a fishing gear, e.g., gill net catches certain targeted fish, may differ from the efficiency of a trawler for the same targeted fish. Using the above-mentioned relationship between CPUE and biomass, a gear with lower 'q' for a given fish can give lower CPUE compared to the gear with higher 'q' for the same fishable biomass.

(b) ***Technological improvement:*** The efficiency of fishing method also changes over time because of technological improvements such as use of SONAR, Fish-finder device to track fishing ground, increase in engine power and net design etc. significantly increase the catchability coefficient of fishing method. This effect is called 'technological creep' or 'effort creep'. However, under the influence of 'effort creep', a higher catch and catch rate can be got from the unchanged biomass or even from a decreasing biomass. As catch rate is used as a proxy of abundance, the increase in catch rate under technological advancement can give a false impression about the increase of biomass, where it is either stagnant or even declining. The widely used catch base methods (CMSY++ and sraplus) address this issue by using a correction factor expecting an annual increase in efficiency of fishing methods (1-5%).

(c) **Spatio-temporal factors:** Change in the composition of fishing fleet, change in fishing grounds and seasonal change in fish abundance can influence CPUE. For example, fishing in different areas or seasons might yield different catch rates, unrelated to fish abundance. Similarly, environmental factors like temperature, rainfall, upwelling etc. can affect the catch rate irrespective of abundance.

The effect of any of the above-mentioned factors other than the abundance is eliminated or minimized from the CPUE data through an effort standardization process that improves the reliability of CPUE in reflecting the true abundance of the fish. Therefore, effort standardization is a foundational step in catch-based assessments. Several methods have been proposed for the standardization of fishing effort and CPUE, such as (1) the standard vessel/gear based approach by Beverton and Holt (1957), (2) the relative effort based approach by Robson (1966), (3) the derived effort based approach by Sparre (1998), (4) the multi-gear mean standardization (MGMS) by Daniel et al. (2016), and (5) the generalized linear models (GLMs) and their variants such as generalized linear mixed effects models and generalized additive models (Maunder and Punt, 2004; Zuur et al., 2009 and Okamura et al., 2018).

In case of tropical fisheries that uses multiple gears to target multiple species, fishing effort varies in terms of units (e.g., hours fished, number of vessels, engine power or net length), raw catch per unit effort (CPUE) values are not directly comparable.

Standardizing effort adjusts for these differences, often by scaling CPUE to a common baseline gear type, facilitating robust assessments of relative abundance and fishing efficiency. The present description deals only with the simple analytical framework developed by Varghese et al. (2020) for the standardization of the fishing efforts and the CPUE for a targeted species exploited by different fishing gears which is frequently encountered in tropical fisheries.

3.1.1. Effort standardisation using FESta R package

In tropical fisheries, multiple gear types (e.g., trawls, gillnets, bag nets, etc.) are commonly used to target the same species. Each gear type operates differently, requiring distinct levels of effort, such as time spent fishing, area covered, etc. The catch per unit effort (CPUE), a key measure of abundance and fishing efficiency, differs significantly across gear types because of their varying efficiency and fishing methods. The scales of CPUE for different gears may not be directly comparable. For example, a trawl might yield a much higher CPUE than a gillnet because of its larger coverage area, even when targeting the same species. Therefore, the standardization of the effort across different gear types is critical for assessing fish abundance, fishing pressure, and overall stock health in tropical regions.

This method of standardization requires the species catch, total catch and total fishing effort. Let Y_{ijk} represents the catch of kth species (k = 1, 2,..., s) from ith (i = 1, 2,..., g) gear at the jth (j = 1, 2,..., t) year and corresponding effort is expressed as X_{ij}.

Calculation of species catch proportion

The proportion of the species catch (catch or yield of the interested species, i.e., Y_{ijk}) in the total catch (Y_{ij}) is calculated for each gear and each year as follows:

$$P_{ijk} = \frac{Y_{ijk}}{Y_{ij}}$$

Calculation of mean and variance of catch proportion

The mean and variance of species catch proportion are calculated using catch proportions over the t years of observations for each gear as follows:

$$\bar{P}_{ik} = \frac{1}{t} \times \sum_{j=1}^{t} P_{ijk} \quad and \quad \sigma_{ik}^{2} = \frac{1}{t} \times \sum_{j=1}^{t} \left(P_{ijk} - \bar{P}_{ik}\right)$$

Calculation of weighing factor

The weighing factors for each gear are calculated as follows:

$$W_{ik} = \frac{\bar{P}_{ik}}{\sigma_{ik}^{2} + 1}$$

Calculation of standardised weighing factor

The calculated weighing factor for each gear is standardized to unity as follows:

$$W_{ik}' = \frac{W_{ik}}{\sum_{i=1}^{g} W_{ik}}$$

Calculation of standardised fishing effort

The standardized fishing effort for the species was calculated for every gear and year by multiplying respective total efforts with species catch proportions and standardized weighing factors for the gears as follows:

$$E_{ijk} = X_{ij} \times P_{ijk} \times W'_{ik}$$

Calculation of CPUE multiplication or conversion factor

Calculate the catch per unit effort (CPUE$_{ij}$) of every gear and year by dividing the respective total catch with total effort as follows:

$$CPUE_{ij} = \frac{Y_{ij}}{X_{ij}}$$

Calculate the mean CPUE ($\overline{CPUE}_i$) for every gear by averaging the annual CPUEs (CPUE$_{ij}$) for the respective gear as follows:

$$\overline{CPUE}_i = \frac{1}{t} \times \sum_{j=1}^{t}(CPUE_{ij})$$

Calculate the CPUE multiplication or conversion factor ($\overline{CPUE}_{imf}$) for each gear by dividing the respective mean CPUE of the gear ($\overline{CPUE}_i$) with the mean CPUE of the selected base gear ($\overline{CPUE}_{ibase}$) as follows:

$$\overline{CPUE}_{imf} = \frac{\overline{CPUE}_i}{\overline{CPUE}_{ibase}}$$

Conversion of standardised fishing effort

The standardized fishing effort of the species for every observed gear and year is finally multiplied by the CPUE multiplication or conversion factor to express the standardized effort in terms of selected base gear as follows:

$$E_{base\ ijk} = E_{ijk} \times \overline{CPUE}_{imf}$$

Calculation of standardised CPUE

The summation of these converted standardized efforts (equivalent to the selected base gear) from different gears for every year produces the total annual standardized effort in terms of selected base gear, which can be expressed as follows:

$$\sum_{i=1}^{g}(E_{base\ ijk})$$

Similarly, the gear-wise catches of every year are added to calculate the total annual catches as follows: $\sum_{i=1}^{g}(Y_{ijk})$

Finally, the standardized CPUE is calculated by dividing the annual catch with the annual standardized effort as: $\sum_{i=1}^{g}(Y_{ijk}) \,/\, \sum_{i=1}^{g}(E_{base\ ijk})$

FESta: R Implementation

3.1.2. Requirements for FESta

Installing and loading FESta R package

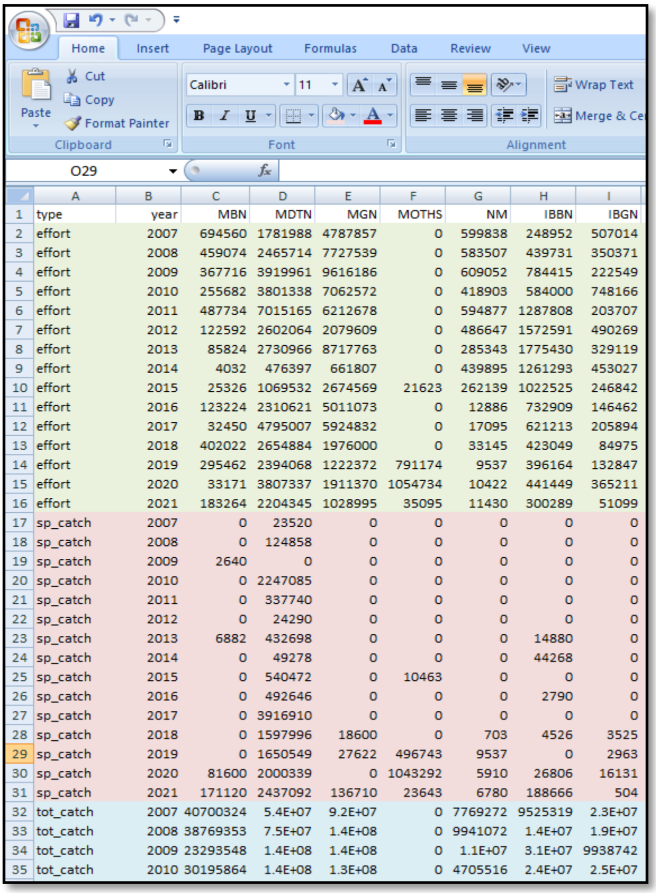
Use the following codes to install and load the '**FESta**':

```
install.packages("FESta")
library(FESta)
```

Importing the catch and effort data

The analysis requires the time-series gear-wise total efforts, total catch and catch data of the species of interest in the below-mentioned format. Refer ***'Example data file download link'*** in the last page to download and use the example data.

Catch and effort data

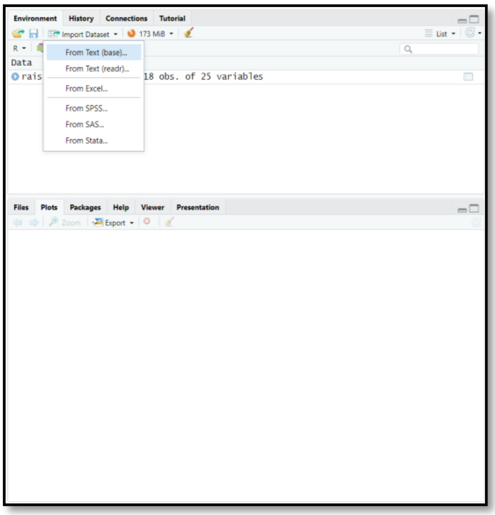


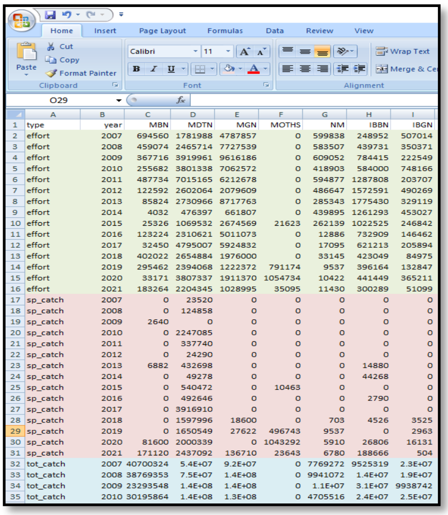
	A	B	C	D	E	F	G	H	I
1	type	year	MBN	MDTN	MGN	MOTHS	NM	IBBN	IBGN
2	effort	2007	694560	1781988	4787857	0	599838	248952	507014
3	effort	2008	459074	2465714	7727539	0	583507	439731	350371
4	effort	2009	367716	3919961	9616186	0	609052	784415	222549
5	effort	2010	255682	3801338	7062572	0	418903	584000	748166
6	effort	2011	487734	7015165	6212678	0	594877	1287808	203707
7	effort	2012	122592	2602064	2079609	0	486647	1572591	490269
8	effort	2013	85824	2730966	8717763	0	285343	1775430	329119
9	effort	2014	4032	476397	661807	0	439895	1261293	453027
10	effort	2015	25326	1069532	2674569	21623	262139	1022525	246842
11	effort	2016	123224	2310621	5011073	0	12886	732909	146462
12	effort	2017	32450	4795007	5924832	0	17095	621213	205894
13	effort	2018	402022	2654884	1976000	0	33145	423049	84975
14	effort	2019	295462	2394068	1222372	791174	9537	396164	132847
15	effort	2020	33171	3807337	1911370	1054734	10422	441449	365211
16	effort	2021	183264	2204345	1028995	35095	11430	300289	51099
17	sp_catch	2007	0	23520	0	0	0	0	0
18	sp_catch	2008	0	124858	0	0	0	0	0
19	sp_catch	2009	2640	0	0	0	0	0	0
20	sp_catch	2010	0	2247085	0	0	0	0	0
21	sp_catch	2011	0	337740	0	0	0	0	0
22	sp_catch	2012	0	24290	0	0	0	0	0
23	sp_catch	2013	6882	432698	0	0	0	14880	0
24	sp_catch	2014	0	49278	0	0	0	44268	0
25	sp_catch	2015	0	540472	0	10463	0	0	0
26	sp_catch	2016	0	492646	0	0	0	2790	0
27	sp_catch	2017	0	3916910	0	0	0	0	0
28	sp_catch	2018	0	1597996	18600	0	703	4526	3525
29	sp_catch	2019	0	1650549	27622	496743	9537	0	2963
30	sp_catch	2020	81600	2000339	0	1043292	5910	26806	16131
31	sp_catch	2021	171120	2437092	136710	23643	6780	188666	504
32	tot_catch	2007	40700324	5.4E+07	9.2E+07	0	7769272	9525319	2.3E+07
33	tot_catch	2008	38769353	7.5E+07	1.4E+08	0	9941072	1.4E+07	1.9E+07
34	tot_catch	2009	23293548	1.4E+08	1.4E+08	0	1.1E+07	3.1E+07	9938742
35	tot_catch	2010	30195864	1.4E+08	1.3E+08	0	4705516	2.4E+07	2.5E+07

Click the Import Dataset of the Environment tab (top right side panel) and then select from Excel. Browse the Excel file (e.g., 'catcheffort_data' sheet of 'catcheffort_data' excel file) and then import.

Environment> Import Dataset> from Excel and then browse the file on disk and import.

Excel Import Window **Catch and Effort data**




Home
Insert
Page Layout
Formulas
Data
Review
View
Paste
Cut
Copy
Format Painter
Clipboard
Calibri
11
Font
Wrap Text
Merge & Ce
Alignment
O29

	A	B	C	D	E	F	G	H	I
1	type	year	MBN	MDTN	MGN	MOTHS	NM	IBBN	IBGN
2	effort	2007	694560	1781988	4787857	0	599838	248952	507014
3	effort	2008	459074	2465714	7727539	0	583507	439731	350371
4	effort	2009	367716	3919961	9616186	0	609052	784415	222549
5	effort	2010	255682	3801338	7062572	0	418903	584000	748166
6	effort	2011	487734	7015165	6212678	0	594877	1287808	203707
7	effort	2012	122592	2602064	2079609	0	486647	1572591	490269
8	effort	2013	85824	2730966	8717763	0	285343	1775430	329119
9	effort	2014	4032	476397	661807	0	439895	1261293	453027
10	effort	2015	25326	1069532	2674569	21623	262139	1022525	246842
11	effort	2016	123224	2310621	5011073	0	12886	732909	146462
12	effort	2017	32450	4795007	5924832	0	17095	621213	205894
13	effort	2018	402022	2654884	1976000	0	33145	423049	84975
14	effort	2019	295462	2394068	1222372	791174	9537	396164	132847
15	effort	2020	33171	3807337	1911370	1054734	10422	441449	365211
16	effort	2021	183264	2204345	1028995	35095	11430	300289	51099
17	sp_catch	2007	0	23520	0	0	0	0	0
18	sp_catch	2008	0	124858	0	0	0	0	0
19	sp_catch	2009	2640	0	0	0	0	0	0
20	sp_catch	2010	0	2247085	0	0	0	0	0
21	sp_catch	2011	0	337740	0	0	0	0	0
22	sp_catch	2012	0	24290	0	0	0	0	0
23	sp_catch	2013	6882	432698	0	0	0	14880	0
24	sp_catch	2014	0	49278	0	0	0	44268	0
25	sp_catch	2015	0	540472	0	10463	0	0	0
26	sp_catch	2016	0	492646	0	0	0	2790	0
27	sp_catch	2017	0	3916910	0	0	0	0	0
28	sp_catch	2018	0	1597996	18600	0	703	4526	3525
29	sp_catch	2019	0	1650549	27622	496743	9537	0	2963
30	sp_catch	2020	81600	2000339	0	1043292	5910	26806	16131
31	sp_catch	2021	171120	2437092	136710	23643	6780	188666	504
32	tot_catch	2007	40700324	5.4E+07	9.2E+07	0	7769272	9525319	2.3E+07
33	tot_catch	2008	38769353	7.5E+07	1.4E+08	0	9941072	1.4E+07	1.9E+07
34	tot_catch	2009	23293548	1.4E+08	1.4E+08	0	1.1E+07	3.1E+07	9938742
35	tot_catch	2010	30195864	1.4E+08	1.3E+08	0	4705516	2.4E+07	2.5E+07

***Note:** The codes are spelling and case sensitive. Use the exact spelling and case for the different attributes mentioned in the above data format while preparing the data set.*

Formatting of imported catch and effort data

Format the freshly imported data (e.g., catcheffort_data) by splitting the imported time-series of gear-wise catch and effort data into three different data, i.e., species catch data, total catch data, and total effort data.

```
newdata<-split(catcheffort_data, catcheffort_data$type)
newdata $sp_catch<-as.data.frame(newdata $sp_catch[, -c(1)])
newdata $tot_catch<-as.data.frame(newdata $tot_catch[,-c(1)])
newdata $effort<-as.data.frame(newdata $effort[,-c(1)])
```

3.1.3. Running FESta R package

Standardize effort in terms of least efficient gear

Use the following code to standardize effort in terms of least efficient gear:

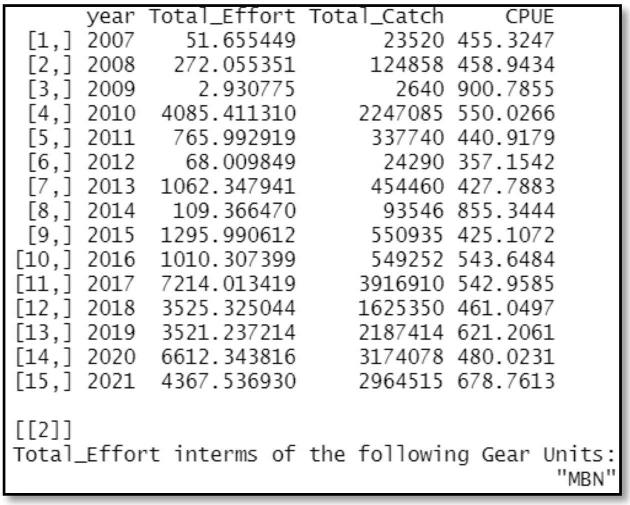
```
StdEffort(sp_catch=newdata$sp_catch,tot_catch=newdata$tot_catch,effort=newdata$effo
rt,meg=FALSE)
```

Standardize effort in terms of most efficient gear

Use the following code to standardize effort in terms of most efficient gear:

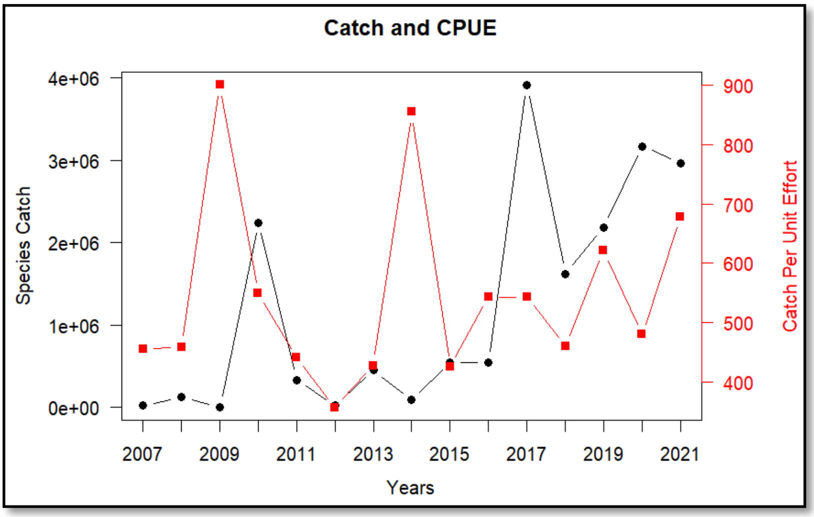
```
StdEffort(sp_catch=newdata$sp_catch,tot_catch=newdata$tot_catch,effort=newdata$effo
rt,meg=TRUE)
```

FESta tabular output



```
      year Total_Effort Total_Catch     CPUE
 [1,] 2007    51.655449       23520 455.3247
 [2,] 2008   272.055351      124858 458.9434
 [3,] 2009     2.930775        2640 900.7855
 [4,] 2010  4085.411310     2247085 550.0266
 [5,] 2011   765.992919      337740 440.9179
 [6,] 2012    68.009849       24290 357.1542
 [7,] 2013  1062.347941      454460 427.7883
 [8,] 2014   109.366470       93546 855.3444
 [9,] 2015  1295.990612      550935 425.1072
[10,] 2016  1010.307399      549252 543.6484
[11,] 2017  7214.013419     3916910 542.9585
[12,] 2018  3525.325044     1625350 461.0497
[13,] 2019  3521.237214     2187414 621.2061
[14,] 2020  6612.343816     3174078 480.0231
[15,] 2021  4367.536930     2964515 678.7613

[[2]]
Total_Effort interms of the following Gear Units:
                                            "MBN"
```

FESta graphical output



3.1.4. Effort standardization using tweaked implementation of FESta

The effort standardization process is quite straightforward when it is done using the R package '**FESta**'. However, the only limitation with the package is that it gives the standardized effort and CPUE only in terms of most efficient gear (**meg = TRUE**) or least efficient gear (**meg = FALSE**). It does not standardize the effort and CPUE in terms of gear of interest (predominantly exploiting gear). This section explains an improvised FESta implementation by a series of R codes prepared following the conceptual framework of

FESta developed by Varghese et al. (2020). For more detail, refer to '***3.1.1.Effort standardisation using FESta R package***'.

3.1.5. Requirements for tweaked implementation of FESta

Tweaked R codes following the concept of FESta

Installation of FESta R package is not required as the approach uses tweaked R codes following the concept of FESta. The calculation is basically done using a series of tweaked R codes (R-scripts).

Importing the catch and effort data

Follow the steps mentioned in '***Importing the catch and effort data***' mentioned in '***3.1.2. Requirements for FESta***' to import catch and effort data. Refer ***'Example data file download link'*** in the last page to download and use the example data.

Formatting of imported catch and effort data

Format the freshly imported data (e.g., catcheffort_data) by splitting the imported time-series of gear-wise catch and effort data into three different data, i.e., species catch data, total catch data, and total effort data.

```
newdata<-split(catcheffort_data, catcheffort_data $type)
newdata $sp_catch<-as.data.frame(newdata $sp_catch[,-c(1:2)])
newdata $tot_catch<-as.data.frame(newdata $tot_catch[,-c(1:2)])
newdata $effort<-as.data.frame(newdata $effort[,-c(1:2)])
```

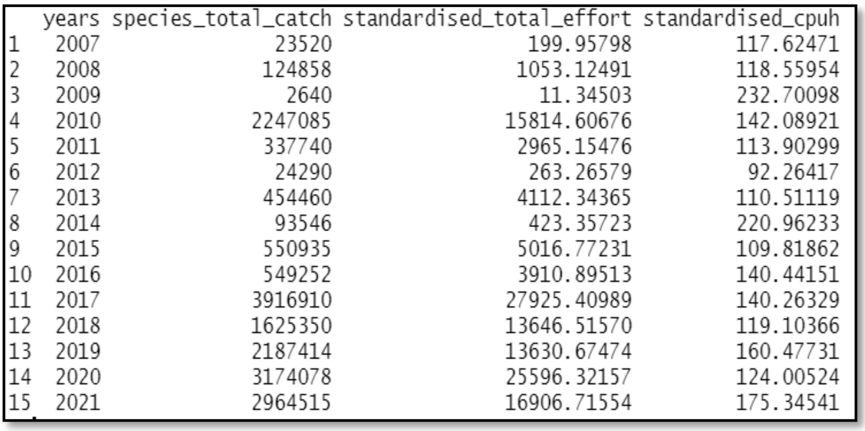
3.1.6. Running tweaked FESta R codes

Use the following R codes prepared following the conceptual framework of FESta (Varghese et al., 2020) to standardize effort and CPUE in terms of any gear of interest.

```
sp_catch_proportion<-newdata$sp_catch/newdata$tot_catch
weights<-(colMeans(sp_catch_proportion, na.rm =
TRUE)/(sapply(sp_catch_proportion, na.rm = TRUE,
var)+1)/sum(colMeans(sp_catch_proportion, na.rm =
TRUE)/(sapply(sp_catch_proportion, na.rm = TRUE, var)+1)))
weights<-as.data.frame(t(weights))
weights<-rbind(weights, weights[rep(1, nrow(sp_catch_proportion)-1), ])
row.names(weights) <- c(1:nrow(weights))
sp_effort_proportion<-newdata$effort*sp_catch_proportion*weights
gearwise_cpuh<-newdata$tot_catch/newdata$effort
gearwise_cpuh_standardization_factor<-colMeans(gearwise_cpuh, na.rm =
TRUE)/mean(gearwise_cpuh$MDTN, na.rm = TRUE)
```

Note: *In the above-mentioned code, the catch, effort and catch rate have been standardized in* *terms of multi-day trawlers (**MDTN**). The gear code of any desired gear can be mentioned in the above code to perform the standardization for that gear. For example, the standardization for the mechanized gill netters can be achieved by using **MBN** instead of **MDTN***

```
gearwise_cpuh_standardization_factor<-
as.data.frame(t(gearwise_cpuh_standardization_factor))

gearwise_cpuh_standardization_factor <-rbind(gearwise_cpuh_standardization_factor,
gearwise_cpuh_standardization_factor [rep(1, nrow(gearwise_cpuh)-1), ])

row.names(gearwise_cpuh_standardization_factor) <-
c(1:nrow(gearwise_cpuh_standardization_factor))

standardised_efforts_selected_gear<-
gearwise_cpuh_standardization_factor*sp_effort_proportion

standardised_total_effort<-rowSums (standardised_efforts_selected_gear, na.rm =
TRUE)

species_total_catch<-rowSums(newdata$sp_catch, na.rm = TRUE)

standardised_cpuh<-species_total_catch/standardised_total_effort

years<- catcheffort_data$year[1:(length(catcheffort_data$year)/3)]

standardization_results<-as.data.frame(t(rbind(years, species_total_catch,
standardised_total_effort, standardised_cpuh)))

standardization_results
```



```
   years species_total_catch standardised_total_effort standardised_cpuh
1   2007               23520                 199.95798         117.62471
2   2008              124858                1053.12491         118.55954
3   2009                2640                  11.34503         232.70098
4   2010             2247085               15814.60676         142.08921
5   2011              337740                2965.15476         113.90299
6   2012               24290                 263.26579          92.26417
7   2013              454460                4112.34365         110.51119
8   2014               93546                 423.35723         220.96233
9   2015              550935                5016.77231         109.81862
10  2016              549252                3910.89513         140.44151
11  2017             3916910               27925.40989         140.26329
12  2018             1625350               13646.51570         119.10366
13  2019             2187414               13630.67474         160.47731
14  2020             3174078               25596.32157         124.00524
15  2021             2964515               16906.71554         175.34541
```

3.1.7. Exporting the standardized catch, effort and CPUH as CSV

Use the following R codes to export the species catch, standardized effort and CPUE for the selected species by selected fishing gear:

```
write.csv(standardization_results, "C:\\Users\\Dell\\Desktop\\catch & catch
rate.csv",  row.names = FALSE)
```

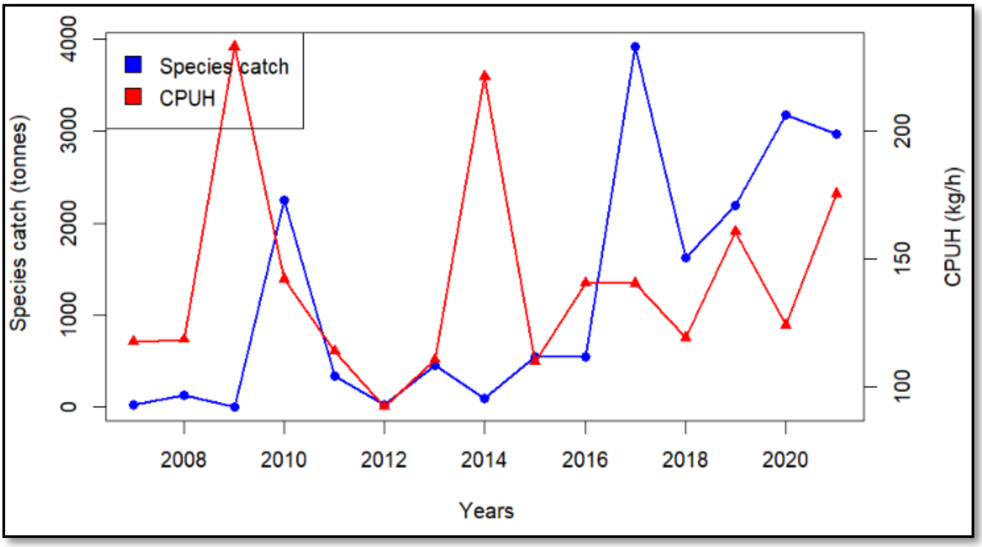
***Note:** To know the path of the location where the CSV is saved, just right click on any other file in that location > go to the 'properties' > under the 'General' copy the location of that file (ex:*

*C:\Users\Dell\Desktop). Now replace this copied file location with the bold portion of the code mentioned above and then suffix with the desired name of the file with extension (ex: **\\catch & catch rate.csv**). Remember to put a double backslash (\\) or a single forward slash (/) between each string and the entire path inside the quote mark ("....").*

3.1.8. Plotting the effort and CPUE standardization results

Use the following codes to generate the effort and CPUE standardization plots:

```
par(mar = c(5, 4, 4, 5) + 0.3)
plot(years, species_total_catch/1000, type="o", lwd=2, pch = 16, col = "blue", xlab = " Years", ylab = " Species catch (tonnes)")
par(new = TRUE)
plot(years, standardised_cpuh, type="o", lwd=2, pch = 17, col = "red", axes = FALSE, xlab = " ", ylab = "")
axis(side = 4, at = pretty(range(standardised_cpuh)))
mtext("CPUH (kg/h)", side = 4, line = 3)
legend(x = "topleft", bg="transparent", legend=c("Species catch", "CPUH"), fill = c("blue","red"))
```



References

Journal articles

Maunder, M. N., & Punt, A. E. (2004). Standardizing catch and effort data: A review of recent approaches. *Fisheries Research*, *70*(2–3), 141–159. https://doi.org/10.1016/j.fishres.2004.08.002

Okamura, H., Morita, S. H., Funamoto, T., Ichinokawa, M., & Eguchi, S. (2018). Target-based catch-per-unit-effort standardization in multispecies fisheries. *Canadian*

Journal of Fisheries and Aquatic Sciences, 75(3), 452–463. https://doi.org/10.1139/cjfas-2016-0460

Varghese, E., Sathianandan, T. V., Jayasankar, J., Kuriakose, S., Mini, K. G., & Muktha, M. (2020). Bayesian state-space implementation of Schaefer production model for assessment of stock status for multi-gear fishery. *Journal of the Indian Society of Agricultural Statistics, 74*(1), 33–40. https://eprints.cmfri.org.in/14401/

Books

Zuur, A. F., Ieno, E. N., Walker, N. J., Saveliev, A. A., & Smith, G. M. (2009). *Mixed effects models and extensions in ecology with R*, Springer New York, NY, p. 574. https://doi.org/10.1007/978-0-387-87458-6

3.2. Surplus production model (SPM)

Introduction

Surplus production models, also known as biomass dynamic models (Hilborn and Walters, 1992), are some of the simplest and most widely used tools in stock assessment. The biomass of a fish population changes over time because of natural processes such as birth, death, and growth. Each year, the biomass can be calculated based on the previous year's biomass by adding the biomass contributed by new recruits and growth, while subtracting the biomass lost due to natural mortality (Hilborn and Walters, 1992).

The biomass for the next year can be expressed as:

Next biomass = Last Biomass + Recruitment + Body Growth - Natural Mortality

However, with an exploited population, fishing mortality in the form of catch (harvesting) also reduces biomass, which can be expressed as:

Next Biomass = Last Biomass + Recruitment + Body Growth -Natural Mortality - Catch

In this context, production refers to the sum of recruitment and body growth, while surplus production is what remains after accounting for losses due to natural mortality (surplus production = production - natural mortality) (Prager, 1994). As long as the exploited catch aligns with the surplus production, the biomass will remain in a steady state.

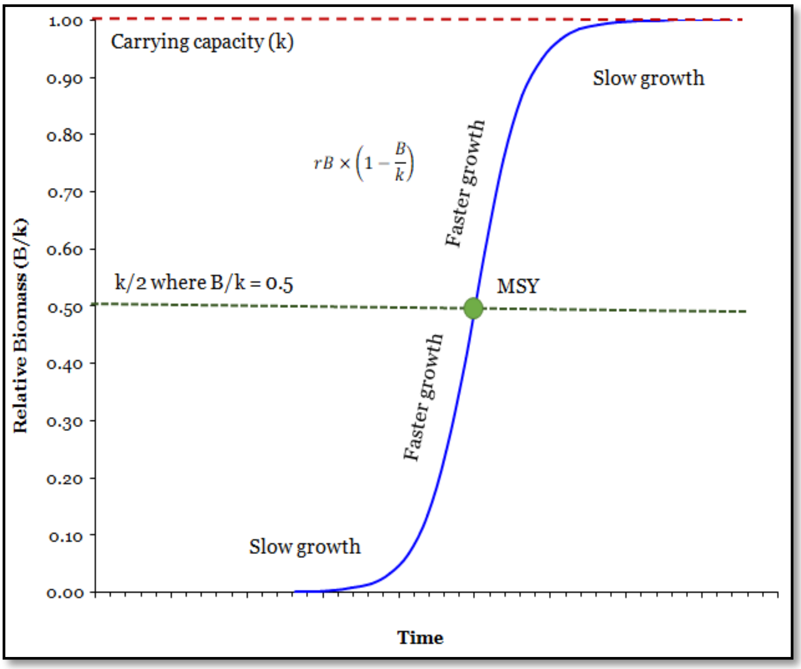
$$\text{Next biomass} = \text{Last Biomass} + \underbrace{\overbrace{\textbf{Recruitment + Body Growth}}^{\textbf{Production}} - \overbrace{\textbf{Natural Mortality - Catch}}^{\textbf{Natural Loss}}}_{\textbf{Surplus Production}}$$

The surplus production models (SPMs) are based on the ecological principle of density-dependent population growth, which is characterized by rapid exponential growth at low population sizes (whether in numbers or biomass) and a slowdown in growth as the population approaches its carrying capacity (K). In SPMs, the stock is treated as a single, undifferentiated biomass unit, with no distinction made between age, size, or sex.

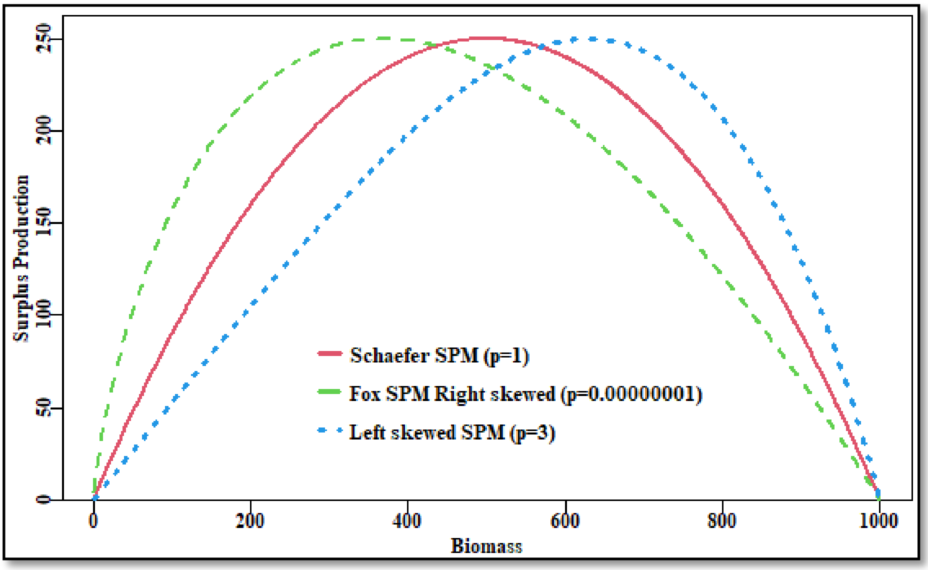
Continuous and discrete forms of the production function (dB_t/dt) in the presence of exploitation (catch) can be expressed as follows:

	SPM continuous form	SPM discrete form
Schaefer SPM model	$\frac{dB_t}{dt} = rB_t \times \left(1 - \frac{B_t}{k}\right) - C_t$	$B_{t+1} - B_t = rB_t \times \left(1 - \frac{B_t}{k}\right) - C_t$
Fox SPM model	$\frac{dB_t}{dt} = rB_t \times \left(1 - \frac{ln(B_t)}{ln(k)}\right) - C_t$	$B_{t+1} - B_t = rB_t \times \left(1 - \frac{ln(B_t)}{ln(k)}\right) - C_t$
Pella and Tomlinson SPM model	$\frac{dB_t}{dt} = \frac{r}{p} \times B_t \times \left(1 - \left(\frac{B_t}{k}\right)^p\right) - C_t$	$B_{t+1} - B_t = \frac{r}{p} \times B_t \times \left(1 - \left(\frac{B_t}{k}\right)^p\right) - C_t$

Surplus production models capture population dynamics based on logistic or more generalized theta-logistic growth principles (Pedersen et al., 2011), resulting in a dome-shaped relationship between surplus biomass production and population biomass. The Schaefer SPM (1954) produces a symmetric, dome-shaped curve, with maximum sustainable yield (MSY) occurring at half the carrying capacity (0.5K). In contrast, the Fox SPM (1970) creates an asymmetric, right-skewed dome, with MSY occurring at approximately 36.8% of the carrying capacity (0.37K). Overall, the Fox SPM generally shows higher productivity compared to the Schaefer model.



The Pella and Tomlinson SPM (Pella and Tomlinson, 1969) incorporates an additional shape parameter (p, which is also expressed as m, where p = m - 1) that influences the skewness or shape of the production function. This parameter allows the model to achieve maximum production (MSY) at any biomass level below the carrying capacity. When p approaches 1 (or m ≈ 2), the Pella and Tomlinson model closely resembles the Schaefer SPM. If p is near zero (e.g., 10−8 or m ≈ 1), the model behaves similarly to the Fox SPM, resulting in a right-skewed, asymmetric production curve. On the other hand, when p is around 2 or greater (or m ≈ 3 or higher), the model produces a left-skewed, asymmetric production curve, typical of marine mammal population dynamics. The figure below illustrates the adaptability of the Pella and Tomlinson SPM, which serves as a generalized surplus production model. This simulation assumes a carrying capacity (K) of 1000 tonnes and a maximum intrinsic population growth rate (r) of 1.



The formulas for deriving management reference points from some of the most well-known surplus production models (SPMs), including the Schaefer, Fox, and Pella-Tomlinson models, are summarized below:

Models	B_{MSY}	F_{MSY}	MSY
Schaefer SPM (1954)	$\frac{k}{2}$	$\frac{r}{2}$	$\frac{rk}{4}$
Fox SPM (1970)	$\frac{k}{exp^{(1)}}$	$\frac{r}{\ln(k)}$	$\frac{rk}{exp^{(1)} \times \ln(k)}$
Pella and Tomlinson SPM (1969)	$\frac{k}{(1+p)^{\frac{1}{p}}}$	$\frac{r}{1+p}$	$\frac{rk}{(1+p)^{\frac{1+p}{p}}}$

Note: *FMSY denotes the fishing mortality rate that results in the maximum sustainable yield (MSY). It should not be confused with E_{MSY}, which represents the effort level that generates MSY. E_{MSY} can be calculated by dividing the catchability coefficient (q) by F_{MSY} ($E_{MSY} = F_{MSY}/q$), and is sometimes referred to as f_{MSY} or f_{opt}. The catchability coefficient (q) reflects the proportion of stock biomass captured per unit of fishing effort, and is calculated as q = CPUE/B. While q is typically assumed to be constant, it can fluctuate over time because of improvements in fishing technology. For example, a trawler equipped with a fish-finding device will have a higher catchability, enabling it to catch more fish in a given time compared to a standard trawler without such equipment. This increase in fishing efficiency, resulting from technological improvements, is known as technological creep or effort creep.*

3.2.1. Equilibrium SPM

The initial deterministic versions of surplus production models (SPMs) were developed based on the equilibrium assumption. The stock is assumed to be in a long-term equilibrium state (steady state of biomass), where the biomass, recruitment, and fishing mortality have reached a stable equilibrium point with reference to a particular fishing effort. The model assumes that the observed catch and effort data represents a balance between population growth and removals by fishing. Under equilibrium conditions, the biomass remains constant over time because any surplus production is exactly balanced by losses due to fishing. Mathematically, this means the temporal rate of change of biomass (dB/dt) is zero. The continuous form of the Schaefer production function (dB/dt) under the steady state biomass (equilibrium condition) can be presented as:

$$\frac{dB}{dt} = rB \times \left(1 - \frac{B}{k}\right) - C = 0$$

Where $rB \times \left(1 - \frac{B}{k}\right)$ is the surplus production and 'C' is the catch. From the above equation, it is evident that under equilibrium conditions, surplus production and catch are equal, resulting in no net change in the original biomass.

$$rB \times \left(1 - \frac{B}{k}\right) = C$$

Under equilibrium conditions, the fishing mortality rate (F) is assumed to be constant, leading to a stable catch from the biomass (C = F × B), which is replenished by surplus production from the remaining biomass. As a result, the catch per unit effort (CPUE) (CPUE = C/E = q × B) remains steady over time at a given level of fishing effort (E) because the biomass (B) does not fluctuate significantly. CPUE will only change if there is a shift in fishing effort (E) or catchability (q), which in turn influences biomass. For example, an increase in effort (E) leads to a reduction in biomass because of greater catch (C). The remaining biomass then compensates by producing surplus, stabilizing at a new level where the surplus production matches the removals (C). This new stable biomass generates a corresponding equilibrium catch and CPUE (C/E). Over time, a gradual increase in fishing effort decreases biomass to a lower steady state, resulting in a decline in equilibrium CPUE. If effort remains constant, CPUE will stabilize at a new equilibrium, representing the sustainable catch level for that level of effort. In equilibrium models, CPUE is considered a linear indicator of biomass, and changes in CPUE are typically interpreted as changes in biomass over extended time periods.The Catch can be replaced with Effort (E), Biomass (B) and catchability coefficient (q) in the equilibrium surplus production function using the rearranged proportional relationship between CPUE and Biomass (i.e., C = q × E × B) as follows:

$$\left[rB \times \left(1 - \frac{B}{k}\right) = C\right] \rightarrow \left[rB \times \left(1 - \frac{B}{k}\right) = qEB\right]$$

By dividing both the side with B and expanding the terms, the above equation can be expressed as:

$$\left[r \times \left(1 - \frac{B}{k}\right) = q\,E\right] \rightarrow \left[r - \frac{rB}{k} = q\,E\right] \rightarrow \left[\frac{rB}{k} = r - q\,E\right] \rightarrow \left[B = k\left(1 - \frac{qE}{r}\right)\right]$$

Since stock biomass (B) is typically unknown, the observable proxy, CPUE, can be used in the equation above, based on the relationship between biomass and CPUE (i.e., B = CPUE/q), as shown below:

$$\left[B = k\left(1 - \frac{qE}{r}\right)\right] \rightarrow \left[\frac{CPUE}{q} = k\left(1 - \frac{qE}{r}\right)\right] \rightarrow \left[CPUE = qk\left(1 - \frac{qE}{r}\right)\right] \rightarrow \left[CPUE = qk - \frac{q^2k}{r}E\right]$$

Finally, by expressing 'qk' as 'a' and 'q^2K/r' as 'b,' the linear relationship between CPUE and effort (E) can be written as:

$$CPUE = a - b \times E$$

In the equation above, the negative slope (b) shows a decline in CPUE with increasing effort. The regression coefficients (a and b) can be estimated using the least squares method and subsequently applied to calculate MSY and E_{MSY} (or f_{MSY}) as follows:

$$MSY = \frac{a^2}{4b} = \frac{q^2k^2}{4 \times \frac{q^2k}{r}} = \frac{rk}{4} \quad E_{MSY} = \frac{a}{2b} = \frac{qk}{2 \times \frac{q^2k}{r}} = \frac{r}{2q}$$

Since CPUE is regressed against effort (E) to get the regression coefficients (a and b), sufficient variation (contrast) in CPUE across different effort levels is essential to enhance the reliability and accuracy of the regression. The production function, or curve, represents an equilibrium yield (catch) curve in response to equilibrium CPUE, which serves as a proxy for biomass. Each point on this curve corresponds to different equilibrium yields (catches) associated with varying equilibrium CPUE values (proxies for biomass levels). Thus, constructing a reliable production curve requires a broad range of equilibrium CPUE values. This indicates that greater contrast in CPUE data results in a more reliable SPM model. Insufficient variation in CPUE can lead to an unstable and unreliable SPM (Hilborn and Walters, 1992). Besides contrast, the SPM also requires a sufficient number of observations on catch and CPUE for the model to be reliable. A minimum time series of at least 10 years of catch and CPUE data is recommended for the model's reliability (Cousido-Rocha et al., 2022).

These models are relatively simple to apply, as they assume a constant environment and a stable relationship between stock biomass and fishing effort. However, equilibrium-based SPMs often oversimplify reality by ignoring year-to-year variability in environmental factors, recruitment, and fishing pressure. The equilibrium assumption implies that each level of fishing effort yields a sustainable catch that precisely matches the surplus production in a stable population. In practice, this assumption frequently breaks down in complex real-world conditions, leading to potential overestimation of MSY. Several major fisheries, such as the Peruvian anchovy fishery, have collapsed following catch recommendations based on equilibrium-based SPMs (Borema and Gulland, 1973; Larkin, 1977; Hilborn and Walters, 1992). This limitation of deterministic equilibrium models has rendered them obsolete, prompting the development of non-equilibrium SPMs that incorporate uncertainties (stochasticity) in the surplus production process. The examples of the SPMs that follow the equilibrium assumption are the initial deterministic versions of popular SPMs such as Schaefer SPM, Fox SPM and Pella Tomlinson SPM.

3.2.2. Non-equilibrium SPM

The biomass is assumed to be in a dynamic state, where the stock biomass fluctuates over time, not only because of a change in fishing effort but also because of recruitment variation, growth imbalance, environmental variability, etc. In non-equilibrium state the biomass changes over time, as surplus production of the stock does not balance the catch because of different factors such as over-fishing, under-fishing, recruitment variability, growth imbalance, fluctuations in natural mortalities or changes in environmental conditions etc. Mathematically, this means the rate of change of biomass (dB/dt) is not zero. The continuous form of the Schaefer production function (dB_t/dt) under the dynamic state biomass (non-equilibrium condition) can be presented as:

$$\frac{dB_t}{dt} = rB_t \times \left(1 - \frac{B_t}{k}\right) - C_t \neq 0$$

Where $rB_t \times \left(1 - \frac{B_t}{k}\right)$ is the surplus production and ‘C_t’ is the catch. From the above equation, it can be seen that under the equilibrium condition, both the surplus production and catch are not equal to each other, leading to an effective change in the original biomass.

$$rB_t \times \left(1 - \frac{B_t}{k}\right) \neq C_t$$

Two conditions are possible under non-equilibrium condition:

(1) If catch is higher than surplus production, i.e., $C > rB_t \times \left(1 - \frac{B_t}{k}\right)$, then biomass decreases, leading to a $\frac{dB_t}{dt} < 0$.

(2) If catch is higher than surplus production, i.e., $C < rB_t \times \left(1 - \frac{B_t}{k}\right)$, then biomass increases, leading to a $\frac{dB_t}{dt} > 0$

Unlike the equilibrium model, biomass is not stable and changes over time in non-equilibrium models as the stock responds to fishing pressure, recruitment variability, environmental changes, etc. Therefore, CPUE also fluctuates over time and can provide real-time insight into the stock's changing biomass. CPUE is still proportional to biomass, but it now varies with time not only in response to effort but also because of natural population dynamics, recruitment, mortality, and environmental impacts on the stock. For example, if there’s a strong recruitment year, CPUE might increase even without reducing fishing effort. Alternatively, adverse environmental conditions may reduce biomass and CPUE, even if fishing effort remains stable.

The model uses time-series data on catch and effort, assuming that the stock is not in a stable equilibrium. It accounts for temporal changes in biomass, including potential process errors (ε_{proc}) to capture environmental, ecological and biological variability, and observation errors (ε_{obs}) to address inaccuracies or noise in catch and CPUE data. These models are complex and realistic, as they reflect changes in stock size, recruitment, and fishing mortality over time. This approach is well-suited for managing fluctuations in stock size, making it more effective in capturing the true dynamics of a fishery under variable conditions. The models are particularly useful for stocks impacted by overfishing, environmental stressors, or shifts in fishing practices.

3.2.3. Incorporating uncertainty (stochasticity) in the non-equilibrium SPM

The fundamental deterministic formulation of surplus production models under equilibrium assumption are too simplistic to accurately capture the population dynamics of a real-world stock affected by variability in size structure, species interactions, recruitment, catchability, selectivity, and environmental conditions (Pella and Tomlinson 1969). Exploited stocks are influenced not only by fishery-dependent factors but also by fishery-independent factors. Fisheries data often exhibit noise because of the complex interactions of biophysical factors on the stocks, as well as sampling errors in the observation of catch or CPUE, which ultimately introduce uncertainties into the models. These uncertainties (stochasticity) are addressed by incorporating random stochastic errors into the equations (Polacheck et al., 1993 and Srinath, 2002). The stochastic errors can be categorized into two broad categories: (1) Process error and (2) Observation error.

Process error

It assumes that there are no errors in the observation of catch, CPUE, or any other index of abundance. All uncertainties (errors) in the model arise solely from the complex and dynamic biological processes involved in biomass generation within the population.

The most widely used Schaefer SPM with process error (ε_{proc}) can be expressed as:

$$B_{t+1} = B_t + rB_t \times \left(1 - \frac{B_t}{k}\right) - C_t + \varepsilon_{proc}$$

Process error (ε_t) accounts for natural fluctuations in biomass resulting from factors such as environmental changes, variations in ecological interactions, or recruitment fluctuations that are not solely because of fishing activities.

Observation error

It assumes that there are no errors in the processes related to the population dynamics of fish stocks. All uncertainty (error) in the model arises solely from inaccuracies in the observation or sampling of catch and CPUE data.

Since Catch per Unit Effort (CPUE) increases with biomass, it is frequently used as a proxy for biomass. The directly proportional relationship between CPUE and biomass (i.e., CPUE = C/E = q × B) allows to express CPUE and catch in terms of effort (E), biomass (B), and the catchability coefficient (q), along with the associated observation error (ε_{obs}) as follows:

$$CPUE_t = \frac{C_t}{E_t} = q\ B_t\ + \varepsilon_{obs}\ \ and\ \ C_t = qE_tB_t\ + \varepsilon_{obs}$$

Observation error (ε_{obs}) represents inaccuracies in the observed data (CPUE), addressing issues such as sampling errors or measurement errors in catch and biomass indices.

Both types of uncertainties (errors) are incorporated into the model using a state-space implementation of SPMs, such as the Schaefer SPM, Fox SPM, or generalized SPMs like the Pella & Tomlinson SPM. State-space models are typically fitted using one of two approaches: (1) the Bayesian approach, which employs Markov Chain Monte Carlo (MCMC) methods that uses prior distributions on parameters for better estimation of parameters and associated uncertainty, or (2) the frequentist approach, which uses maximum likelihood estimation (MLE) to maximize the log-likelihood function for the model parameters and associated uncertainty. Nevertheless, similar to deterministic SPMs, the stochastic SPMs under non-equilibrium assumptions also require sufficient contrast in catch and CPUE and a minimum of at least 10 years time series data for the model to be reliable (Cousido-Rocha et al., 2022). The examples of the SPMs that follow the non-equilibrium assumption are the stochastic versions of popular SPMs such as Schaefer SPM, Fox SPM and Pella Tomlinson SPM follow non-equilibrium assumption.

The working principle and method of implementation of commonly used stochastic SPMs under the non-equilibrium assumption such as ASPIC, SPiCT, JABBA, CMSY++ have been described in this section.

3.2.4. ASPIC (A Stock-Production Model Incorporating Covariates)

It is a widely used stock assessment tool developed by Prager in the early 1990s (Prager, 1992, 1994 and 1996) to manage fisheries with time-series data. ASPIC is particularly useful for stock assessments in data-limited or low-complexity scenarios, where minimal data input and a straightforward structure allow for effective management advice without extensive computational overhead.

Data Requirements: It requires time series data of removals (catch) and either standardized fishing effort or relative abundance (CPUE) data to assess stock dynamics. However, it does not support sub-annual data (e.g., monthly or seasonal catch and CPUE), so it is best suited for assessments based on annual data.

Error Components: It assumes observation error is associated only with CPUE, treating the catch data as error-free. Unlike some other models, ASPIC does not account for process errors associated with biomass or fishing mortality, which simplifies its structure but limits its flexibility in incorporating uncertainty in these processes.

Model Structure: It fits either a logistic (Schaefer) surplus production model or a generalized Pella–Tomlinson model. The logistic model assumes symmetrical growth around carrying capacity, while the Pella–Tomlinson model allows for an asymmetric shape in the population growth curve, which can capture different species-specific resilience and productivity levels.

Estimation Methods*:* It uses four estimation methods, i.e., Least squares, least absolute values, maximum likelihood, and maximum a posteriori (MAP), which is similar to maximum likelihood, but incorporates prior information, making it a Bayesian-like approach while remaining computationally efficient for model fitting.

ASPIC: R Implementation

The independent executable program for the ASPIC can be downloaded using the following links:

To download the earlier version (ASPIC5), use the following link:

https://noaa-fisheries-integrated-toolbox.github.io/ASPIC

To download the latest version (ASPIC7), use the following link:
https://www.mhprager.com/aspic.html

The R implementation of the ASPIC7 can be achieved through a package **connectASPIC** which can be installed using following links:
https://github.com/IMPRESSPROJECT/connectASPIC

3.2.5. SPiCT (Surplus Production model in Continuous Time)

It is a flexible and advanced stock assessment tool well-suited for fisheries with time-series data. Developed by Pedersen and Berg (2017), SPiCT is valuable for its capacity to incorporate both seasonal patterns and sub-annual data, enhancing the model's forecasting capabilities. The key features of the method are given below:

Data Requirements: It requires a time series of removals (catch data) alongside either standardized fishing effort or relative abundance (CPUE). The model can handle sub-annual (e.g., seasonal) data for both catch and CPUE, making it effective for seasonal fisheries and short-term forecasting.

Error Components: SPiCT models both observation and process errors, considering observation errors in CPUE and catch data, while process errors relate to biomass, fishing mortality, and seasonality in fishing mortality.

Model Structure: It fits a generalized Pella–Tomlinson surplus production model (SPM), a flexible approach that allows fitting to different growth rates and carrying capacities.

Estimation Methods: The model uses both the Bayesian approach with Markov Chain Monte Carlo (MCMC) and the Frequentist approach via Maximum Likelihood Estimation (MLE), making it adaptable for various estimation preferences.

SPiCT: R Implementation

The model is implemented in R through the SPiCT package, which requires the Template Model Builder (TMB) package. TMB provides the computational framework for efficient model fitting and optimization. The detailed instruction for R-implementation can be got using the following link: https://github.com/DTUAqua/spict

3.2.6. JABBA (Just another Bayesian Biomass Assessment)

This is a Bayesian stock assessment tool developed by Winker et al. (2018) well-suited for fisheries with time-series data. JABBA provides probabilistic estimates of biomass and fishing mortality, making it a valuable tool for assessing stock status and guiding fisheries.

Data Requirements: It requires a time series of removals (catch) and either standardized fishing effort or relative abundance (CPUE) data. The model can not use the sub-annual catch and relative abundance data and therefore lacks the capability to model seasonal patterns directly.

Error Components: It assumes observation error only in CPUE and treats catch data as error-free. It also includes process error for biomass only, simplifying the model by excluding fishing mortality or other errors..

Model Structure: It fits a generalized Pella–Tomlinson surplus-production model combined with a generic 'hockey stick' recruitment function to constrain the recruitment when the relative biomass (B_t/k) falls below a threshold level of 0.20 to 0.25 levels. This makes it especially useful for stocks with lower abundance and over exploitation.

Estimation Methods: It uses a Bayesian state-space approach with MCMC via JAGS (Just Another Gibbs Sampler), enabling incorporation of prior information and producing probabilistic outcomes.

JABBA: R Implementation

R implementation of the JABBA can be achieved through a package JABBA which, along with dependent JAGS program, can be downloaded and installed using the following links: https://github.com/jabbamodel/JABBA

The section presents a detailed, step-by-step guide for implementing CMSY++, offering additional flexibility to apply methods like the Catch-Only Method (COM), such as CMSY, under data-poor conditions, and Catch-and-Effort-based methods, such as BSM, under data-moderate conditions.

3.2.7. CMSY++

Introduction

This Bayesian stock assessment tool was originally developed as catch-MSY by Martell and Froese (2013) and later modified by Rosenberg et al. (2014) and Froese et al. (2017) into CMSY++. It offers flexibility for users to apply either CMSY or BSM, allowing for analysis under varying data conditions—from catch-only data in CMSY to more detailed catch and CPUE data in BSM. This adaptability makes the methods broadly applicable to data-limited fisheries, especially for initial stock assessments and management planning.

CMSY requires only time-series catch data for the target species, which is often found in data-poor condition. It also requires priors for parameters such as the intrinsic growth rate (r) and carrying capacity (k), based on species resilience and life-history traits. Additionally, it requires estimated ranges for relative stock biomass (B_t/k) in the initial and final years of the time series. BSM, in contrast, requires time-series data on both catch and biomass or index of relative abundance (e.g., CPUE), making it suitable for fisheries with data-moderate condition.

Both the CMSY and BSM use a conditional implementation of a modified Schaefer surplus-production model that includes an additional conditional multiplier of $4 \times B_t/k$ to solve the unrealistic high productivity at very low biomass

The CMSY++ uses a generalised Schaefer surplus production model when the B_t is more than a quarter of the virgin biomass (k), i.e. $B_t > 0.25$ k

$$B_{t+1} = B_t + rB_t \times \left(1 - \frac{B_t}{k}\right) - C_t$$

The CMSY++ uses a modified Schaefer surplus production model when the B_t is less than a quarter of the virgin biomass (k) i.e. $B_t < 0.25$ k. The modified SPM includes an additional conditional multiplier **4×B_t/k**. This multiplier equals to 1 when B_t/K=0.25, but when B_t/k drops below 0.25, it linearly reduces recruitment to zero as biomass approaches zero. This effectively emulates a 'hockey stick' recruitment function, similar to JABBA, and is particularly useful for assessing stocks with low abundance and high exploitation levels.

The modified version of Schaefer surplus production model is expressed as:

$$B_{t+1} = B_t + \left(\mathbf{4} \times \frac{\boldsymbol{B_t}}{\boldsymbol{k}}\right) \times rB_t \times \left(1 - \frac{B_t}{k}\right) - C_t$$

When only catch data is available (data-poor condition), CMSY is used which performs a typical stock reduction analysis (SRA) that employs users guess on the values (prior) for certain productivity indicators r, k, and initial and final year relative biomass levels (B/K) to estimate "viable" pairs of r and k. This is achieved through a Markov Chain Monte Carlo (MCMC) bootstrap approach, simulating biomass trajectories with a Schaefer SPM (or a modified version of the Schaefer SPM if biomass falls below 0.25k) that can produce the observed catch over time without exceeding the carrying capacity, and collapsing the stock or resulting in a final year depletion (B_t/K) outside the bounds of the supplied priors. The most probable r and k pairs are estimated from the viable r and k pairs which is used to estimate MSY (MSY = rk/4), and related reference points such as biomass (B_{MSY} = K/2), exploitation rates (F_{MSY} = r/2), etc.

However, under data-moderate conditions, when time-series data on catch is available along with biomass or an index of abundance (e.g., CPUE), the calculated CPUE (derived from the simulated biomass) is fitted to the observed CPUE through the implementation of Bayesian surplus production model (BSM). This approach accounts for variability (uncertainty) in both population dynamics process (process error) and the variability (uncertainty) in measurement/sampling of catch and CPUE (observation error) through the state-space implementation of the Bayesian approach using the Markov Chain Monte Carlo (MCMC) method, which is often implemented with JAGS or Stan, to derive productivity parameters (r, k and B/k).

CMSY++: R Implementation

3.2.8. Requirements for CMSY++

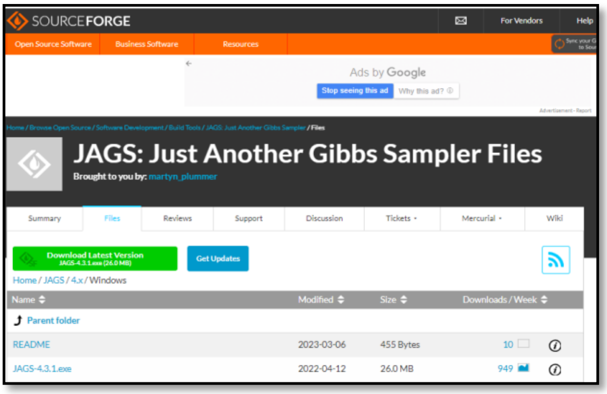
Installing dependent R-packages

CMSY++ is not available as an R package, and therefore it can not be downloaded and installed directly using the command 'install.packages("....")' or indirectly (remotely) from the github using the command 'remotes::install_github("....")'. Therefore, CMSY++ approach requires four different files (mentioned below), which should be externally downloaded. The CMSY++ includes functions for both CMSY and BSM, allowing users to seamlessly implement either model based on the data available. The LBB analysis requires several dependent R packages ("R2jags", "coda", "parallel", "foreach", "doParallel", "gplots", "mvtnorm", "snpar", "neuralnet", "conicfit"), which will be prompted for installation when the R-script (e.g., CMSY++16.R) is loaded for the first time on RStudio. All these dependent packages should be installed as and when they are prompted for installation.

Installing the JAGS (Just Another Gibbs Sampler)

JAGS is Just Another Gibbs Sampler. It is a program for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation. The correct file for the Windows Operating System can be downloaded from the following website:

https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Windows/



JAGS: Just Another Gibbs Sampler Files

Brought to you by: martyn_plummer

Home / JAGS / 4.x / Windows

Name	Modified	Size	Downloads / Week
Parent folder			
README	2023-03-06	455 Bytes	10
JAGS-4.3.1.exe	2022-04-12	26.0 MB	949

The correct file for the Mac Operating System can be downloaded from the following website:

https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Mac%20OS%20X/

Check the version of R installed in the system, using the following code

```
R.version$version.string
```

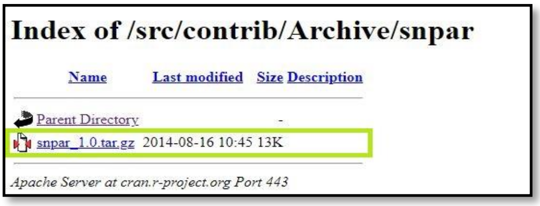
If the current version of R in the system is 4.2.0 or later, then install JAGS-4.3.2.exe

If the current version of R in the system is 4.1.3 or earlier, then install JAGS-4.3.0.exe

If the current version of R in the system is 4.1.3 or earlier, then it is strongly recommended to update the R.

Installing 'snpar' R package

The 'snpar' package is required to run the code from the source. As the package 'snpar' has been removed from R, it can not be downloaded and installed directly using the command 'install.packages("snpar")'. Therefore, first download 'snpar' from https://cran.r-project.org/src/contrib/Archive/snpar/

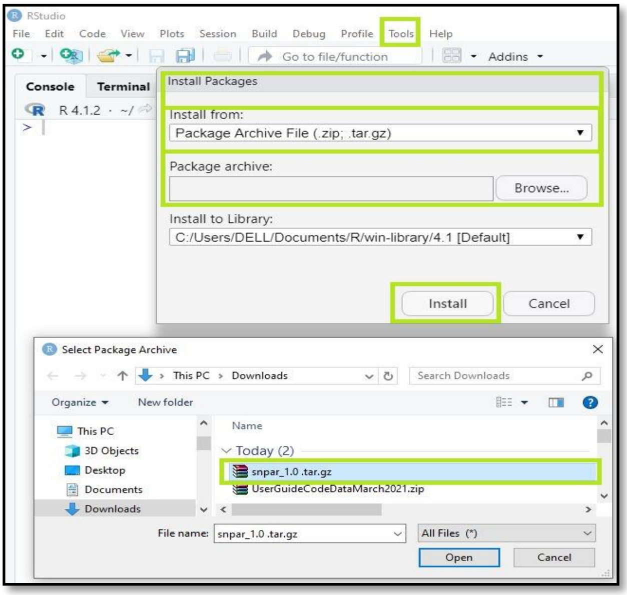


Index of /src/contrib/Archive/snpar

Name	Last modified	Size	Description
Parent Directory		-	
snpar_1.0.tar.gz	2014-08-16 10:45	13K	

Apache Server at cran.r-project.org Port 443

Click the 'snpar_1.0.tar.gz' inside the above-mentioned green border box to download the file. Install the downloaded 'snpar' in RStudio as follows: go to Tools in RStudio > Install Packages > Install From (select Package Archive File .zip, .tar.gz) > Browse under the Package archive (select the freshly downloaded snpar) > Install



Install Packages

Install from: Package Archive File (.zip; .tar.gz)

Package archive: Browse...

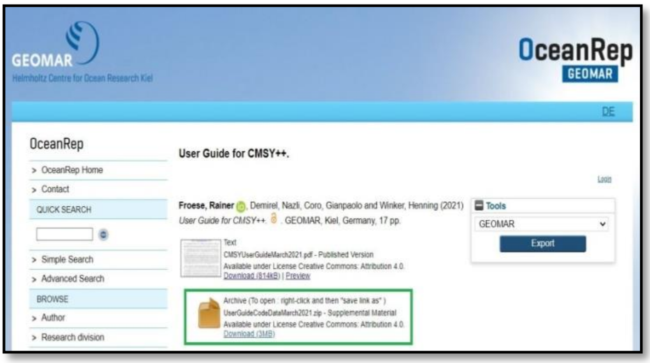
Install to Library: C:/Users/DELL/Documents/R/win-library/4.1 [Default]

snpar_1.0 .tar.gz
UserGuideCodeDataMarch2021.zip

File name: snpar_1.0 .tar.gz
All Files (*)

Downloading the CMSY++ files

CMSY++ implementation module contains 4 essential files i.e., (1) CMSY++16.R (An R-script file containing all the R codes for implementing CMSY++), (2) ffnn.bin (A file with the trained neural network data), (3) Train_Catch_9e.csv (A data file having information on catch and effort data) and (4) Train_ID_9e.csv (A data file having information on various other dependent attributes) which can be downloaded from the following website.https://oceanrep.geomar.de/id/eprint/52147/



User Guide for CMSY++.

Froese, Rainer, Demirel, Nazli, Coro, Gianpaolo and Winker, Henning (2021) *User Guide for CMSY++.* . GEOMAR, Kiel, Germany, 17 pp.

Text
CMSYUserGuideMarch2021.pdf - Published Version
Available under License Creative Commons: Attribution 4.0.
Download (814kB) | Preview

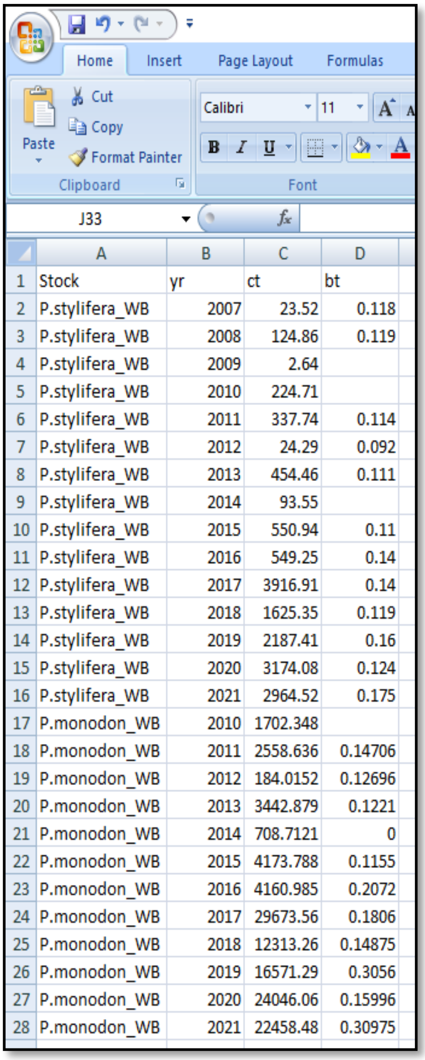
Archive (To open : right-click and then "save link as")
UserGuideCodeDataMarch2021.zip - Supplemental Material
Available under License Creative Commons: Attribution 4.0.
Download (3MB)

Click the 'Download under Archive' inside the above-mentioned green border box to download all the four files. After download, keep all the four files (i.e. CMSY++16.R, ffnn.bin, Train_Catch_9e.csv, Train_ID_9e.csv) preferably in a single folder in the same directory as the R-script (CMSY++16.R). Refer ***'Example data file download link'*** in the last page to download and use the example data.

Supplying essential catch (and effort) data and input parameters

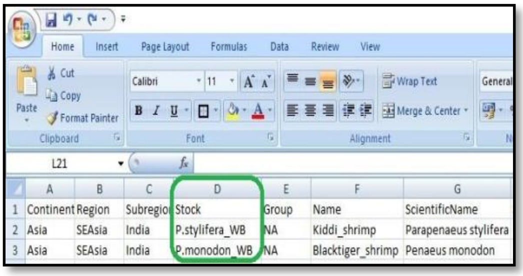
As the R-script is both spelling and case sensitive, utmost care should be taken while preparing own input data files. While working with own data, it is recommended to copy and paste (overwrite) the own input data (both the 'catch & effort' and 'ID') on the original training data supplied in the example files, i.e., 'Train_Catch_9e.csv' and 'Train_ID_9e.csv' without changing the file names of these two CSVs. The copies of the original Train_Catch_9e.csv and Train_ID_9e.csv can be prepared as per requirement and populated with desired information for different species.

Catch and effort data

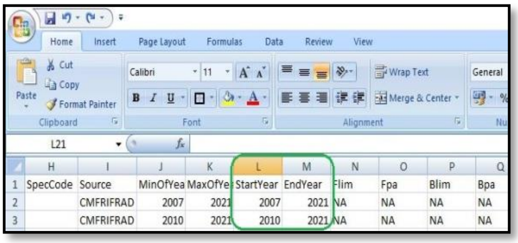


	A	B	C	D
1	Stock	yr	ct	bt
2	P.stylifera_WB	2007	23.52	0.118
3	P.stylifera_WB	2008	124.86	0.119
4	P.stylifera_WB	2009	2.64	
5	P.stylifera_WB	2010	224.71	
6	P.stylifera_WB	2011	337.74	0.114
7	P.stylifera_WB	2012	24.29	0.092
8	P.stylifera_WB	2013	454.46	0.111
9	P.stylifera_WB	2014	93.55	
10	P.stylifera_WB	2015	550.94	0.11
11	P.stylifera_WB	2016	549.25	0.14
12	P.stylifera_WB	2017	3916.91	0.14
13	P.stylifera_WB	2018	1625.35	0.119
14	P.stylifera_WB	2019	2187.41	0.16
15	P.stylifera_WB	2020	3174.08	0.124
16	P.stylifera_WB	2021	2964.52	0.175
17	P.monodon_WB	2010	1702.348	
18	P.monodon_WB	2011	2558.636	0.14706
19	P.monodon_WB	2012	184.0152	0.12696
20	P.monodon_WB	2013	3442.879	0.1221
21	P.monodon_WB	2014	708.7121	0
22	P.monodon_WB	2015	4173.788	0.1155
23	P.monodon_WB	2016	4160.985	0.2072
24	P.monodon_WB	2017	29673.56	0.1806
25	P.monodon_WB	2018	12313.26	0.14875
26	P.monodon_WB	2019	16571.29	0.3056
27	P.monodon_WB	2020	24046.06	0.15996
28	P.monodon_WB	2021	22458.48	0.30975

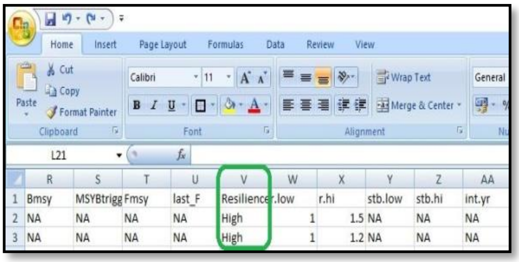
ID data



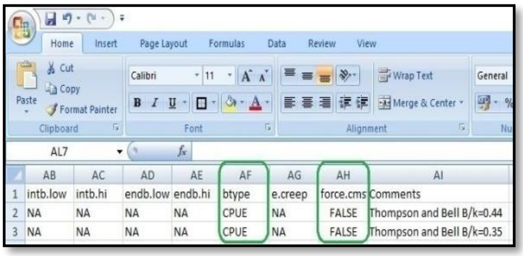
	A	B	C	D	E	F	G
1	Continent	Region	Subregio	Stock	Group	Name	ScientificName
2	Asia	SEAsia	India	P.stylifera_WB	NA	Kiddi_shrimp	Parapenaeus stylifera
3	Asia	SEAsia	India	P.monodon_WB	NA	Blacktiger_shrimp	Penaeus monodon



	H	I	J	K	L	M	N	O	P	Q
1	SpecCode	Source	MinOfYea	MaxOfYe	StartYear	EndYear	Flim	Fpa	Blim	Bpa
2		CMFRIFRAD	2007	2021	2007	2021	NA	NA	NA	NA
3		CMFRIFRAD	2010	2021	2010	2021	NA	NA	NA	NA



	R	S	T	U	V	W	X	Y	Z	AA
1	Bmsy	MSYBtrigg	Fmsy	last_F	Resilience	.low	r.hi	stb.low	stb.hi	int.yr
2	NA	NA	NA	NA	High	1	1.5	NA	NA	NA
3	NA	NA	NA	NA	High	1	1.2	NA	NA	NA



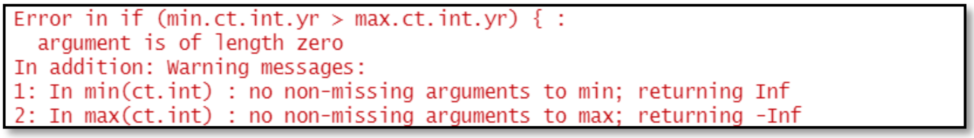
	AB	AC	AD	AE	AF	AG	AH	AI
1	intb.low	intb.hi	endb.low	endb.hi	btype	e.creep	force.cms	Comments
2	NA	NA	NA	NA	CPUE	NA	FALSE	Thompson and Bell B/k=0.44
3	NA	NA	NA	NA	CPUE	NA	FALSE	Thompson and Bell B/k=0.35

Preparing own input data file

A minimum time series of at least 10 years of continuous catch data is required for the catch-based MSY (CMSY) analysis. Populate the catch data in the 'ct' column (the C column) in Train_Catch_9e.csv file. If additional information on CPUE or biomass is available, then populate this information in the 'bt' column (the D column) in Train_Catch_9e.csv file. This additional CPUE or biomass information is essentially required to perform the Bayesian Schaefer surplus production model (BSM). It is not compulsory to provide the continuous CPUE or biomass data for every available year. Only provide the reliable CPUE or biomass data, preferably from fishery independent surveys. If fishery independent information, such as the CPUE or biomass data from experimental trawlings or acoustic surveys, is not available, then use the standardized CPUE information from the commercial fisheries. The 'bt' column (the D column) in Train_Catch_9e.csv should be populated with the standardized CPUE (here CPUH) from the effort standardization process. Refer 'Effort Standardization' to standardize effort from multi-species and multi-gear fisheries. The 'bt' column (the D column) should preferably contain CPUE information. If CPUE information is not available, then biomass or spawning stock biomass information can also be provided. The name of the stock (ex: P.stylifera_WB) should be identical both in the 'Stock' column (the A column) of the Train_Catch_9e.csv file and in the 'Stock' column (the D column) of the Train_ID_9e.csv file. If multiple stocks are available, then list them one after the other as shown in the example data files (Train_Catch_9e.csv and Train_ID_9e.csv).

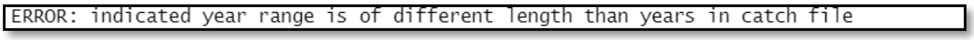
Note: *It is compulsory to provide continuous time series catch data for at least 10 years. However, it is not compulsory to provide continuous time series CPUE or biomass data for all the years. Only use the CPUE or biomass data for the credible years.*

Error message produced if time series catch data is less than 10 years



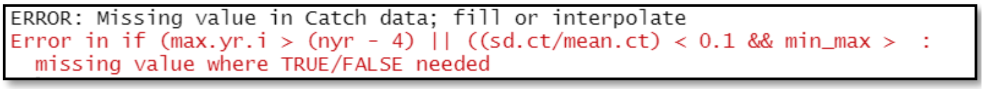
```
Error in if (min.ct.int.yr > max.ct.int.yr) { :
  argument is of length zero
In addition: Warning messages:
1: In min(ct.int) : no non-missing arguments to min; returning Inf
2: In max(ct.int) : no non-missing arguments to max; returning -Inf
```

Error message generated if time series catch data is not continuous (some years and their corresponding catches are unavailable/missing from the time series)



```
ERROR: indicated year range is of different length than years in catch file
```

Error message generated if time series catch data is not continuous (years are continuous but only for some intermittent years catches are unavailable/missing).



```
ERROR: Missing value in Catch data; fill or interpolate
Error in if (max.yr.i > (nyr - 4) || ((sd.ct/mean.ct) < 0.1 && min_max >  :
  missing value where TRUE/FALSE needed
```

In such a situation the unavailable/missing catch values need to be provided either by guessing or interpolating or taking the mean of the previous year and next year catch data. Some values need to be put to prevent error messages and carry on the analysis.

Preparing own input parameter file

The name of the stock (ex: P.stylifera_WB) should be identical both in the 'Stock' column (the A column) of the Train_Catch_9e.csv file and in the 'Stock' column (the D column) of the Train_ID_9e.csv file. The following six columns (D, L, M, V, AF and AH) in the Train_ID_9e.csv file essentially require certain inputs (information) without which the analysis does not work. Error messages will be generated if these six columns of the Train_Catch_9e.csv file were not populated with the following suggested values.

<table>
<tr><th>Column</th><th>Attribute</th><th>Action</th></tr>
<tr><td><b>D</b></td><td>Stock</td><td>compulsorily provide stock name same as the A column of the Train_Catch_9e.csv file</td></tr>
<tr><td><b>L</b></td><td>StartYear</td><td>Compulsorily provide the desired starting year for the analysis</td></tr>
<tr><td><b>M</b></td><td>EndYear</td><td>Compulsorily provide the desired end year for the analysis</td></tr>
<tr><td><b>V</b></td><td>Resilience</td><td>Compulsorily provide the resilience information such as 'High' or 'Medium' or 'Low' or 'Very low' for the species.
The resilience of the species can be guessed as
R ≈ 2× M (Natural mortality rate) or
R ≈ 2× F_{msy} (Maximum sustainable fishing mortality rate)
Refer to FishBase (https://www.fishbase.se/search.php) or SeaLifeBase (https://www.sealifebase.se/search.php) to get prior information on the species resilience.
As per these sites, the r has been categorized as
<table>
<tr><th>Resilience</th><th>Prior r range</th></tr>
<tr><td>High</td><td>0.6-1.5</td></tr>
<tr><td>Medium</td><td>0.2-0.8</td></tr>
<tr><td>Low</td><td>0.05-0.5</td></tr>
<tr><td>Very low</td><td>0.015-0.1</td></tr>
</table></td></tr>
<tr><td><b>AF</b></td><td>btype</td><td>Compulsorily provide the information as 'CPUE' or 'biomass' or 'None'.
CPUE or biomass: If CPUE or Biomass information is available, provide such input to additionally perform BSM analysis.
None: To perform only the CMSY analysis under the circumstances where actually such information is not available. Also to suppress the CPUE or biomass information where such information is available to perform only the</td></tr>
</table>

		CMSY analysis.
AH	force.cmsy	Compulsorily provide the information as 'TRUE' or 'FALSE'. TRUE: To get management outputs from the CMSY. FALSE: To get management outputs from the BSM.
Rest of the columns		Not compulsory to provide information. Mention as not available (NA). When NA is provided, the numerical values are guessed by the neural network file (ffnn.bin)

If multiple stocks are available, then list them one after the other as shown in the example data files (Train_Catch_9e.csv and Train_ID_9e.csv).

Controlling the analysis

To do only the catch-based CMSY analysis

This requires only the time series catch information (i.e., populated ct column or C column) in the Train_Catch_9e.csv. However, in the absence of CPUE or biomass information, it is essential to mention the btype column (the AF column shown in green) in the Train_ID_9e.csv as None. If additional information on the CPUE or biomass (i.e., populated bt column or D column) is available in the Train_Catch_9e.csv, then suppress such information by mentioning the btype column (the AF column shown in green) in the Train_ID_9e.csv as None. This will trigger the analysis to follow only the CMSY approach suppressing the BSM approach in the absence of CPUE or biomass information.

To do both the catch-based CMSY analysis and catch and effort-based BSM analysis

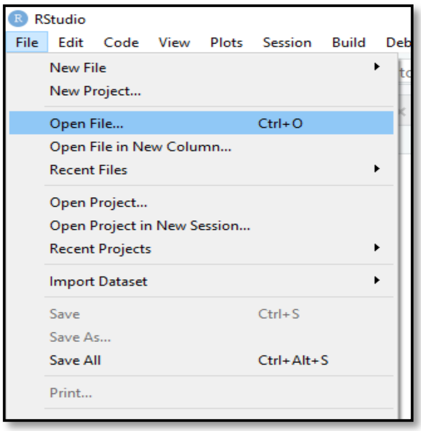
This requires both the Catch (i.e., populated ct column or C column) and CPUE or biomass (i.e., populated bt column or D column) information in the Train_Catch_9e.csv. In the presence of CPUE or biomass information, it is essential to mention the btype column (the AF column) in the Train_ID_9e.csv as CPUE or biomass. This will trigger the analysis to follow both the catch-based MSY (CMSY) and Bayesian Schaefer surplus production model (BSM) approaches.

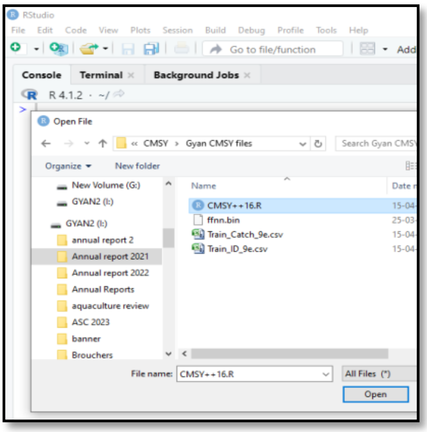
1. To get management results only from the CMSY analysis, mention the force.cmsy (AH column shown in red) as TRUE in the Train_ID_9e.csv.
2. To get management results only from the BSM analysis, mention the force.cmsy (AH column shown in red) as FALSE in the Train_ID_9e.csv.

3.2.9. Running the CMSY++ analysis

Open the R-script file in RStudio and set the working directory

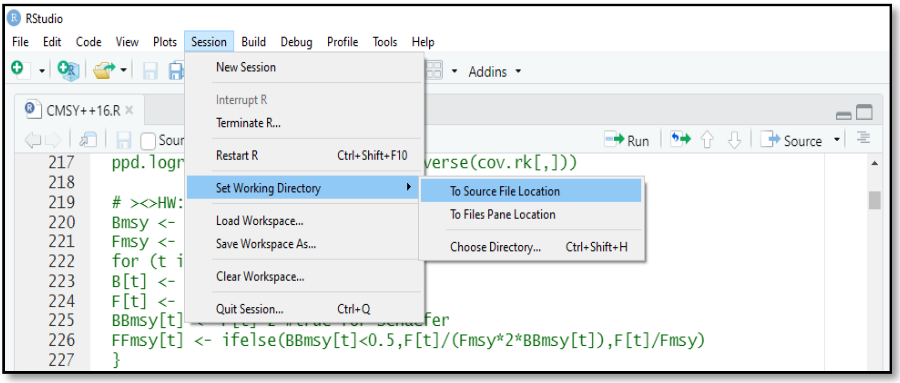
Click the 'File' in RStudio and then 'Open File...' (or simply CTRL+O). This will open up a browsing window to search and load the R-script file. Browse to the folder, where all the four files (i.e. CMSY++16.R, ffnn.bin, Train_Catch_9e.csv, Train_ID_9e.csv) are previously saved and load (open) only the R-script file, i.e., CMSY++16.R.





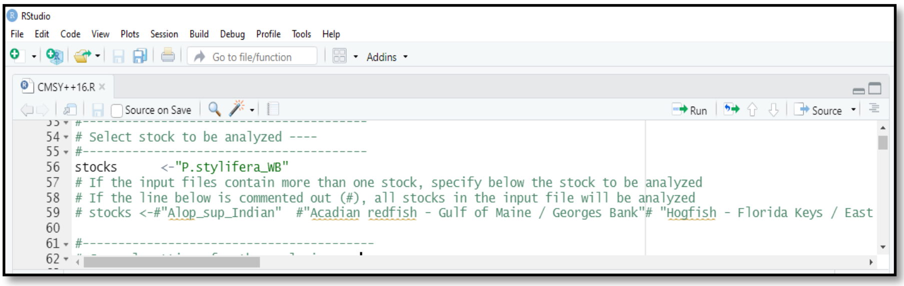
If it is a first time CMSY++ operation, then the system may ask to install few more dependent packages such as conicfit, gplots, neuralnet while loading the CMSY++16.R, which may be installed by clicking install in the warning message itself.

Set the working directory by clicking 'Session' in RStudio and then select "Set Working Directory" to "To Source File Location". Setting of working directory is a crucial step to facilitate the R-script (CMSY++16.R) find the remaining three files (i.e., ffnn.bin, Train_Catch_9e.csv, Train_ID_9e.csv) without which analysis can not be done. It also helps in storing the outputs from the analysis in the same working directory.



Define the stock to be analyzed in the opened R-script

To analyze any particular stock in the data go to the "Select stock to be analyzed" section of the code and enter the desired stock name (e.g., P.stylifera_WB) in line 56 (stocks <- "ple.27.7d") by replacing the example stock name (i.e., ple.27.7d). After the change, it should look like stocks <-"P.stylifera_WB" in line 56.

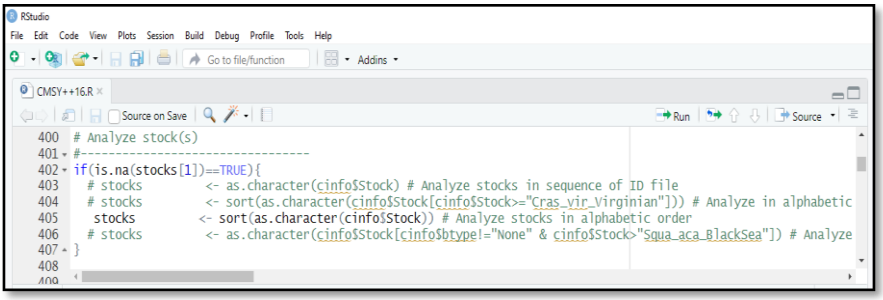


```
54 # Select stock to be analyzed ----
55 #----------------------------------------
56 stocks      <-"P.stylifera_WB"
57 # If the input files contain more than one stock, specify below the stock to be analyzed
58 # If the line below is commented out (#), all stocks in the input file will be analyzed
59 # stocks <-#"Alop_sup_Indian"  #"Acadian redfish - Gulf of Maine / Georges Bank"# "Hogfish - Florida Keys / East
60
61 #----------------------------------------
```

In the case where more stocks are there to be analysed (information is already there in Train_Catch_9e.csv, Train_ID_9e.csv), define the names of the stocks (same as the Train_Catch_9e.csv and Train_ID_9e.csv) one after the other in line 56 as follows:

stocks <-c("P.stylifera_WB", “P.monodon_WB” ,“P.semisulcatus_WB”)

The sequence of analysis, i.e., whether the stocks are going to be analysed in alphabetic order or in the sequence they appear in the ID file or by Region or subregion, can be controlled by uncommenting (removing # before stocks) in lines 403 of the “Analyze stocks” section, and below.



```
400 # Analyze stock(s)
401 #--------------------------------
402 if(is.na(stocks[1])==TRUE){
403   # stocks         <- as.character(cinfo$Stock) # Analyze stocks in sequence of ID file
404   # stocks         <- sort(as.character(cinfo$Stock[cinfo$Stock>="Cras_vir_Virginian"])) # Analyze in alphabetic
405    stocks        <- sort(as.character(cinfo$Stock)) # Analyze stocks in alphabetic order
406   # stocks         <- as.character(cinfo$Stock[cinfo$btype!="None" & cinfo$Stock>"Squa_aca_BlackSea"]) # Analyze
407 }
408
```

Define the output settings for the analysis

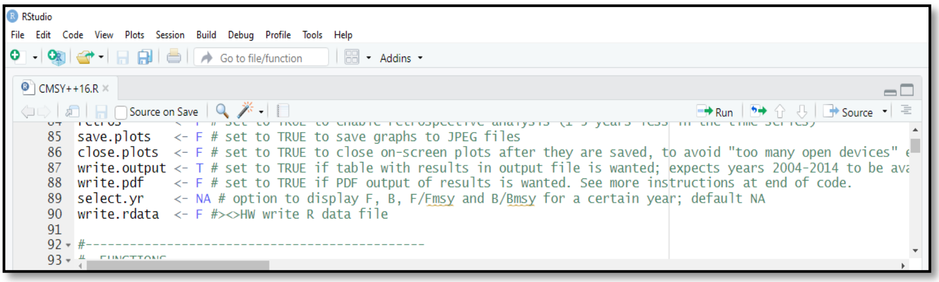
Several settings can be controlled in the “General settings for the analysis” to produce different outputs as follows:

Line 85: Set the graphical output “save.plots” as false ‘F’ (save.plots <- F) in the “General settings for the analysis” section to prevent the default graphical output savings in low resolution JPEG format.

Line 86: Set the graphical output “close.plots” as false ‘F’ (close.plots <- F) in the “General settings for the analysis” section to prevent the graphical window from disappearing, which then can be used to save the graphs manually in pdf format, which are better in resolution.

Line 88: Set the graphical output “write.pdf” to false ‘F’ (write.pdf <- F) in the “General settings for the analysis” section to suppress the errors that will be generated while saving the pdf files in the absence of "pdflatex" package. “pdflatex” package is not readily available in R and therefore, can not be installed using install.packages("pdflatex"). Therefore, the graphical output should be saved manually in the pdf format.

Line 87: Set the “write.output” to true ‘T’ (write.output <- T) in the “General settings for the analysis” section to save the analysis outputs in a tabular .csv format.



```
85 save.plots   <- F # set to TRUE to save graphs to JPEG files
86 close.plots  <- F # set to TRUE to close on-screen plots after they are saved, to avoid "too many open devices" e
87 write.output <- T # set to TRUE if table with results in output file is wanted; expects years 2004-2014 to be ava
88 write.pdf    <- F # set to TRUE if PDF output of results is wanted. See more instructions at end of code.
89 select.yr    <- NA # option to display F, B, F/Fmsy and B/Bmsy for a certain year; default NA
90 write.rdata  <- F #><>HW write R data file
91
92 #-------------------------------------------
93 #  FUNCTIONS
```

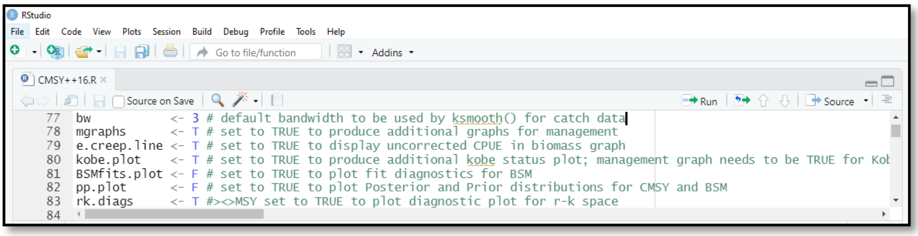
Line 78 (optional): Set the graphical output “mgraphs” to true ‘T’ (mgraphs <- T) in the “General settings for the analysis” section to produce additional graphs for management.

Line 80 (optional): Set the graphical output “kobe.plot” to true ‘T’ (kobe.plot <- T) in the “General settings for the analysis” section to produce an additional Kobe status plot.

Line 81 (optional): Set the graphical output “BSMfits.plot” to true ‘T’ (BSMfits.plot <- T) in the “General settings for the analysis” section to produce additional diagnostic plots for the BSM analysis.

Line 82 (optional): Set the graphical output “pp.plot” to true ‘T’ (pp.plot <- T) in the “General settings for the analysis” section to plot posterior and prior distributions for CMSY and BSM.

Line 83 (optional): Set the graphical output “rk.diags” to true ‘T’ (rk.diags <- T) in the “General settings for the analysis” section to plot diagnostic plot for r-k space.



```
77 bw           <- 3 # default bandwidth to be used by ksmooth() for catch data
78 mgraphs      <- T # set to TRUE to produce additional graphs for management
79 e.creep.line <- T # set to TRUE to display uncorrected CPUE in biomass graph
80 kobe.plot    <- T # set to TRUE to produce additional kobe status plot; management graph needs to be TRUE for Kob
81 BSMfits.plot <- F # set to TRUE to plot fit diagnostics for BSM
82 pp.plot      <- F # set to TRUE to plot Posterior and Prior distributions for CMSY and BSM
83 rk.diags     <- T #><>MSY set to TRUE to plot diagnostic plot for r-k space
84
```

Run CMSY++ codes

In RStudio, click on “Source” (shown in green colour box) or simply press Ctrl + Shift + S to execute the CMSY++ codes.

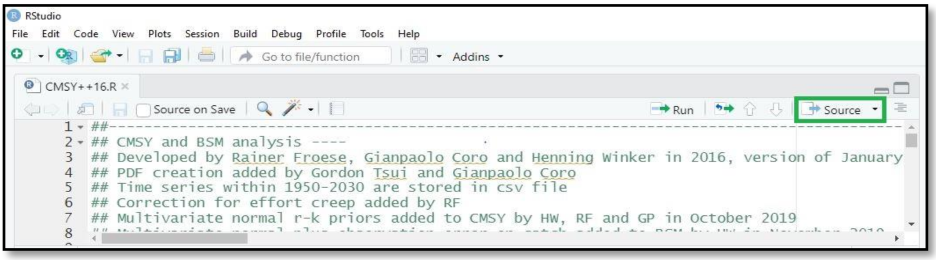
To do only the catch-based CMSY analysis

This requires only the time series catch information (i.e., populated ct column) in the Train_Catch_9e.csv. However, in the absence of CPUE or biomass information, it is essential to mention the btype column (the AF column) in the Train_ID_9e.csv as None. If additional information on the CPUE or biomass (i.e., populated bt column) is available in the Train_Catch_9e.csv, then suppress such information by mentioning the btype column (the AF column) in the Train_ID_9e.csv as None. This will trigger the analysis to follow only the CMSY approach suppressing the BSM approach in the absence of CPUE or biomass information.

To do both the catch-based CMSY analysis and catch and effort-based BSM analysis

This requires both the Catch (i.e., populated ct column) and CPUE or biomass (i.e., populated bt column) information in the Train_Catch_9e.csv. In the presence of CPUE or biomass information, it is essential to mention the btype column (the AF column) in the Train_ID_9e.csv as CPUE or biomass. This will trigger the analysis to follow both the catch-based MSY (CMSY) and Bayesian Schaefer surplus production model (BSM) approaches.

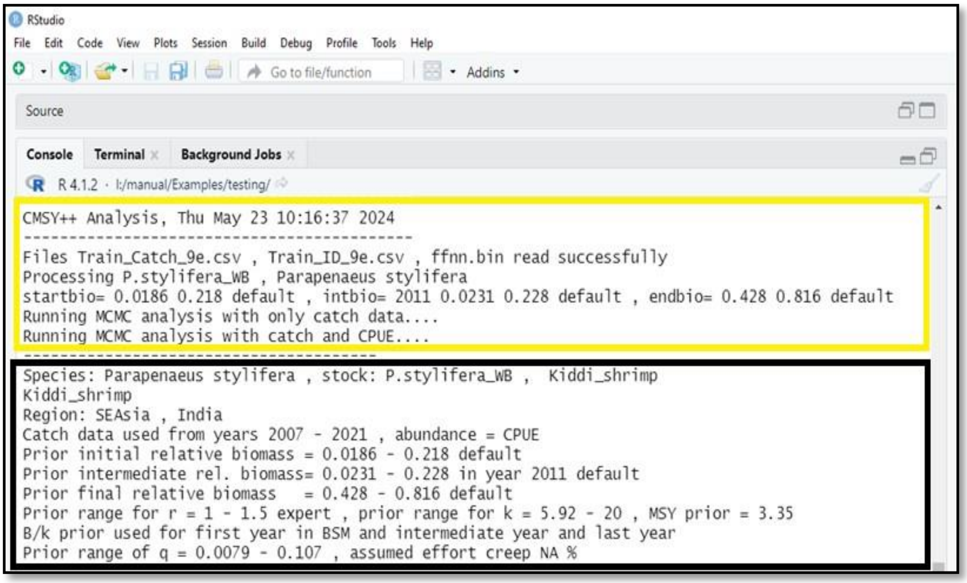
1. To get management results only from the CMSY analysis, mention the force.cmsy (AH column) as TRUE in the Train_ID_9e.csv.
2. To get management results only from the BSM analysis, mention the force.cmsy (AH column) as FALSE in the Train_ID_9e.csv.



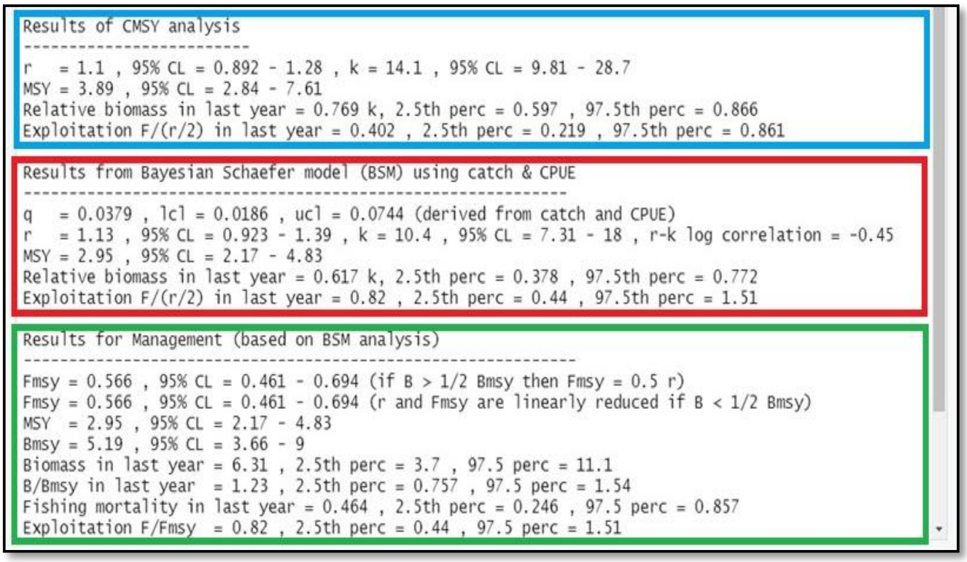
```
##---------------------------------------------------------------------------
## CMSY and BSM analysis ----
## Developed by Rainer Froese, Gianpaolo Coro and Henning Winker in 2016, version of January
## PDF creation added by Gordon Tsui and Gianpaolo Coro
## Time series within 1950-2030 are stored in csv file
## Correction for effort creep added by RF
## Multivariate normal r-k priors added to CMSY by HW, RF and GP in October 2019
## Multivariate normal plus observation error on catch added to BSM by HW in November 2019
```

3.2.10. CMSY++ text output

The result of the analysis is displayed after the successful completion of CMSY++ run as follows:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.1.2 · I:/manual/Examples/testing/

```
CMSY++ Analysis, Thu May 23 10:16:37 2024
------------------------------------------
Files Train_Catch_9e.csv , Train_ID_9e.csv , ffnn.bin read successfully
Processing P.stylifera_WB , Parapenaeus stylifera
startbio= 0.0186 0.218 default , intbio= 2011 0.0231 0.228 default , endbio= 0.428 0.816 default
Running MCMC analysis with only catch data....
Running MCMC analysis with catch and CPUE....
----------------------------------------
Species: Parapenaeus stylifera , stock: P.stylifera_WB ,  Kiddi_shrimp
Kiddi_shrimp
Region: SEAsia , India
Catch data used from years 2007 - 2021 , abundance = CPUE
Prior initial relative biomass = 0.0186 - 0.218 default
Prior intermediate rel. biomass= 0.0231 - 0.228 in year 2011 default
Prior final relative biomass   = 0.428 - 0.816 default
Prior range for r = 1 - 1.5 expert , prior range for k = 5.92 - 20 , MSY prior = 3.35
B/k prior used for first year in BSM and intermediate year and last year
Prior range of q = 0.0079 - 0.107 , assumed effort creep NA %
```



```
Results of CMSY analysis
------------------------
r   = 1.1 , 95% CL = 0.892 - 1.28 , k = 14.1 , 95% CL = 9.81 - 28.7
MSY = 3.89 , 95% CL = 2.84 - 7.61
Relative biomass in last year = 0.769 k, 2.5th perc = 0.597 , 97.5th perc = 0.866
Exploitation F/(r/2) in last year = 0.402 , 2.5th perc = 0.219 , 97.5th perc = 0.861

Results from Bayesian Schaefer model (BSM) using catch & CPUE
------------------------------------------------------------
q   = 0.0379 , lcl = 0.0186 , ucl = 0.0744 (derived from catch and CPUE)
r   = 1.13 , 95% CL = 0.923 - 1.39 , k = 10.4 , 95% CL = 7.31 - 18 , r-k log correlation = -0.45
MSY = 2.95 , 95% CL = 2.17 - 4.83
Relative biomass in last year = 0.617 k, 2.5th perc = 0.378 , 97.5th perc = 0.772
Exploitation F/(r/2) in last year = 0.82 , 2.5th perc = 0.44 , 97.5th perc = 1.51

Results for Management (based on BSM analysis)
-------------------------------------------------------------
Fmsy = 0.566 , 95% CL = 0.461 - 0.694 (if B > 1/2 Bmsy then Fmsy = 0.5 r)
Fmsy = 0.566 , 95% CL = 0.461 - 0.694 (r and Fmsy are linearly reduced if B < 1/2 Bmsy)
MSY  = 2.95 , 95% CL = 2.17 - 4.83
Bmsy = 5.19 , 95% CL = 3.66 - 9
Biomass in last year = 6.31 , 2.5th perc = 3.7 , 97.5 perc = 11.1
B/Bmsy in last year  = 1.23 , 2.5th perc = 0.757 , 97.5 perc = 1.54
Fishing mortality in last year = 0.464 , 2.5th perc = 0.246 , 97.5 perc = 0.857
Exploitation F/Fmsy  = 0.82 , 2.5th perc = 0.44 , 97.5 perc = 1.51
```

Yellow box: The contents inside the highlighted yellow box show the correct reading of Train_Catch_9e.csv, Train_ID_9e.csv and ffnn.bin files by the system.

Black box: The contents inside the highlighted black box show the priors used by the system for the analysis.

Blue box: The contents inside the highlighted blue box show the stock status from the CMSY analysis.

Red box: The contents inside the highlighted red box show the stock status from the BSM analysis.

Green box: The contents inside the highlighted green box show the management results from the BSM analysis.

Note: *The analysis can be forced to produce the management results from the CMSY analysis by forcing the CMSY as TRUE (force.cmsy = TRUE) in the AH column of Train_ID_9e.csv. The analysis can be forced to produce both the stock status and management results only from the CMSY analysis by suppressing the biomass information (btype = None) in the AF column of Train_ID_9e.csv. The BSM results will disappear from the above mentioned computational output.*

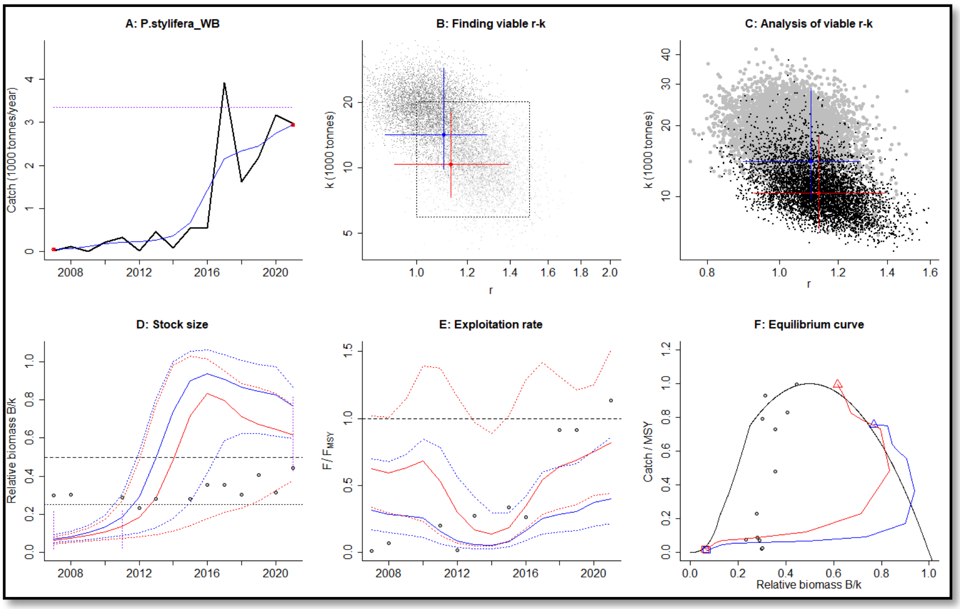
3.2.11. CMSY++ graphical outputs

The graphical result of the analysis is displayed after the successful completion of CMSY++ run as follows:

CMSY and BSM plots

Panel A: Black line shows the time series of catches. The blue curve shows the smoothed data and the red dots show the highest and lowest catch.

Panel B: The explored log r-k space. The rectangle shows the range of the r and k priors provided in the ID file. The point in the center of the blue cross is the most likely r-k pair, while horizontal and vertical error bars approximate 95% confidence limits.



Panel C: Zoomed view of Panel B

Panel D: The solid and dotted blue curve shows the median of the biomass trajectories and their confidence interval estimated by CMSY. The solid and dotted red curve shows the median of the biomass trajectories and their confidence interval estimated by BSM. Vertical purple lines show the prior biomass ranges, dotted if provided by the neural network and solid if set by the user.

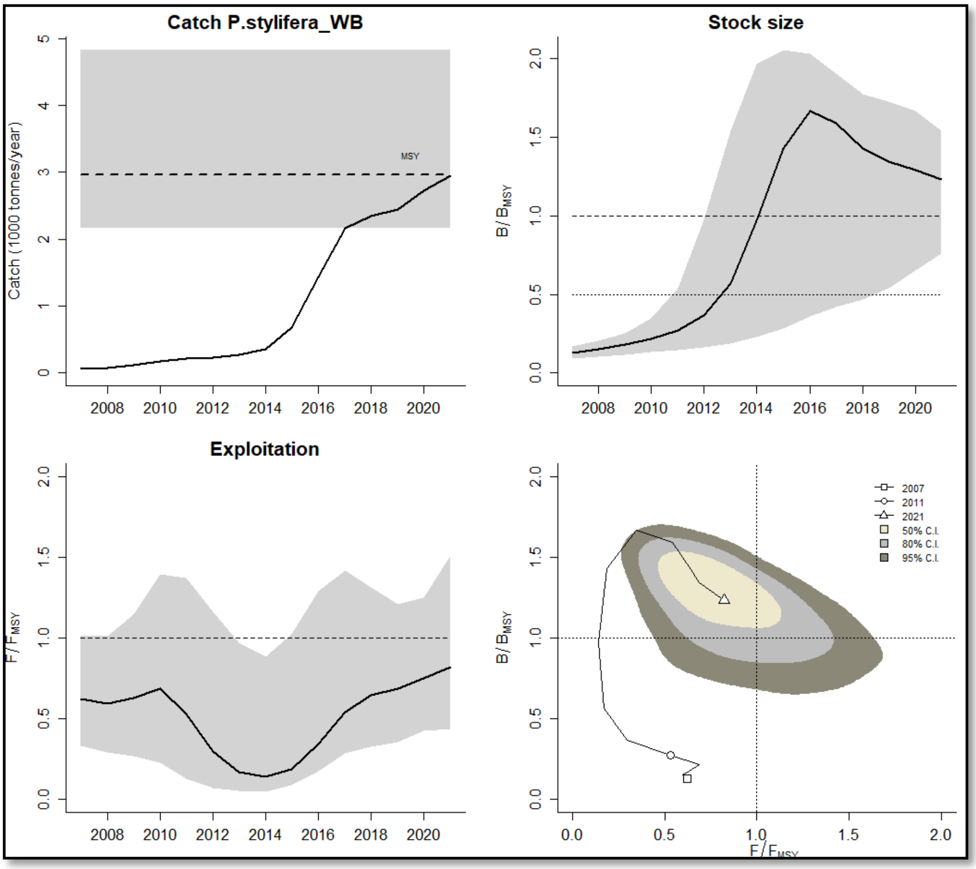
Panel E: The solid and dotted blue curve shows the median of the F/Fmsy trajectories and their confidence interval estimated by the CMSY. The solid and dotted red curve shows the median of the F/Fmsy trajectories and their confidence interval estimated by BSM.

Panel F: Panel F shows the Schaefer equilibrium curve of catch/MSY relative to B/k, indented at B/k < 0.25 to account for reduced recruitment at low stock sizes. The blue and red curves show the predictions by CMSY and BSM, from the first year (square) to the last years (triangle).

Note: *The analysis can be forced to produce the stock status and management results only from the CMSY analysis by suppressing the biomass information (btype = None) in the AF column of Train_ID_9e.csv. The red lines (BSM results) will disappear from the above graphical results.*

Management plots

Set the graphical output “**mgraphs**” to true ‘**T**’ (mgraphs <- T) in the “**General settings for the analysis**” (**Line 78**) section to produce additional graphs for management.



The upper left panel shows catches relative to MSY (dashed line) as estimated by CMSY, with an indication of 95% confidence limits in light grey.

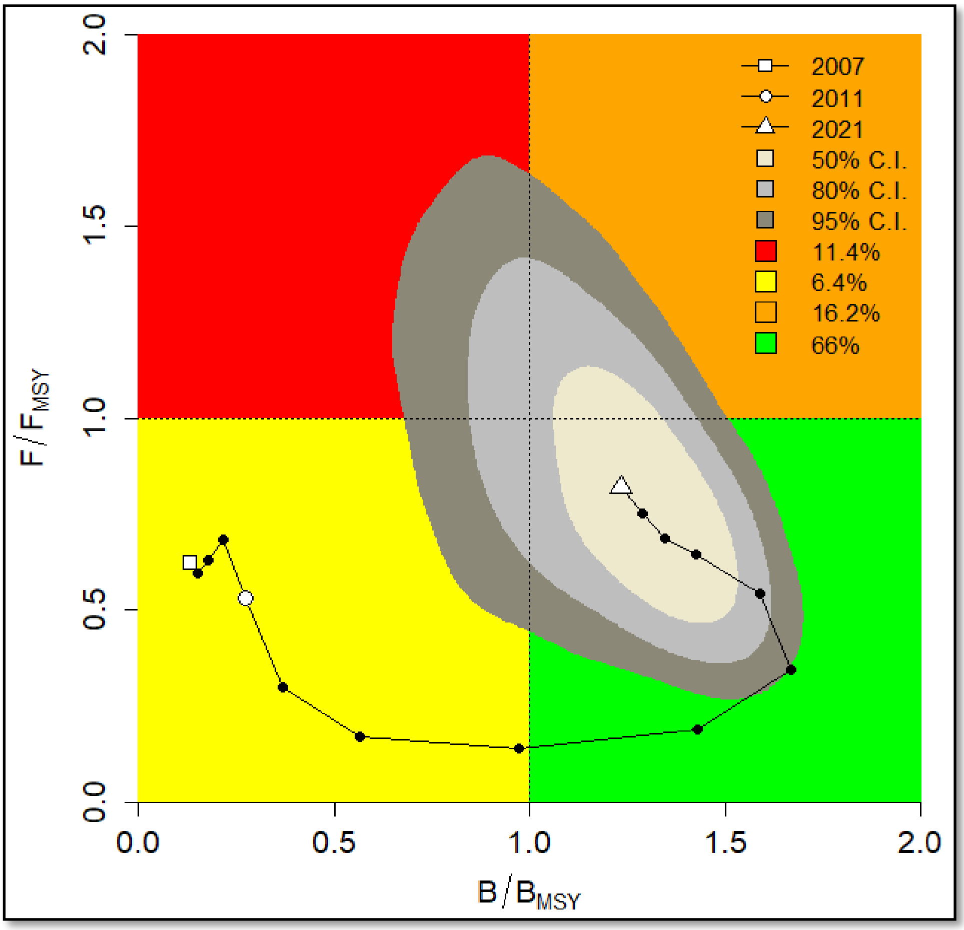
The upper right panel shows the time series of predicted total biomass relative to Bmsy (dashed line) and to the border of reduced recruitment (dotted line), with the light grey area showing uncertainty.

The lower left panel shows relative exploitation (F/Fmsy).

The lower-right panel is a flipped Kobe plot which shows the trajectory of relative stock size (B/Bmsy) as a function of fishing pressure (F/Fmsy). The “banana” shape around the assessment of the final year (triangle) shows uncertainty with yellow for 50%, grey for 80% and dark grey for 95% confidence levels.

Kobe plot

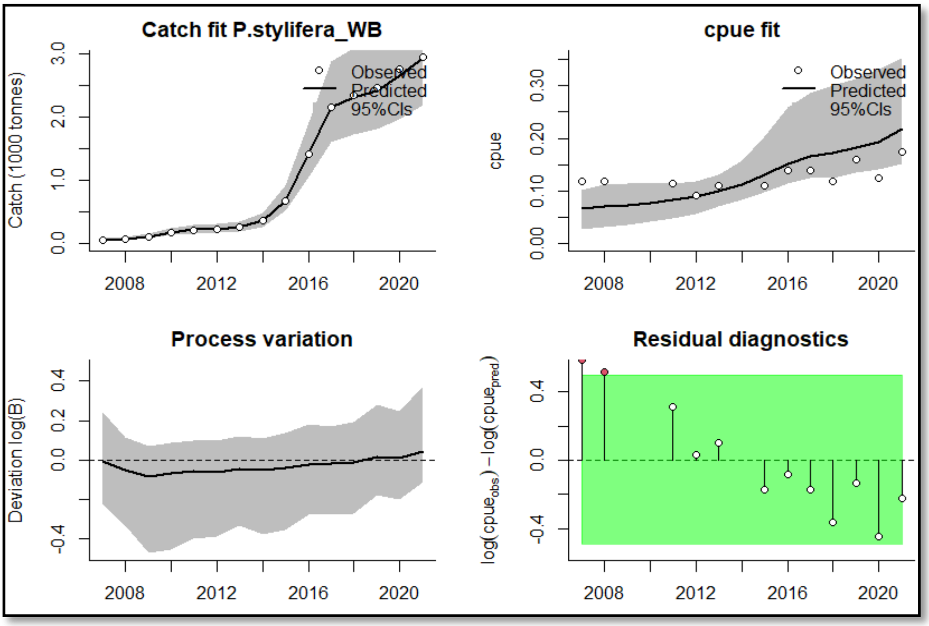
Additionally, set the graphical output "kobe.plot" to true 'T' (kobe.plot <- T) in the "General settings for the analysis" section (**Line 80**) to produce the Kobe status plot.



The Kobe plot represents the time series plotting of fishing pressure (F/F_{MSY}) on the Y-axis against the stock biomass status (B/B_{MSY}) on the X-axis. The plot is divided into four quadrants, i.e. (1) The orange area shows healthy stock sizes that are about to be depleted by overfishing, (2) The red area shows that the stock is overfished and is undergoing overfishing, with biomass levels being too low to produce maximum sustainable yields, (3) The yellow area shows reduced fishing pressure on stocks recovering from still too low biomass levels and (4) The green area is the target area for management, showing sustainable fishing pressure and healthy stock size capable of producing high yields close to MSY. The "banana" shape around the assessment of the final year (triangle) shows uncertainty with yellow for 50%, grey for 80% and dark grey for 95% confidence levels. The legend in the upper right graph also shows the probability of the last year falling into one of the colored areas, i.e., in this example there is a 66% probability that the stock is in the green area and a 11.4% probability that it is in the red area. Target would be, e.g., a higher than 75% probability that the stock is in the green area.

BSM diagnostic plots

The graphical output "BSMfits.plot" can be set to true 'T' (BSMfits.plot <- T) in the "General settings for the analysis" section (**Line 81**) to produce additional diagnostic plots for the BSM analysis.

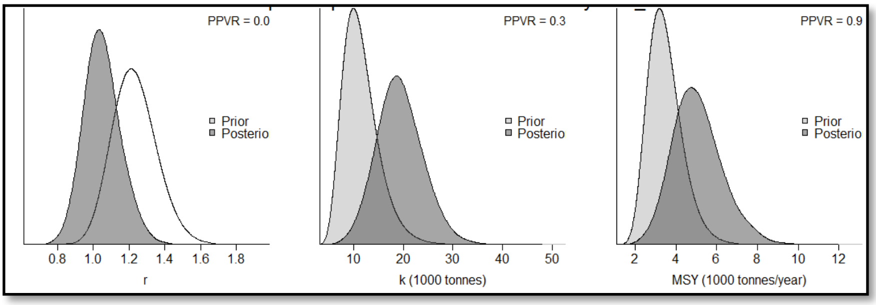


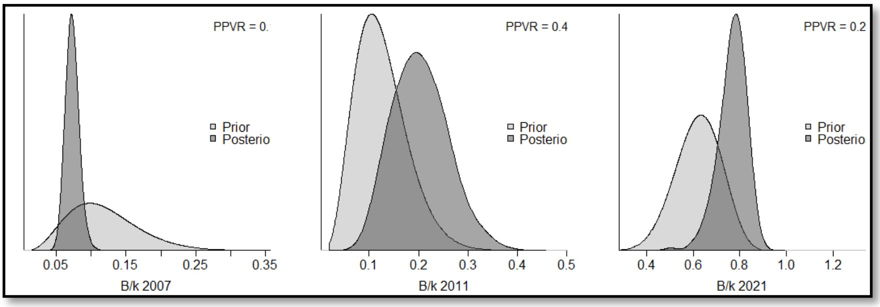
The upper left panel shows the fit represented by the median of predicted catch posterior, with 95% confidence limits (grey shaded area), compared to the observed catch (points). The upper right panel shows a similar graph for predicted versus observed CPUE. The lower left panel shows the deviation between deterministic expectation (surplus production minus catch) and the stochastic realization (after adding process error), where a strong deviation of the bold curve from the dashed line would show that changes in biomass diverge from the Schaefer model expectations due to, e.g., (1) strong environmental variation, (2) CPUE not properly describing the abundance or (3) the priors being mis-specified. The lower right panel shows an analysis of the log-CPUE residuals, which should preferably be randomly distributed.

Posterior and prior distributions plot

The graphical output "pp.plot" can be set to true 'T' (pp.plot <- T) in the "General settings for the analysis" section (**Line 82**) to plot the Posterior and Prior distributions for CMSY and BSM input and output variables. The graphs below show the comparison of prior and posterior densities (area under curves) for resilience or productivity (r), unexploited stock size (k), maximum sustainable yield (MSY), and relative stock size (B/k) at the beginning, the end, and an intermediate year of the available time series of catch data.

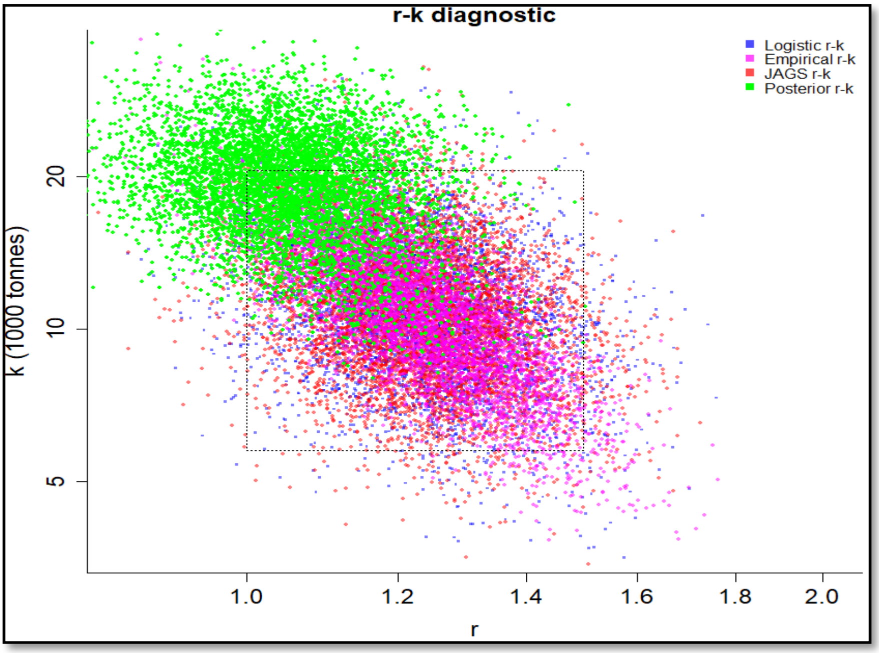
Scroll down to get the outputs from CMSY and BSM analysis. Lower the prior and posterior variance ratio (PPVR) better is the result.





r-k diagnostic plot

The graphical output "**rk.diags**" can be set to true 'T' (rk.diags <- T) in the "**General settings for the analysis**" section (**Line 83**) to plot the diagnostic plot for r-k.



The above graphs show a random distribution of r-k pairs (dots) generated from different approaches, i.e., (1) logistic approach (blue dots), (2) empirical approach (purple dots); (3) JAGS modeling approach (Orange dots); and (4) posterior distribution of r-k points as a result of the Bayesian modeling approach (green dots)

References:

Journal articles

Borema, L. K., & Gulland, J. A. (1973). Stock assessment of the Peruvian anchovy (*Engraulis ringens*) and management of the fishery. *Journal of the Fisheries Research Board of Canada, 30*, 2226–2235. https://doi.org/10.1139/f73-351

Cousido-Rocha, M., Pennino, M. G., Izquierdo, F., Paz, A., Lojo, D., Tifoura, A., Zanni, M. Y., & Cerviño, S. (2022). Surplus production models: A practical review of recent approaches. *Reviews in Fish Biology and Fisheries, 32*, 1085–1102. https://doi.org/10.1007/s11160-022-09731-w

Fox, W. W. (1970). An exponential surplus yield model for optimizing exploited fish populations. *Transactions of the American Fisheries Society, 99*(1), 80–88. https://doi.org/10.1577/1548-8659(1970)99<80:AESMFO>2.0.CO;2

Froese, R., Demirel, N., Coro, G., Kleisner, K. M., & Winker, H. (2017). Estimating fisheries reference points from catch and resilience. *Fish and Fisheries, 18*(3), 506–526. https://doi.org/10.1111/faf.12190

Kokkalis, A., Berg, C. W., Kapur, M. S., Winker, H., Jacobsen, N. S., Taylor, M. H., Ichinokawa, M., Miyagawa, M., Medeiros-Leal, W., Nielsen, J. R., & Mildenberger, T. K. (2024). Good practices for surplus production models. *Fisheries Research, 275*, Article 107010. https://doi.org/10.1016/j.fishres.2024.107010

Larkin, P. A. (1977). An epitaph for the concept of maximum sustainable yield. *Transactions of the American Fisheries Society, 106*(1), 1–11. https://doi.org/10.1577/1548-8659(1977)106<1:AEFTCO>2.0.CO;2

Pella, J. J., & Tomlinson, P. K. (1969). A generalized stock production model. *Bulletin of the Inter-American Tropical Tuna Commission, 13*, 419–496.

Pedersen, M. W., Berg, C. W., Thygesen, U. H., Nielsen, A., & Madsen, H. (2011). Estimation methods for nonlinear state-space models in ecology. *Ecological Modelling, 222*(8), 1394–1400. https://doi.org/10.1016/j.ecolmodel.2011.01.007

Prager, M. H. (1992). ASPIC: A surplus-production model incorporating covariates. *Collective Volume of Scientific Papers, ICCAT, 38*, 218–229. https://www.iccat.int/Documents/CVSP/CV038_1992/CV038000218.pdf

Prager, M. H. (1994). A suite of extensions to a nonequilibrium surplus-production model. *Fishery Bulletin, 92*(2), 374–389. www.mhprager.com/reports/Prager_1994_ASPIC_FishBull.pdf

Prager, M. H., Goodyear, C. P., & Scott, G. P. (1996). Application of a surplus production model to a swordfish-like simulated stock with time-changing gear selectivity. *Transactions of the American Fisheries Society, 125*(5), 729–740.

https://doi.org/10.1577/1548-8659(1996)125<0729:AOASPM>2.3.CO;2

Williams, E., & Prager, M. H. (2002). Comparison of equilibrium and nonequilibrium estimators for the generalized production model. *Canadian Journal of Fisheries and Aquatic Sciences, 59*(9), 1533–1552. https://doi.org/10.1139/f02-123

Winker, H., Carvalho, F., & Kapur, M. (2018). JABBA: Just Another Bayesian Biomass Assessment. *Fisheries Research, 204*, 275–288. https://doi.org/10.1016/j.fishres.2018.03.010

Books

Froese, R., Demirel, N., Coro, G., & Winker, H. (2021). *User guide for CMSY++* (17 pp.). GEOMAR, Kiel, Germany. https://oceanrep.geomar.de/id/eprint/52147

Hilborn, R., & Walters, C. J. (1992). *Quantitative fisheries stock assessment: Choice, dynamics, and uncertainty*. Springer New York, NY, p.570. https://doi.org/10.1007/978-1-4615-3598-0

Ricker, W. E. (1975). *Computation and interpretation of biological statistics of fish populations*. Bulletin 191, Fisheries Research Board of Canada, p. 382. https://waves-vagues.dfo-mpo.gc.ca/Library/1485.pdf

Schaefer, M. B. (1954). Some aspects of the dynamics of populations important to the management of marine fisheries. *Bulletin of the Inter-American Tropical Tuna Commission, 1*, 25–56. https://www.iattc.org/GetAttachment/62d510ee-13d0-40f2-847b-0fde415476b8/Vol-1-No-2-1954-SCHAEFER,-MILNER-B-_Some-aspects-of-the-dynamics-of-populations-important-to-the-management-of-the-commercial-marine-fisheries.pdf

3.3. Stock Reduction Analysis (SRA)

Introduction

These are the simplest stock assessment tools used under data poor situations when only catch data is available and therefore, the methods are also known as catch only methods (COMs). Unlike the surplus production model (SPM) which use both catch and an index of abundance (CPUE) to derive maximum intrinsic rate of population growth (r) and carrying capacity (k) which are further used to derive maximum sustainable yield (MSY) and related fisheries management reference points, the stock reduction analysis (SRA) in its basic form use alternative approach to derive the fisheries management reference points from only the catch data. Instead of using catch and CPUE data to directly derive r and k for the estimation of fisheries management points, the method use computational simulation to prepare biomass trajectories (through a biomass dynamic model) with a probable range of r and k pair values which will produce the observed catch while confirming with the assumed initial and final biomass levels (more precisely relative B/k or depletion levels) without exceeding the carrying capacity (k) or collapsing the stock.

The simplest deterministic method for the SRA was initially developed by Kimura and Tagart (1982), which was further refined by Kimura et al. (1984). The SRA of Kimura and Tagart (1982) assumed a simple biomass production function with constant recruitment. However, the simple SRA formulation proposed by Kimura and Tagart (1982) did not give any consideration for the growth in biomass and variability in recruitment. Later, Kimura et al. (1984) improved the SRA by including Deriso's delay difference model in the biomass production function to account for growth and inter-annual change in the biomass. He also included variability in recruitment either by incorporating a stock recruitment relationship (Cushing recruitment) or by using an exogenous variable (proportional recruitment) in the biomass production functions. The variables of the models M, B_1, P, R and Fs of simple SRA by Kimura and Tagart (1982) and B_1, R, P, Fs, M, ρ, r and p_i by Kimura et al. (1984) are solved by conditioning (fixing) certain parameters using an interactive SRA plot and then iteratively solving the remaining parameters to arrive at an acceptable solution for the parameters that will produce the observed catch. Once the parameters are resolved, the exploitation rates (U_i) and the biomass (B_i) for the corresponding years are determined using the standard formula. Later, Walter (2006) developed a stochastic approach for SRA. The working principle and method of implementation of commonly used SRA approaches, such as Depletion Corrected Average Catch (DCAC), Depletion Based Stock Reduction Analysis (DB-SRA) and sraplus, have been described in this section.

3.3.1. Depletion corrected average catch (DCAC)

Ideally, a sustainable yield can be inferred if a fishery has shown a long period of steady catch without a decrease in the underlying resource abundance (biomass). In this case, the long-term average annual catch might serve as an estimate of sustainable yield. However, it is rare for a fishery to maintain a constant abundance, as exploitation often leads to an initial depletion of the stock. For new or recently developed fisheries, part of the catch often comes from this onetime decline in underlying biomass rather than from sustainable production. Including this portion of the catch, that will never be used for the sustainable production cycle, during averaging procedure might overestimate the sustainable yield. Depletion-corrected average catch (DCAC) addresses this issue.

DCAC is based on the potential-yield formula of Alverson and Pereyra (1969) and Gulland (1970). By approximating B_{MSY} = 0.5 × B_0 and F_{MSY} = M, the potential yield can be expressed as:

$$Y_{pot} = \mathrm{B_{MSY}} \times \mathrm{F_{MSY}} = 0.5 \times \mathrm{B_0} \times \mathrm{M}$$

This maximum yield (MSY) can be considered as an onetime harvest (windfall harvest or W) that deplete the virgin stock biomass (B_0) to half of the B_0 (0.5 ×B_0) which can be expressed as:

$$W = 0.5 \times \mathrm{B_0}$$

After windfall reduction in biomass, Y_{pot} can be considered a tentatively sustainable annual yield. Under the potential-yield assumptions, the ratio of the onetime windfall yield to the sustainable yield can be calculated as:

$$\frac{W}{Y_{pot}} = \frac{0.5 \times \mathrm{B_0}}{0.5 \times \mathrm{B_0} \times \mathrm{M}} = \frac{1}{\mathrm{M}}$$

W/Y_{pot} expresses the magnitude of the windfall harvest relative to a single year of potential yield. For example, if M is 0.1 per year, the estimated windfall harvest is 10 times the value of estimated annual sustainable yield.

However, these above-mentioned generalized empirical relationships are not very accurate or universal. Most fishery stock–recruitment relationships (SRRs) show that the B_{MSY} of fish is less than the generally accepted 0.5 ×B_0, and 0.4 × B_0 has been proposed as a more realistic proxy for B_{MSY} (Clark, 1991; NMFS, 1998; Restrepo et al., 1998). Similarly, fishery experience suggests that the F_{MSY} may not be exactly the same as M and most often a time requires a correction factor (c) so that the F_{MSY} = cM. Now using these revised B_{MSY} = 0.4 × B_0 and F_{MSY} = cM, the potential yield can be expressed as:

$$Y_{pot} = \mathrm{B_{MSY}} \times \mathrm{F_{MSY}} = 0.4 \times \mathrm{B_0} \times \mathrm{cM}$$

Similarly, the windfall harvest can be expressed as the relative reduction in vulnerable stock abundance from the first year (FYR) to the last year (LYR) of the catch time-series, i.e. where W= B_{FYR} - B_{LYR}. In data poor condition where biomass has not been estimated, it is still possible to assume relative decline in abundance, Δ, which can be estimated as:

$$\Delta = \frac{B_{FYR} - B_{LYR}}{B_0}$$

Now multiplying B0 at both side the windfall harvest can be estimated as:

$$\Delta \times B_0 = \frac{B_{FYR} - B_{LYR}}{B_0} \times B_0 = B_{FYR} - B_{LYR} = W = \Delta \times B_0$$

Now, using the preceding equations, the general windfall ratio can be expressed as:

$$\frac{W}{Y_{pot}} = \frac{\Delta \times \mathrm{B_0}}{0.4 \times \mathrm{B_0} \times \mathrm{cM}} = \frac{\Delta}{0.4 \times \mathrm{cM}}$$

This generalized equation for the windfall ratio forms the basis for a depletion corrected average catch method. It is assumed that, on average, each year produces one unit of annual sustainable yield, which produces a catch that consists of a portion derived from sustainable annual production, whereas the remaining portion comes from a onetime windfall harvest. For a catch (C) series of length n = LYR - FYR + 1, the total cumulative catch (∑C) comprises n years of sustainable production, plus a windfall equivalent to W/Y_{pot} years of potential yield. The DCAC provides an estimate of the yield that could have been sustained (Y_{sust}) during that period as:

$$Y_{sust} = \frac{\sum C}{n + \frac{W}{Y_{pot}}} = \frac{\sum C}{n + \frac{\Delta}{0.4 \times \mathrm{cM}}}$$

Note: *if there has been no underlying change in abundance, then Δ becomes zero, which makes W/Y_{pot} = 0, turning the above equation a simple averaging for a multiyear catch. On the contrary, if the abundance has increases than the Δ and so does the W/Y_{pot} become a negative number, which will increase the estimated sustainable yield larger than the historical average catch.*

The DCAC uses reliable cumulative annual catch or individual year's catch for many years. It also assumes that the natural mortality rate (M) is not greater than 0.2 yr^{-1} as at a value above 0.2 yr^{-1} the effect of depletion correction becomes negligible. Assuming a log-normal distribution of M, a CV of 0.5 could be used as a minimal default value. The correction factor (c) while estimating the annual fishing mortality rate (F) from M, should not be higher than 1 and therefore, c=1 should be used as a target or upper limit. Walters and Martell (2004) suggest that the coefficient c is commonly 0.8, but may be 0.6 or less for vulnerable stocks. A standard error of 0.2 is suggested as the default estimate of precision. In a data-poor situation where the index of abundance is not available, it is difficult to estimate the relative reduction in biomass over the duration of a catch series. An estimate of relative depletion (Δ) can be got by questioning the experienced fishers and from the expert opinion. When no information is available, a Δ of 0.5 with a standard error of 0.15 is recommended for DCAC. The DCAC is implemented through a Monte Carlo exploration technique for the estimation parameters and their confidence intervals.

DCAC: R implementation

The DCAC is implemented through the '**DLMtool**' using R.

https://dlmtool.github.io/DLMtool/reference/DCAC.html

https://search.r-project.org/CRAN/refmans/DLMtool/html/DCAC.html

https://www.rdocumentation.org/packages/DLMtool/versions/3.1/topics/DCAC

To download the executable file for R independent installation:

https://noaa-fisheries-integrated-toolbox.github.io/DCAC

3.3.2. Depletion based stock reduction analysis (DB-SRA)

The method was developed by Dick and MacCall (2011) by combining stochastic SRA developed by Walter et al. (2006) with depletion corrected average catch (DCAC) of MacCall (2009). It uses a delay difference model to assess the current biomass (B_t) from the previous year biomass (B_{t-1}), catch (C_{t-1}) and latent annual production of parental biomass [$P(B_{t-a})$] which can be expressed as:

$$B_t = B_{t-1} + P(B_{t-a}) - C_{t-1}$$

It uses a Pella–Tomlinson–Fletcher (PTF) production model, a reparameterized version of Pella and Tomlinson SPM (1969) by Fletcher (1978) to calculate the latent annual production.

$$P = gm \times \left(\frac{B_{t-a}}{k}\right) - gm \times \left(\frac{B_{t-a}}{k}\right)^n$$

Where $g = n^{n/(n-1)}/(n\text{-}1)$, m = MSY and k = unfished biomass. The exponent n ($n > 0$) determines the skewness of the production function. The production function becomes symmetric like Schaefer SPM when n = 2 and right skewed like fox SPM, when n approaches 1. The production function becomes left skewed when $n > 2$. The ratio of biomass required to produce MSY to virgin stock biomass (B_{msy}/k) is expressed as B_{mnpl} in DB-SRA and is determined by the exponent n. If $n \neq 1$, then $B_{mnpl} = n^{1/(1-n)}$. If n = 1, and $B_{mnpl} = \exp(-1)$.

The production function is almost equivalent to the Beverton and Holt Stock Recruitment Relation (BHSRR) driven latent production function used for data-rich fisheries while addressing the limitations of the latter. The BHSRR restricts the peak latent productivity at B_{mnpl} (which is B_{msy}/k) < 0.5, whereas the PTF allows the B_{mnpl} to take any value between 0 and 1 (0 < B_{mnpl} < 1). Nevertheless, the PTF predicts unrealistically high productivity at low biomass in the case of highly skewed production curves, especially when B_{mnpl} = exp (-1). High skewness is often encountered under typical high values of BHSRR steepness (h > 0.5, where 'h' is Mace–Doonan steepness, the ratio of recruitment at B = 0.2K to recruitment at B = K; Punt et al., 2008). The issue is solved by the use of a modified hybrid Schaefer PTF model (different from the hybrid Schaefer PTF model by McAllister et al. (2000) which under-estimate productivity at low biomass) which provides a latent production function that has properties similar to the BHSRR while allowing full flexibility in specifying B_{mnpl}. This function has the form of a PTF production model for abundances (B_{t-a}) above a join-point (B_{join}) and has the form of a Schaefer model for abundances below B_{join}. The value of varies from 0 to B_{msy} (0< B_{join}< B_{msy}) and can be controlled to produce a good approximation of the BHSRR model. Following set of linear rules are used to define B_{join} during simulation:

if $B_{mnpl} < 0.3$, $B_{join}/K = 0.5\ B_{mnpl}$;

if $0.3 < B_{mnpl} < 0.5$, $B_{join}/K = 0.75\ B_{mnpl} - 0.075$;

if $B_{mnpl} > 0.5$, use PTF model for all Biomass.

The method requires time-series data on annual catches, an approximation on natural mortality rate (M) and age at maturity (a). The production function is specified based on general fishery knowledge of the relative location of maximum productivity (B_{mnpl}) and the relationship of F_{MSY} to the M. This helps the model to estimate the unfished biomass for a given a depletion level (B_t/k) near the end of the time series. The method uses historic catch data, believing that the catch data is reliable and error free. The catch data is subsequently depletion corrected and used in the analysis. The method uses a log-normal distribution assumption for M and recommends a log scale standard deviation of 0.4 as a default value to draw priors for the analysis. The method also recommends a log-normal distribution assumption for F_{MSY} and the ratio of F_{MSY} to M. The method recommends a mean F_{MSY}/M of 0.8 and a log-scale standard deviation of 0.2 for all species to draw priors. The maximum productivity (B_{mnpl}) which is nothing but the biomass that produces maximum surplus production (MSY) relative to the unfished biomass (B_{MSY}/K) is assumed to follow a bounded beta distribution. The relative location of B_{mnpl} on the production curve depends on the productivity of the species under investigation. However, the method recommends a mean (E) of 0.4 and a standard deviation of 0.05 on the untransformed scale (B_{mnpl} range = 0.31-0.49) to draw B_{mnpl} priors for most of the fish. The relative depletion (B_t/k) which is the final year biomass in relation to the unfished biomass, is also assumed to follow a bounded beta distribution with a mean (E) of 0.4 and a standard deviation of 0.1 if no prior information is available. The experience fishers and the experts can be interviewed to get a more reasonable prior for the relative depletion. Finally, the Monte Carlo exploration technique is used to incorporate the uncertainty associated with the above-mentioned model parameters and to calculate the management reference points and their probability distribution.

DB-SRA: R implementation

The DB-SRA is implemented through the '**DLMtool**' using R.

https://dlmtool.github.io/DLMtool/reference/DBSRA.html

https://rdrr.io/cran/fishmethods/man/dbsra.html

https://www.rdocumentation.org/packages/DLMtool/versions/5.0/topics/DBSRA

https://search.r-project.org/CRAN/refmans/DLMtool/html/DBSRA.html

The section presents a detailed, step-by-step guide for implementing sraplus, offering additional flexibility to apply methods like the Catch-Only Method (COM), such as SRA, under data-poor conditions, and Catch-and-Effort-based methods, such as a Bayesian SPM, under data-moderate conditions.

3.3.3. Stock Reduction Analysis Plus (sraplus)

The sraplus has been developed by Ovando et al. (2021) using the stochastic stock reduction analysis (SRA) earlier developed by Kimura et al. (1984) and Walters et al. (2006). It allows users to combine a biomass dynamics model with a variety of data sources (e.g. catch data, priors on recent stock status or an index of abundance, CPUE) to derive information on the status of a fishery. Under data-poor situation, sraplus works like a catch only method (COM) and implements stochastic SRA (Walters et al., 2006). When more information such as biomass or index of abundance (CPUE) is available, the sraplus go beyond the conventional SRA type analysis and try to fit to the index of abundance using Hamiltonian Monte Carlo with the No-U-Turn sampler (Hoffman & Gelman, 2011).

The sraplus uses a conditional modified implementation of Pella–Tomlinson generalized SPM (Pella & Tomlinson, 1969) constructed in the manner of Winker et al. (2018) to address the unrealistic high productivity issue at very low biomass.

The Pella–Tomlinson generalized SPM is used when B_t is more than the quarter of k (B_t>0.25 k)

$$B_{t+1} = \left(B_t + \frac{r}{m-1} \times B_t \times \left(1 - \left(\frac{B_t}{k}\right)^{m-1}\right) - C_t\right) p_t$$

The modified Pella–Tomlinson generalized SPM containing an additional conditional multiplier of B_t/(0.25×k) is used when B_t drops below the quarter of k (B_t<0.25 k).

$$B_{t+1} = \left(B_t + \frac{\boldsymbol{B_t}}{\boldsymbol{0.25 \times k}} \times \frac{r}{m-1} \times B_t \times \left(1 - \left(\frac{B_t}{k}\right)^{m-1}\right) - C_t\right) p_t$$

The sraplus uses the above Pella–Tomlinson generalized SPM when the B_t is less than a quarter of the virgin biomass (k), i.e., B_t < 0.25 k. The modified SPM includes an additional conditional multiplier **B_t/(0.25×k).** This multiplier equals to 1 when B_t/K=0.25, but when B_t/k drops below 0.25, it linearly reduces recruitment to zero as biomass approaches zero. This effectively emulates a 'hockey stick' recruitment function, similar to CMSY++ and JABBA. The Pella and Tomlinson SPM (1969) generalized SPM includes an additional shape parameter (m, sometime expressed as p which is equal to m^{-1}), which controls the shape or skewness of the production function, allowing the model to reach maximum production (or MSY) at any biomass level below carrying capacity. As m approaches 2, the Pella and Tomlinson SPM resembles the Schaefer SPM. When m is close to 1 the model behaves like the Fox SPM, producing a right-skewed, asymmetric production curve shifting the peak productivity to left. Conversely, with a 'm' value of more than 2, the model generates a left-skewed, asymmetric production curve, shifting the peak productivity to the right. It is difficult to estimate m value using the SPM and therefore, it is fixed following the established ratio of B_{MSY}/K for fish taxa developed by Thorson et al. (2012). Thorson et al. (2012) have shown that the m value varies from 0.599 in clupeiformes to 1.97 in scorpaeniformes, with a mean of 1.478 corresponding to the B_{MSY}/K of 0.404. The process error (p_t) is assumed to be log-normally distributed (Walters et al., 2006) which can be expressed as:

$$\log(p_i) \sim N\left(-\frac{\sigma_{proc}^2}{2}, \sigma_{proc}\right)$$

When only catch data is available (data-poor condition), sraplus performs a stochastic stock reduction analysis following the approach of Walters et al. (2006). The SRA approach uses a prior ranges for productivity indicators r (prior distribution drawn from Fishlife, Thorson (2020)), k and initial and final year relative biomass levels (B/K) to estimate "viable" combination of r-k pairs. This is achieved through a Markov Chain Monte Carlo (MCMC) bootstrap approach, simulating biomass trajectories with the above mentioned Pella and Tomlinson SPM that can produce the catch consistent with observed catch over time without collapsing the stock, exceeding the carrying capacity, or resulting in a final year depletion (B_t/K) outside the bounds of the supplied priors. These viable parameters are used to calculate management reference points, such as MSY, B/B_{MSY}, F/F_{MSY}, etc.

However, under data-moderate conditions, when time-series data on catch is available along with biomass or an index of abundance (e.g., CPUE), the model besides the above-mentioned biomass estimation, tries to fit the observed biomass or index of abundance (CPUE) to the simulated biomass. In the presence of effort data, instead of using the conventional CPUE which represent catch per unit effort, the approach uses a catch per effective harvest rate, which is expressed as:

$$CPUE_t = \frac{C_t}{1 - exp^{-F}}$$

Where F_t is the fishing mortality, which is calculated from catchability coefficient (q_t) and effort (E_t) using the formula, $F_t = q_t \times E_t$. The catchability coefficient is calculated from the effort creep (τ, which is 2.6% as a default in sraplus) using the formula, $q_t = q_{t-1} \times (1+\tau)$

Finally, the observed $CPUE_t$ is fitted to the simulated biomass (or calculated CPUE from simulate biomass including observation error) to get the likelihood of obtaining the observed $CPUE_t$ under different combination of input parameters for the production process which is further used to derive the posterior distribution of input parameters.

$$log\ (CPUE_t) = N(B_t, \sigma_{obs}\ )$$

These posterior distributions of parameters are subsequently used to derive the management reference points (MSY, B_{MSY}, F_{MSY}, etc.).

sraplus: R Implementation

3.3.4. Requirements for sraplus

Updating R

The sraplus requires R version 4.0 or above. Check the version of R installed in the system, using the following code: R.version.string

Download/upgrade the R, if the version of R is lower than 4.0. For more details, refer to '1.4.Updating R'.

Updating Rtools

Check the version of **Rtools** installed in the system, using '**pkgbuild**' R package. Install the 'pkgbuild' directly using the following code:

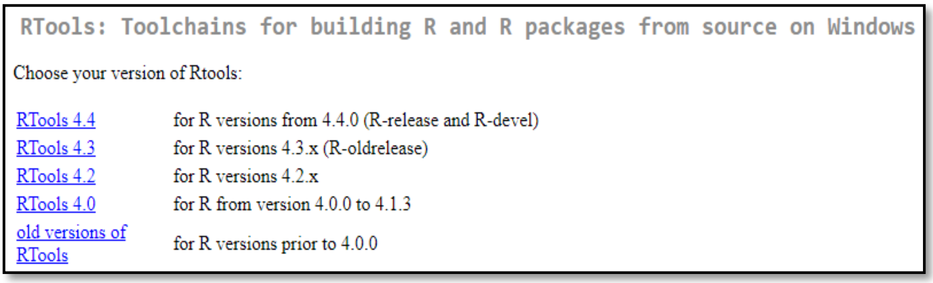
```
install.packages("pkgbuild")
```

If it does not install 'pkgbuild', try installation through devtools:

```
install.packages("devtools")
devtools::install_github("r-lib/pkgbuild")
pkgbuild::find_rtools(debug = TRUE)
```

If the Rtools are obsolete, then download and install the correct version of Rtools depending on the version of R installed in the system. Windows installation pack for the installation of Rtools for the correct version of R can be found in the below link:

https://cran.r-project.org/bin/windows/Rtools/



RTools: Toolchains for building R and R packages from source on Windows

Choose your version of Rtools:

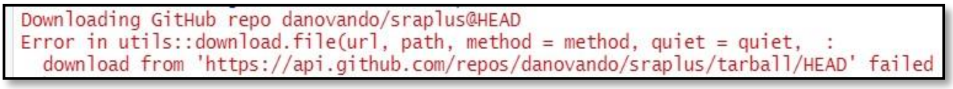
RTools 4.4	for R versions from 4.4.0 (R-release and R-devel)
RTools 4.3	for R versions 4.3.x (R-oldrelease)
RTools 4.2	for R versions 4.2.x
RTools 4.0	for R from version 4.0.0 to 4.1.3
old versions of RTools	for R versions prior to 4.0.0

Installing sraplus

To install the sraplus package from the github, use the following code:

```
install.packages("remotes")
remotes::install_github("danovando/sraplus")
```

If the package is not installed and the following message pop-up, then retry the package installation by increasing the connection timeout as follows:



```
Downloading GitHub repo danovando/sraplus@HEAD
Error in utils::download.file(url, path, method = method, quiet = quiet,  :
  download from 'https://api.github.com/repos/danovando/sraplus/tarball/HEAD' failed
```

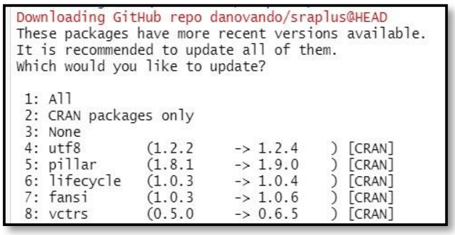
```
options(timeout=500)
```

library(remotes)

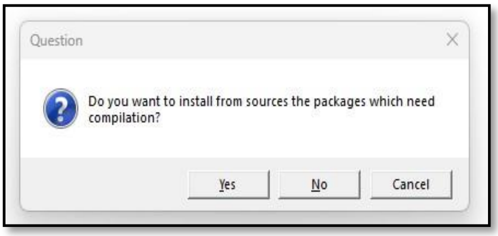
install_github("danovando/sraplus")

Updating dependent R packages

During sraplus installation, the system will ask to update and install many other dependent R packages required for the installation of the sraplus. Give permission to update all the dependent packages (Press 1 and enter key). Press 'yes' to install the packages from the source which need compilation.



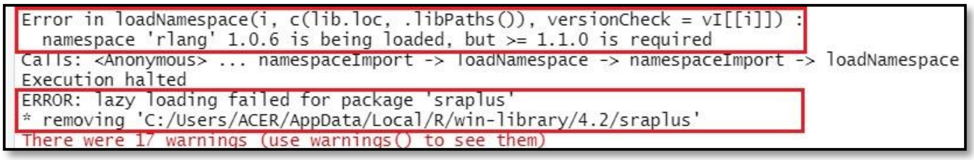
```
Downloading GitHub repo danovando/sraplus@HEAD
These packages have more recent versions available.
It is recommended to update all of them.
Which would you like to update?

 1: All
 2: CRAN packages only
 3: None
 4: utf8      (1.2.2    -> 1.2.4    ) [CRAN]
 5: pillar    (1.8.1    -> 1.9.0    ) [CRAN]
 6: lifecycle (1.0.3    -> 1.0.4    ) [CRAN]
 7: fansi     (1.0.3    -> 1.0.6    ) [CRAN]
 8: vctrs     (0.5.0    -> 0.6.5    ) [CRAN]
```



Do you want to install from sources the packages which need compilation?

Troubleshooting sraplus installation failure

If sraplus installation fails with the following warning, then check and update '**rlang**' and retry installation. If the rlang version is very old (outdated), then **lazy loading failed for package 'sraplus'** warning message is generated during sraplus installation, leading to failure.



```
Error in loadNamespace(i, c(lib.loc, .libPaths()), versionCheck = vI[[i]]) :
  namespace 'rlang' 1.0.6 is being loaded, but >= 1.1.0 is required
Calls: <Anonymous> ... namespaceImport -> loadNamespace -> namespaceImport -> loadNamespace
Execution halted
ERROR: lazy loading failed for package 'sraplus'
* removing 'C:/Users/ACER/AppData/Local/R/win-library/4.2/sraplus'
There were 17 warnings (use warnings() to see them)
```

Check the version of **rlang** installed in the system, using following code: packageVersion("rlang")

Update/install the latest **rlang** for the R using the following code:

install.packages("rlang")

If the above code does not install and update **rlang**, attempt installation through **devtools:**

devtools::install_github("r-lib/rlang")

After updating rlang, retry sraplus installation using the following codes:

options(timeout=500)

library(remotes)

install_github("danovando/sraplus")

If the installation fails even after updating rlang, then look for the R package name in namespace '........' in the Error in loadNamespace () warning message. Update the obsolete R package, creating installation failure using the following code:

```
install.packages("name of the package", dependencies = TRUE)
```

If there are too many obsolete packages, try installing the packages with their dependencies one after the other till the sraplus installation succeeds. Use the following code to update all the R packages. Press 'Yes' for every prompt to proceed with the update.

```
update.packages(checkBuilt = TRUE)
```

> ***Note:*** *The updation of all the available R packages is a time-consuming process and therefore, should be done if* ***Error in loadNamespace ()*** *warnings are generated even after updating rlang or other such dependent R packages, leading to repeated installation failure.*

After updating dependent R packages, retry sraplus installation using the following codes:

```
options(timeout=500)
library(remotes)
install_github("danovando/sraplus")
```

Installing TAF

Use the following code to install the FAO transparent assessment framework (TAF) R package:

```
install.packages("TAF")
```

Installing SOFIA

Use the following code to install the state of world fisheries and aquaculture (SOFIA) R package:

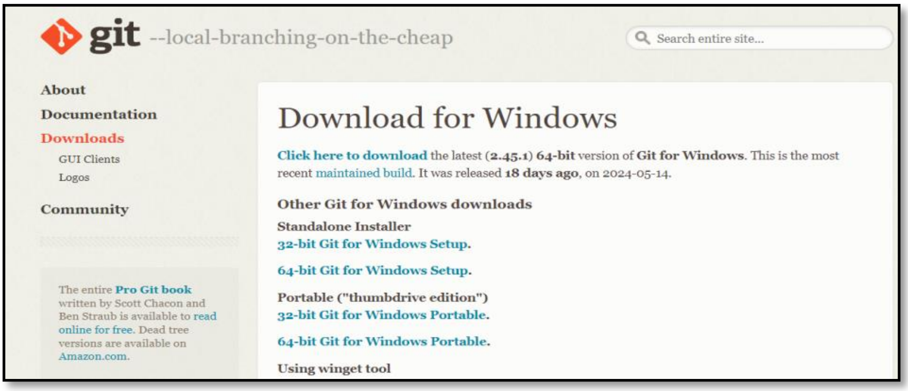
```
library(remotes)
install_github("sofia-taf/SOFIA")
```

Installing Git

Choose the Git depending on the operating system (OS) of the system following the below link: https://git-scm.com/download/

A direct link to get Git for the latest Windows OS 64-bit systems is given below:

https://github.com/git-for-windows/git/releases/download/v2.39.1.windows.1/Git-2.39.1-64-bit.exe



The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Download for Windows

Click here to download the latest (**2.45.1**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **18 days ago**, on 2024-05-14.

Other Git for Windows downloads

Standalone Installer
32-bit Git for Windows Setup.

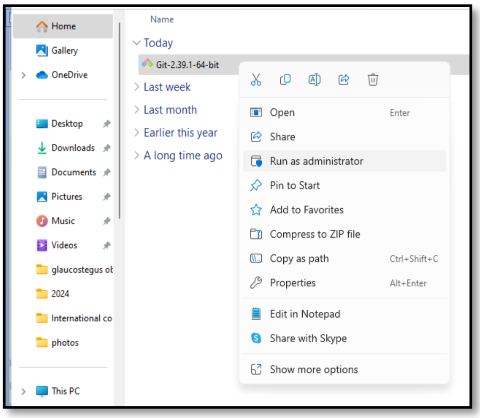
64-bit Git for Windows Setup.

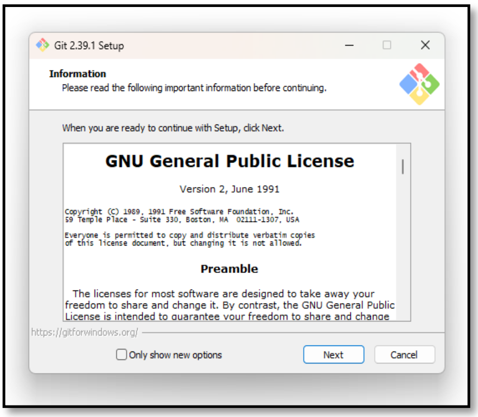
Portable ("thumbdrive edition")
32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

Using winget tool

After the download, install the Git executable file (ex: Git-2.39.1-64-bit.exe) as the Administrator. Give permission (ex: Yes/Next) when asked during the installation of the Git. Restart RStudio after the installation of Git.





Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

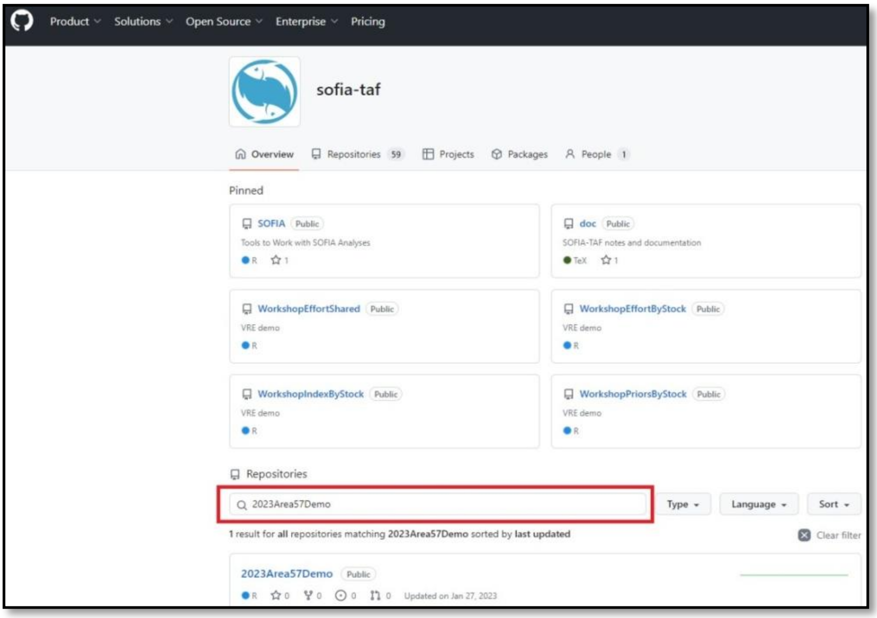
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

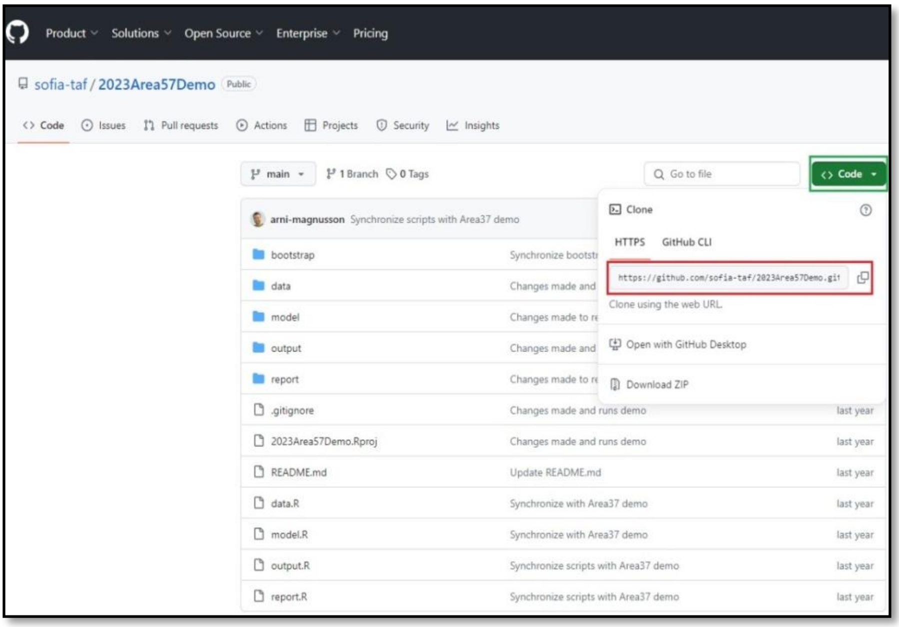
Cloning a demo data from the Git repository

Clone a demo data (ex: 2023Area57Demo) from the github (Git Repository). Go to https://github.com/sofia-taf. Type 2023Area57Demo in the search box (highlighted in red box) under the Repositories and search.

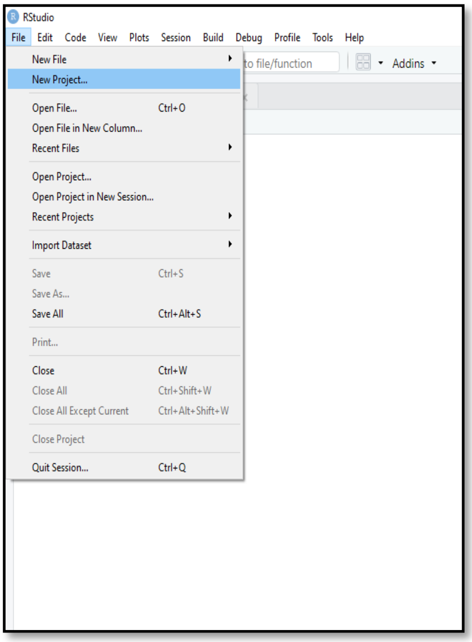


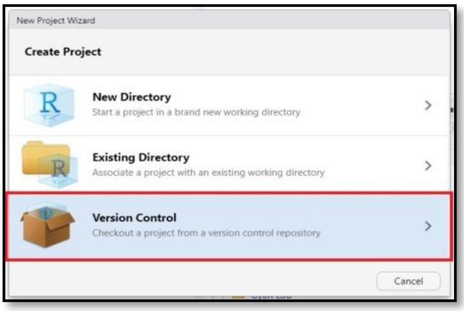
1 result for all repositories matching 2023Area57Demo sorted by last updated

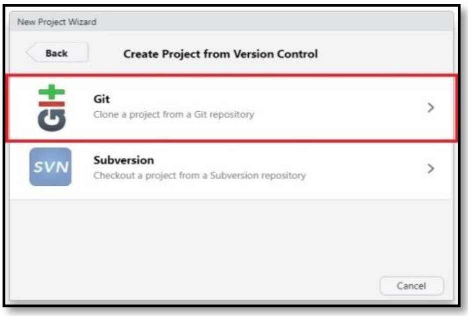
Click on the search result, i.e., 2023Area57Demo. Go to the "Code" tab (right side top green button) and click the link under HTTPS https://github.com/sofia-taf/2023Area57Demo.git



Now go to RStudio > File > new Project > version control > Git

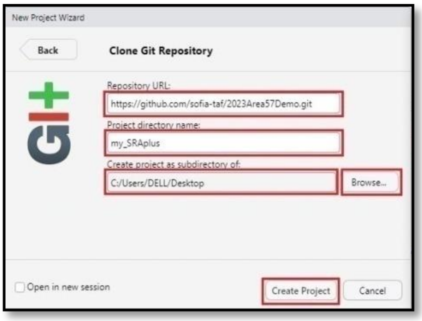



New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel


New Project Wizard
Back
Create Project from Version Control
Git
Clone a project from a Git repository
Subversion
Checkout a project from a Subversion repository
Cancel

Paste the copied link (ex: https://github.com/sofia-taf/2023Area57Demo.git) in the Repository URL: slot of the Clone Git Repository window to clone the sraplus codes and demo data from the Git Repository. Create a main directory (folder) at any desired location (e.g., Desktop) and give a name to the folder (e.g., my_SRAplus). Type the name (e.g., my_SRAplus) in the Project directory name: slot of the Clone Git Repository window. Now

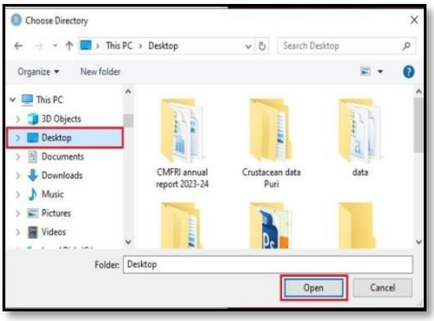
Press the Browse button of the Clone Git Repository window to search for the above-mentioned newly created directory/folder (e.g., my_SRAplus). Search the location (e.g., Desktop) for this newly created directory/folder (e.g., my_SRAplus) and click the same which will create a path (e.g., C:\Users\Dell\Desktop) for the subdirectories (subfolders) in the Create project as subdirectory of: slot of the Clone Git Repository window.



Repository URL: https://github.com/sofia-taf/2023Area57Demo.git

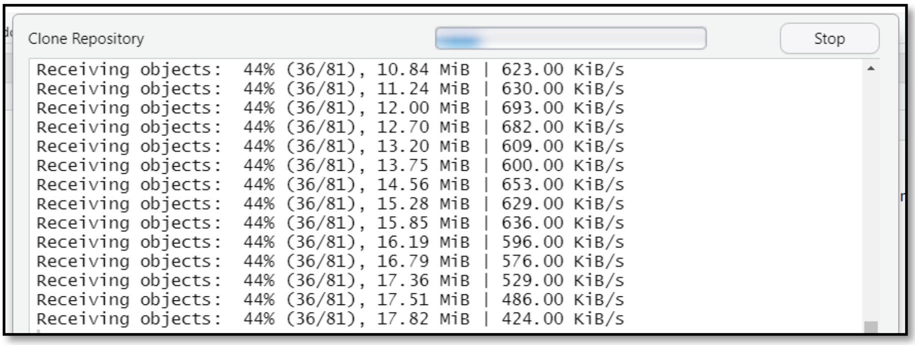
Project directory name: my_SRAplus

Create project as subdirectory of: C:/Users/DELL/Desktop



Folder: Desktop

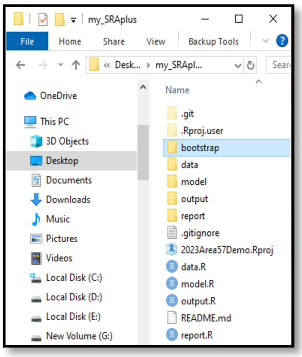
Finally, Press the Create Project button. It will start cloning the Repository.



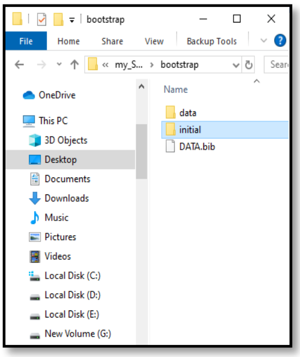
```
Receiving objects:  44% (36/81), 10.84 MiB | 623.00 KiB/s
Receiving objects:  44% (36/81), 11.24 MiB | 630.00 KiB/s
Receiving objects:  44% (36/81), 12.00 MiB | 693.00 KiB/s
Receiving objects:  44% (36/81), 12.70 MiB | 682.00 KiB/s
Receiving objects:  44% (36/81), 13.20 MiB | 609.00 KiB/s
Receiving objects:  44% (36/81), 13.75 MiB | 600.00 KiB/s
Receiving objects:  44% (36/81), 14.56 MiB | 653.00 KiB/s
Receiving objects:  44% (36/81), 15.28 MiB | 629.00 KiB/s
Receiving objects:  44% (36/81), 15.85 MiB | 636.00 KiB/s
Receiving objects:  44% (36/81), 16.19 MiB | 596.00 KiB/s
Receiving objects:  44% (36/81), 16.79 MiB | 576.00 KiB/s
Receiving objects:  44% (36/81), 17.36 MiB | 529.00 KiB/s
Receiving objects:  44% (36/81), 17.51 MiB | 486.00 KiB/s
Receiving objects:  44% (36/81), 17.82 MiB | 424.00 KiB/s
```

Preparing own input catch (and effort) data file

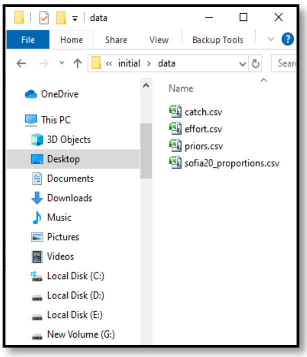
Open the my_SRAplus folder. Open the 'bootstrap' subfolder inside the my_SRAplus folder. Avoid the 'data' subfolder inside the 'bootstrap' subfolder. Now open the 'initial' subfolder inside the 'bootstrap' subfolder and finally the 'data' subfolder inside the initial' subfolder. The path is: C:\Users\Dell\Desktop\my_SRAplus\bootstrap\initial\data.



.git
.Rproj.user
bootstrap
data
model
output
report
.gitignore
2023Area57Demo.Rproj
data.R
model.R
output.R
README.md
report.R

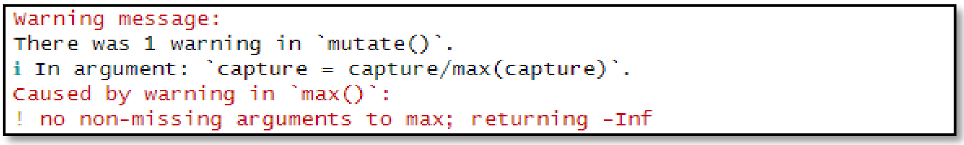


data
initial
DATA.bib



catch.csv
effort.csv
priors.csv
sofia20_proportions.csv

Now open the ‘catch.csv’ and copy and paste the year-wise catch data of your own. A minimum of 11 years of continuous time-series data is required for the sraplus analysis as a default setup. Otherwise, the following error message will be produced.



```
Warning message:
There was 1 warning in `mutate()`.
i In argument: `capture = capture/max(capture)`.
Caused by warning in `max()`:
! no non-missing arguments to max; returning -Inf
```

Open the ‘effort.csv’ and copy and paste the year-wise effort data of your own.

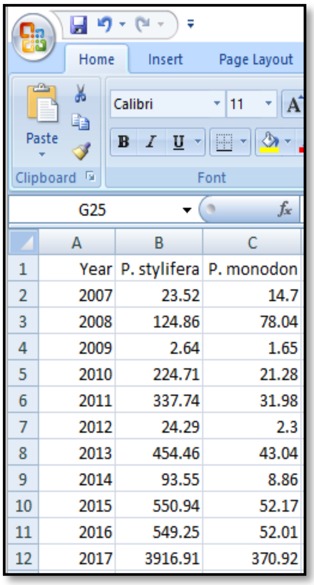
Open the ‘prior.csv’ and give information on priors such as initial and terminal biomass levels and their CVs (proportions).

Open the ‘sofia20_proportions.csv’ to categorise the stock health status.

Note: *Do not alter any catch, effort, priors and stock categorization data inside the ‘data’ subfolder of the ‘bootstrap’ subfolder, as it will be auto-populated from the ‘data’ subfolder inside the ‘initial’ subfolder of the ‘bootstrap’ subfolder after using the ‘taf.bootstrap()’ code that erases all the traces of any previous analysis.*

Catch and effort file types:

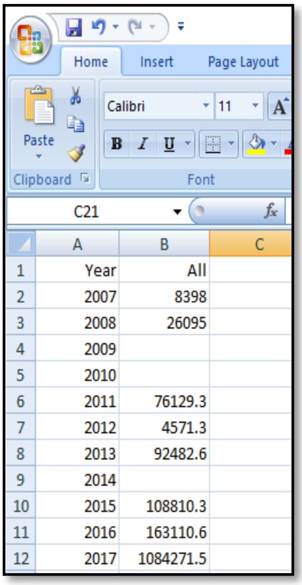
catch.csv
single species or
multi-species catch



	A	B	C
1	Year	P. stylifera	P. monodon
2	2007	23.52	14.7
3	2008	124.86	78.04
4	2009	2.64	1.65
5	2010	224.71	21.28
6	2011	337.74	31.98
7	2012	24.29	2.3
8	2013	454.46	43.04
9	2014	93.55	8.86
10	2015	550.94	52.17
11	2016	549.25	52.01
12	2017	3916.91	370.92

A minimum of 11 years time-series catch data is required

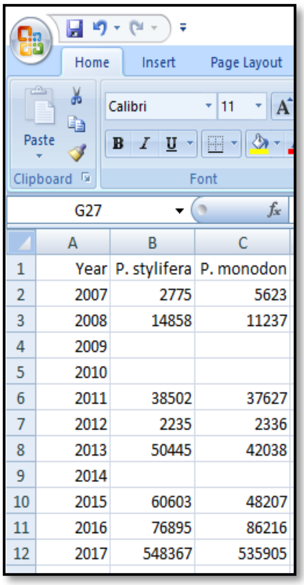
effort.csv
single species effort or
multi-species exploited by
the same level of effort



	A	B	C
1	Year	All	
2	2007	8398	
3	2008	26095	
4	2009		
5	2010		
6	2011	76129.3	
7	2012	4571.3	
8	2013	92482.6	
9	2014		
10	2015	108810.3	
11	2016	163110.6	
12	2017	1084271.5	

Make same.effort=TRUE in the Line 60 of data.R

effort.csv
multi-species exploited by
different level of efforts



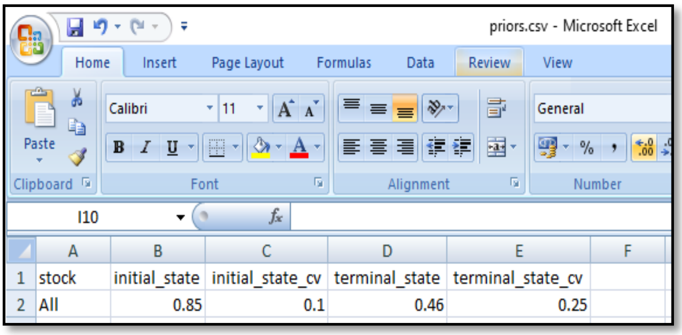
	A	B	C
1	Year	P. stylifera	P. monodon
2	2007	2775	5623
3	2008	14858	11237
4	2009		
5	2010		
6	2011	38502	37627
7	2012	2235	2336
8	2013	50445	42038
9	2014		
10	2015	60603	48207
11	2016	76895	86216
12	2017	548367	535905

Make same.effort=FALSE in the Line 60 of data.R

Preparing own input prior parameter file

Priors file types

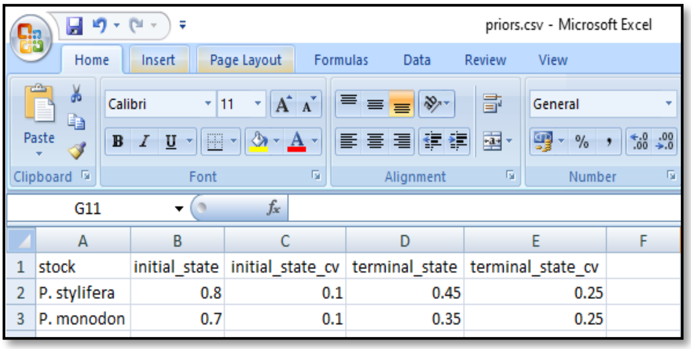
priors.csv



	A	B	C	D	E	F
1	stock	initial_state	initial_state_cv	terminal_state	terminal_state_cv	
2	All	0.85	0.1	0.46	0.25	

A common initial and final biomass status and their CV can be provided as priors for all the available stocks. Make same.priors=TRUE in the Line 70 of data.R.

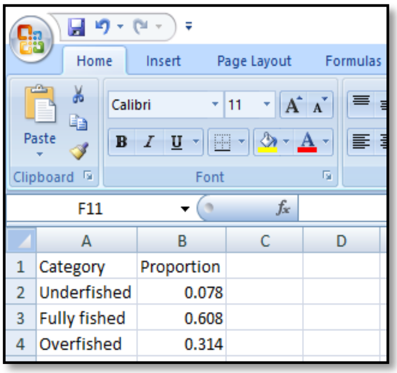
priors.csv



	A	B	C	D	E	F
1	stock	initial_state	initial_state_cv	terminal_state	terminal_state_cv	
2	P. stylifera	0.8	0.1	0.45	0.25	
3	P. monodon	0.7	0.1	0.35	0.25	

A stock-wise different initial and final biomass status and their CV can be provided as priors for all the available stocks. Make same.priors=FALSE in the Line 70 of data.R. Refer ***'Example data file download link'*** in the last page to download and use the example data.

sofia20_proportions.csv



	A	B	C	D
1	Category	Proportion		
2	Underfished	0.078		
3	Fully fished	0.608		
4	Overfished	0.314		

The values provided in the CSV to compare the current analysis results with this previous assessment status of SOFIA, which is depicted in the status_summary.png of report. Instead of SOFIA status, the values from the last assessment status can be supplied here (in sofia20_proportions.csv) to compare the current assessment status with the previous assessment status. Additionally, change the xlab="Last SOFIA" to xlab="Last assessment" in Line 48 of the report.R.

3.3.5. Running the sraplus analysis

Erase traces of previous analysis

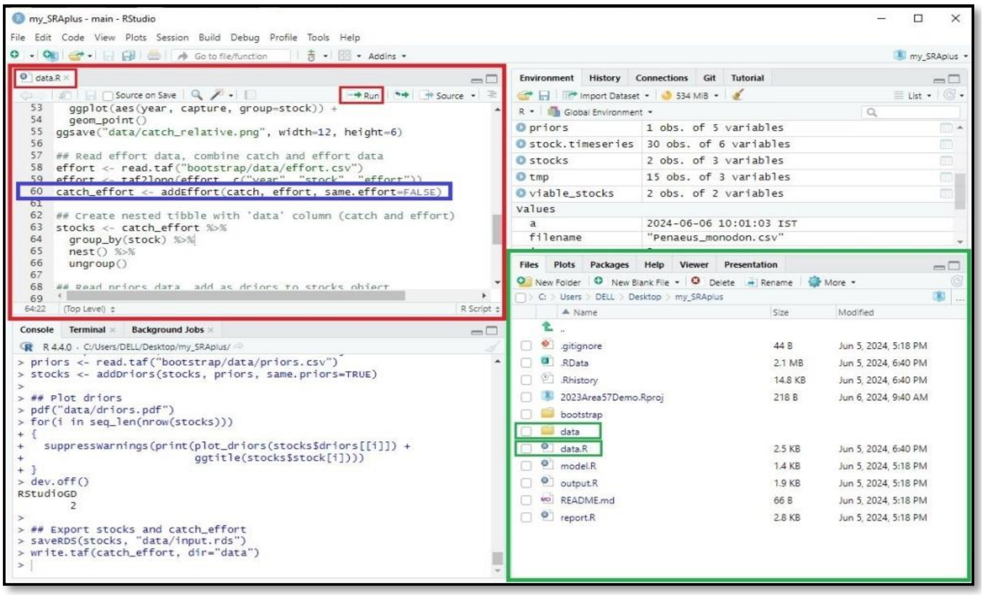
To erase any traces of previous analysis (ex: contents in the previously created data, model, outputs and report subfolders inside the my_SRAplus folder), use the following codes on the just cloned demo data (ex: 2023Area57Demo).

```
library(TAF)
clean()
clean("bootstrap", force=TRUE)
taf.bootstrap()
```

Note: *Repeat the step every time any change is made in the catch, effort, priors and stock data of the 'data' subfolder inside the 'initial' subfolder of the 'bootstrap' subfolder. It will populate the catch, effort, priors and stock data in the 'data' subfolder of the 'bootstrap' subfolder. In simple words, the procedure ensures the flow of data from the boot/initial/data subfolder to boot/data subfolder, and finally to a newly created data subfolder.*

Preparation of data

Click the data.R (right side bottom window shown inside the green box). The R codes in the data.R will be visible on the left side top window (shown inside the red box). Select all the codes by pressing control+A) and Run (shown inside the red box).



```
53    ggplot(aes(year, capture, group=stock)) +
54    geom_point()
55  ggsave("data/catch_relative.png", width=12, height=6)
56
57  ## Read effort data, combine catch and effort data
58  effort <- read.taf("bootstrap/data/effort.csv")
59  effort <- taf2long(effort, c("year", "stock", "effort"))
60  catch_effort <- addEffort(catch, effort, same.effort=FALSE)
61
62  ## Create nested tibble with 'data' column (catch and effort)
63  stocks <- catch_effort %>%
64    group_by(stock) %>%
65    nest() %>%
66    ungroup()
67
68  ## Read priors data, add as driors to stocks object
69
```

```
> priors <- read.taf("bootstrap/data/priors.csv")
> stocks <- addDriors(stocks, priors, same.priors=TRUE)
>
> ## Plot driors
> pdf("data/driors.pdf")
> for(i in seq_len(nrow(stocks)))
+ {
+   suppressWarnings(print(plot_driors(stocks$driors[[i]]) +
+                          ggtitle(stocks$stock[i])))
+ }
> dev.off()
RStudioGD
        2
>
> ## Export stocks and catch_effort
> saveRDS(stocks, "data/input.rds")
> write.taf(catch_effort, dir="data")
> 
```

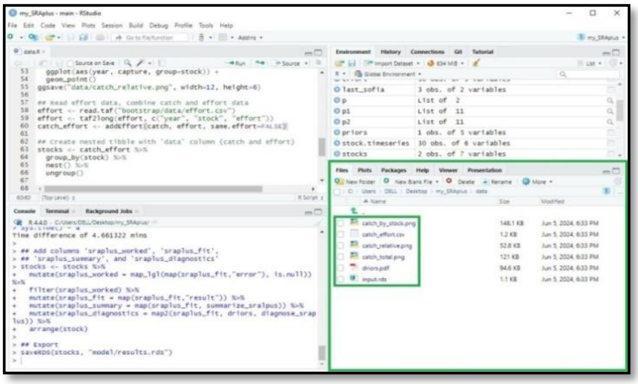
```
priors             1 obs. of 5 variables
stock.timeseries   30 obs. of 6 variables
stocks             2 obs. of 3 variables
tmp                15 obs. of 3 variables
viable_stocks      2 obs. of 2 variables
Values
  a                2024-06-06 10:01:03 IST
  filename         "Penaeus_monodon.csv"
```

Name	Size	Modified
..		
.gitignore	44 B	Jun 5, 2024, 5:18 PM
.RData	2.1 MB	Jun 5, 2024, 6:40 PM
.Rhistory	14.8 KB	Jun 5, 2024, 6:40 PM
2023Area57Demo.Rproj	218 B	Jun 6, 2024, 9:40 AM
bootstrap		
data		
data.R	2.5 KB	Jun 5, 2024, 6:40 PM
model.R	1.4 KB	Jun 5, 2024, 5:18 PM
output.R	1.9 KB	Jun 5, 2024, 5:18 PM
README.md	66 B	Jun 5, 2024, 5:18 PM
report.R	2.8 KB	Jun 5, 2024, 5:18 PM

Note: *If one species or more than one species are there for which the effort levels are the same (Refer to the Catch and effort file types), then the same.effort=TRUE should be kept unchanged and TRUE in Line 60 of data.R (shown inside the blue box). If more than one species are there for which the efforts are different (Refer to Catch and effort file types), then the same.effort=TRUE should be changed to same.effort=FALSE in Line 60 of data.R (shown inside the blue box).*

A data subfolder will be created (right side bottom window shown in the red box). Click it to view the data, plots including input.rds in the RStudio which is going to be used for modeling. These input files can also be seen inside the data subfolder inside the my_SRAplus folder created on the Desktop.

RStudio view of data subfolder | **Windows view of data subfolder**

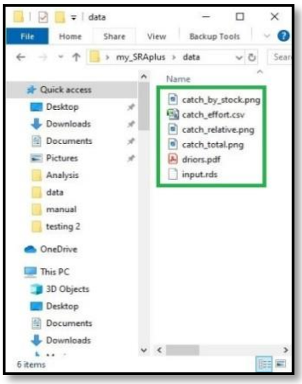

my_SRAplus - main - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
data.R
Source on Save
Run
Source
Console Terminal Background Jobs
R 4.4.0 · C:/Users/DELL/Desktop/my_SRAplus/
Environment History Connections Git Tutorial
Import Dataset
634 MiB
Global Environment
Files Plots Packages Help Viewer Presentation
New Folder New Blank File Delete Rename More
Users › DELL › Desktop › my_SRAplus › data
Name Size Modified
my_SRAplus

```
  ggplot(aes(year, capture, group=stock)) +
  geom_point()
ggsave("data/catch_relative.png", width=12, height=6)

## Read effort data, combine catch and effort data
effort <- read.taf("bootstrap/data/effort.csv")
effort <- taf2long(effort, c("year", "stock", "effort"))
catch_effort <- addEffort(catch, effort, same.effort=FALSE)

## Create nested tibble with 'data' column (catch and effort)
stocks <- catch_effort %>%
  group_by(stock) %>%
  nest() %>%
  ungroup()
```

```
Time difference of 4.661322 mins
>
> ## Add columns 'sraplus_worked', 'sraplus_fit',
> ## 'sraplus_summary', and 'sraplus_diagnostics'
> stocks <- stocks %>%
+   mutate(sraplus_worked = map_lgl(map(sraplus_fit,"error"), is.null))
%>%
+   filter(sraplus_worked) %>%
+   mutate(sraplus_fit = map(sraplus_fit,"result")) %>%
+   mutate(sraplus_summary = map(sraplus_fit, summarize_sralpus)) %>%
+   mutate(sraplus_diagnostics = map2(sraplus_fit, driors, diagnose_srap
lus)) %>%
+   arrange(stock)
>
> ## Export
> saveRDS(stocks, "model/results.rds")
>
```

last_sofia	3 obs. of 2 variables	
p	List of 2	
p1	List of 11	
p2	List of 11	
priors	1 obs. of 5 variables	
stock.timeseries	30 obs. of 6 variables	
stocks	2 obs. of 7 variables	

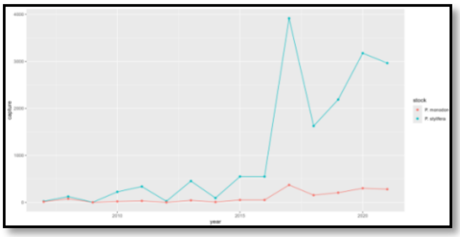
Name	Size	Modified
catch_by_stock.png	148.1 KB	Jun 5, 2024, 6:33 PM
catch_effort.csv	1.2 KB	Jun 5, 2024, 6:33 PM
catch_relative.png	52.8 KB	Jun 5, 2024, 6:33 PM
catch_total.png	121 KB	Jun 5, 2024, 6:33 PM
driors.pdf	94.6 KB	Jun 5, 2024, 6:33 PM
input.rds	1.1 KB	Jun 5, 2024, 6:33 PM



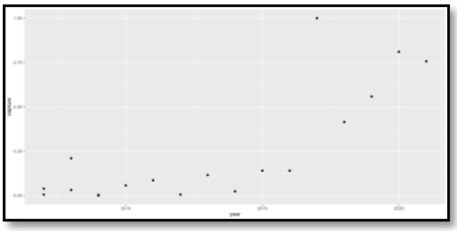
Name
- catch_by_stock.png
- catch_effort.csv
- catch_relative.png
- catch_total.png
- driors.pdf
- input.rds

Clicking the individual items inside the data subfolder will produce the following outputs.

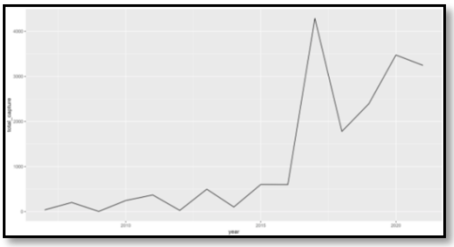
catch_by_stock.png



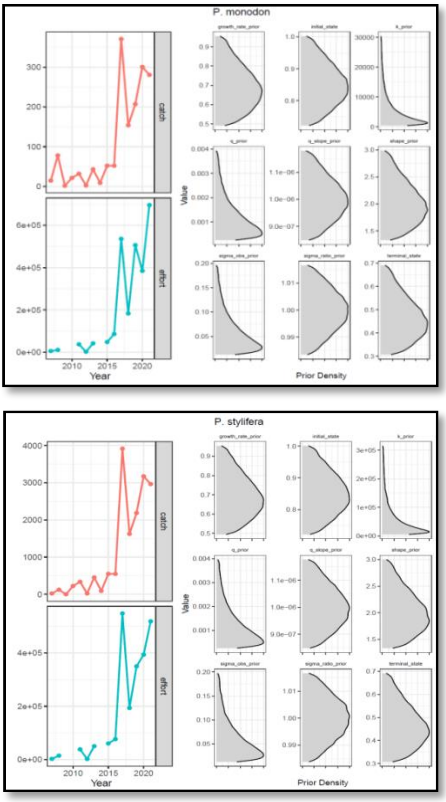
catch_relative.png



catch_total.png

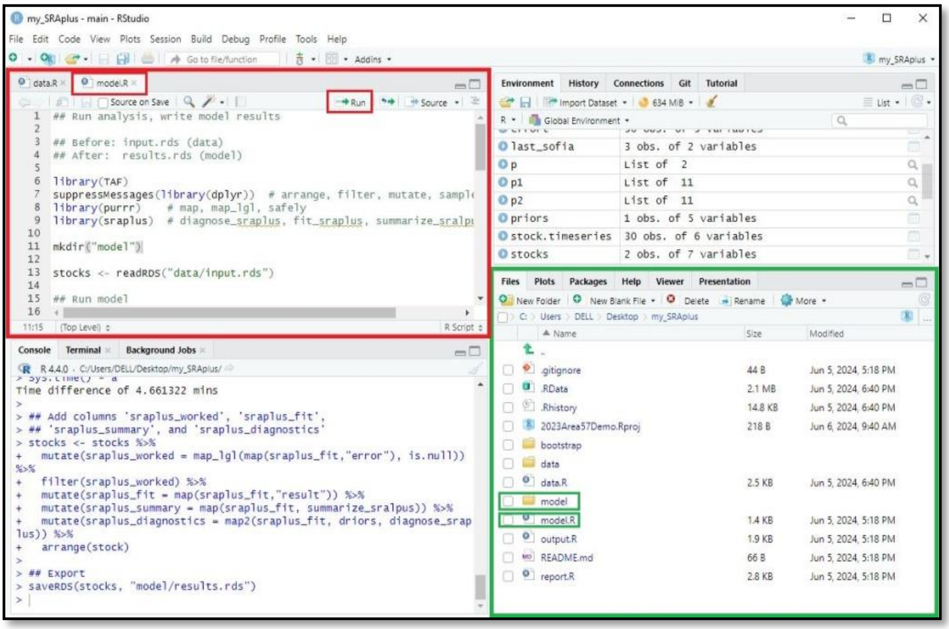


Driors.pdf



sraplus modeling

Click the model.R (right side bottom window shown inside the green box). The R codes in the model.R will be visible on the left side top window (shown inside the red box). Select all the codes (click anywhere on the code and Press control+A) and Run.



```
## Run analysis, write model results

## Before: input.rds (data)
## After:  results.rds (model)

library(TAF)
suppressMessages(library(dplyr))  # arrange, filter, mutate, sampl
library(purrr)     # map, map_lgl, safely
library(sraplus)  # diagnose_sraplus, fit_sraplus, summarize_sralpu

mkdir("model")

stocks <- readRDS("data/input.rds")

## Run model
```

```
R 4.4.0 · C:/Users/DELL/Desktop/my_SRAplus/
> Sys.time() - a
Time difference of 4.661322 mins
>
> ## Add columns 'sraplus_worked', 'sraplus_fit',
> ## 'sraplus_summary', and 'sraplus_diagnostics'
> stocks <- stocks %>%
+   mutate(sraplus_worked = map_lgl(map(sraplus_fit,"error"), is.null))
%>%
+   filter(sraplus_worked) %>%
+   mutate(sraplus_fit = map(sraplus_fit,"result")) %>%
+   mutate(sraplus_summary = map(sraplus_fit, summarize_sralpus)) %>%
+   mutate(sraplus_diagnostics = map2(sraplus_fit, driors, diagnose_srap
lus)) %>%
+   arrange(stock)
>
> ## Export
> saveRDS(stocks, "model/results.rds")
> |
```

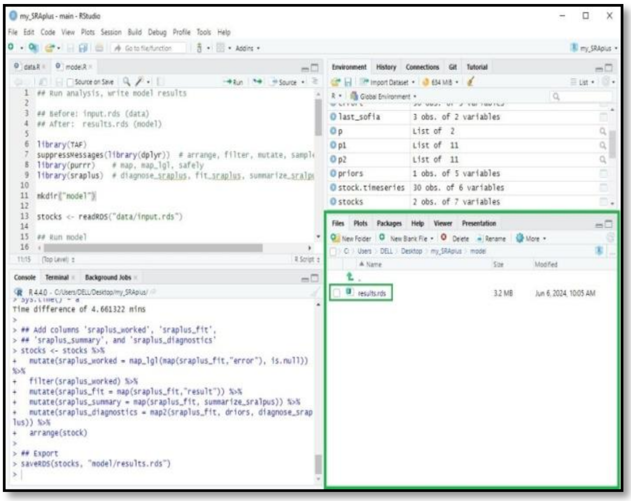
```
last_sofia          3 obs. of 2 variables
p                   List of  2
p1                  List of  11
p2                  List of  11
priors              1 obs. of 5 variables
stock.timeseries    30 obs. of 6 variables
stocks              2 obs. of 7 variables
```

```
C: > Users > DELL > Desktop > my_SRAplus
Name                    Size     Modified
.gitignore              44 B     Jun 5, 2024, 5:18 PM
.RData                  2.1 MB   Jun 5, 2024, 6:40 PM
.Rhistory               14.8 KB  Jun 5, 2024, 6:40 PM
2023Area57Demo.Rproj    218 B    Jun 6, 2024, 9:40 AM
bootstrap
data
data.R                  2.5 KB   Jun 5, 2024, 6:40 PM
model
model.R                 1.4 KB   Jun 5, 2024, 5:18 PM
output.R                1.9 KB   Jun 5, 2024, 5:18 PM
README.md               66 B     Jun 5, 2024, 5:18 PM
report.R                2.8 KB   Jun 5, 2024, 5:18 PM
```

A model subfolder will be created (right side bottom window shown inside the green box). Click it to view the model outputs (results.rds) in the RStudio. This results.rds can also be seen inside the model subfolder inside the my_SRAplus folder created on the Desktop.

RStudio view of model subfolder

Windows view of model subfolder

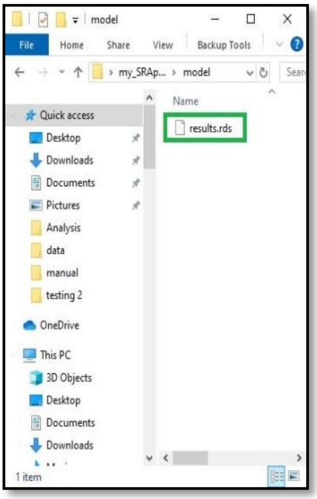


```
C: > Users > DELL > Desktop > my_SRAplus > model
Name           Size     Modified
results.rds    3.2 MB   Jun 6, 2024, 10:05 AM
```

```
## Run analysis, write model results

## Before: input.rds (data)
## After:  results.rds (model)

library(TAF)
suppressMessages(library(dplyr))  # arrange, filter, mutate, sampl
library(purrr)     # map, map_lgl, safely
library(sraplus)  # diagnose_sraplus, fit_sraplus, summarize_sralpu

mkdir("model")

stocks <- readRDS("data/input.rds")

## Run model
```

```
R 4.4.0 · C:/Users/DELL/Desktop/my_SRAplus/
> Sys.time() - a
Time difference of 4.661322 mins
>
> ## Add columns 'sraplus_worked', 'sraplus_fit',
> ## 'sraplus_summary', and 'sraplus_diagnostics'
> stocks <- stocks %>%
+   mutate(sraplus_worked = map_lgl(map(sraplus_fit,"error"), is.null))
%>%
+   filter(sraplus_worked) %>%
+   mutate(sraplus_fit = map(sraplus_fit,"result")) %>%
+   mutate(sraplus_summary = map(sraplus_fit, summarize_sralpus)) %>%
+   mutate(sraplus_diagnostics = map2(sraplus_fit, driors, diagnose_srap
lus)) %>%
+   arrange(stock)
>
> ## Export
> saveRDS(stocks, "model/results.rds")
> |
```

```
last_sofia          3 obs. of 2 variables
p                   List of  2
p1                  List of  11
p2                  List of  11
priors              1 obs. of 5 variables
stock.timeseries    30 obs. of 6 variables
stocks              2 obs. of 7 variables
```



Model engine customization

The sraplus uses three algorithms for the modeling, i.e., (1) SIR, (2) tmb, and (3) stan, which can be controlled by changing the engine (Line no. 27) of the model.R.

The engine argument specifies how the model will be fit.

To implement ***stan* algorithms**, make engine = "stan" in the Line no. 27 of model.R.

(Use STAN as the engine when data on a perfect index of abundance is not available, rather data on catch and effort (CPUE) is available, to fit the model via the Bayesian approach)

To implement ***tmb algorithms***, make engine = "tmb" in the Line no. 27 of model.R.

(Use Template Model Builder (TMB) as the engine when there is a perfect index of abundance (e.g., Biomass) to fit the model via maximum likelihood approach)

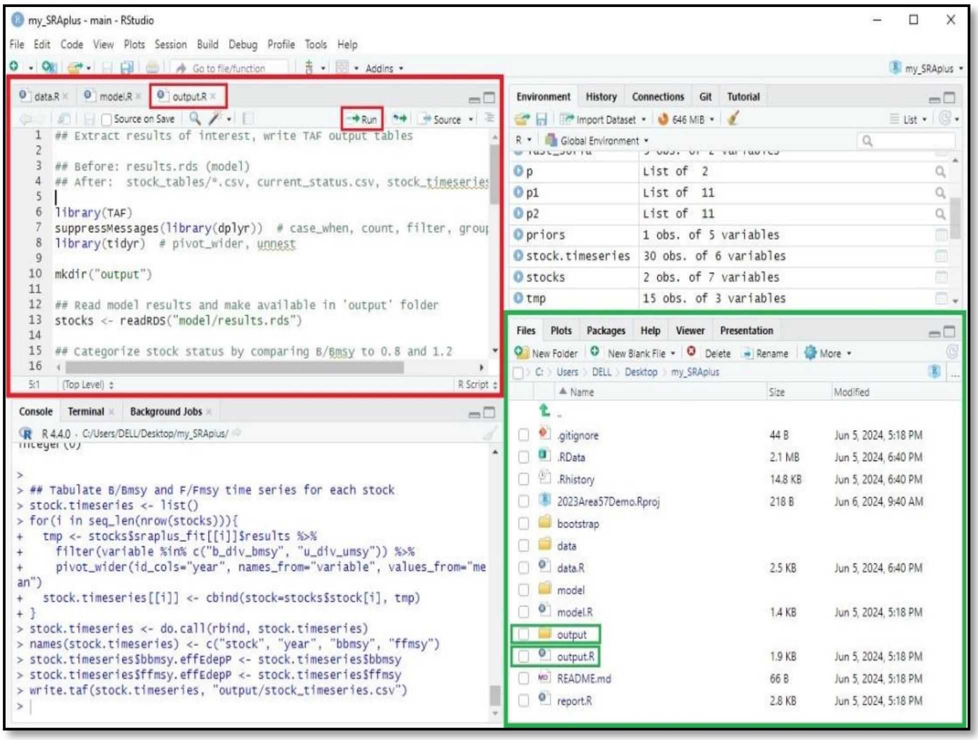
To implement ***SIR algorithms***, make engine = "SIR" in the Line no. 27 of model.R.

(Use SIR as the engine when not actually “fitting” to anything, rather simply sampling from priors that don’t crash the population)

3.3.6. sraplus tabular outputs

Click the output.R (right side bottom window shown inside the green box).

The R codes in the output.R will be visible on the left side top window (shown inside the red box). Select all the codes (click anywhere on the code and press control+A) and Run.



```
## Extract results of interest, write TAF output tables

## Before: results.rds (model)
## After:  stock_tables/*.csv, current_status.csv, stock_timeseries

library(TAF)
suppressMessages(library(dplyr))  # case_when, count, filter, group
library(tidyr)  # pivot_wider, unnest

mkdir("output")

## Read model results and make available in 'output' folder
stocks <- readRDS("model/results.rds")

## Categorize stock status by comparing B/Bmsy to 0.8 and 1.2
```

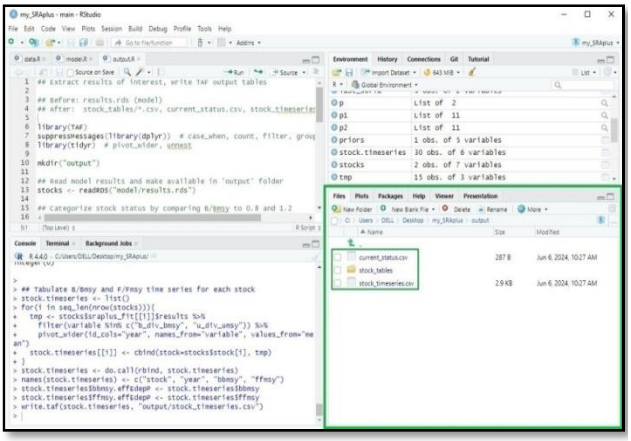
```
integer (0)

>
> ## Tabulate B/Bmsy and F/Fmsy time series for each stock
> stock.timeseries <- list()
> for(i in seq_len(nrow(stocks))){
+   tmp <- stocks$sraplus_fit[[i]]$results %>%
+     filter(variable %in% c("b_div_bmsy", "u_div_umsy")) %>%
+     pivot_wider(id_cols="year", names_from="variable", values_from="me
an")
+   stock.timeseries[[i]] <- cbind(stock=stocks$stock[i], tmp)
+ }
> stock.timeseries <- do.call(rbind, stock.timeseries)
> names(stock.timeseries) <- c("stock", "year", "bbmsy", "ffmsy")
> stock.timeseries$bbmsy.effEdepP <- stock.timeseries$bbmsy
> stock.timeseries$ffmsy.effEdepP <- stock.timeseries$ffmsy
> write.taf(stock.timeseries, "output/stock_timeseries.csv")
> |
```

p	List of 2
p1	List of 11
p2	List of 11
priors	1 obs. of 5 variables
stock.timeseries	30 obs. of 6 variables
stocks	2 obs. of 7 variables
tmp	15 obs. of 3 variables

Name	Size	Modified
..		
.gitignore	44 B	Jun 5, 2024, 5:18 PM
.RData	2.1 MB	Jun 5, 2024, 6:40 PM
.Rhistory	14.8 KB	Jun 5, 2024, 6:40 PM
2023Area57Demo.Rproj	218 B	Jun 6, 2024, 9:40 AM
bootstrap		
data		
data.R	2.5 KB	Jun 5, 2024, 6:40 PM
model		
model.R	1.4 KB	Jun 5, 2024, 5:18 PM
output		
output.R	1.9 KB	Jun 5, 2024, 5:18 PM
README.md	66 B	Jun 5, 2024, 5:18 PM
report.R	2.8 KB	Jun 5, 2024, 5:18 PM

An output folder will be created (right side bottom window shown inside the green box). Click it to view the processed outputs (current_status.csv, stock_timeseries.csv and stock_tables subfolders) in the RStudio. These outputs can also be seen inside the output subfolder inside the my_SRAplus folder created on the Desktop.

RStudio view of output subfolder



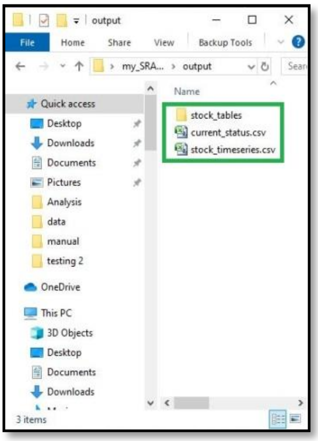
```
## Extract results of interest, write TAF output tables

## Before: results.rds (model)
## After:  stock_tables/*.csv, current_status.csv, stock_timeseries

library(TAF)
suppressMessages(library(dplyr))  # case_when, count, filter, grou
library(tidyr)  # pivot_wider, unnest

mkdir("output")

## Read model results and make available in 'output' folder
stocks <- readRDS("model/results.rds")

## Categorize stock status by comparing B/Bmsy to 0.8 and 1.2
```

```
>
> ## Tabulate B/Bmsy and F/Fmsy time series for each stock
> stock.timeseries <- list()
> for(i in seq_len(nrow(stocks))){
+   tmp <- stocks$sraplus_fit[[i]]$results %>%
+     filter(variable %in% c("b_div_bmsy", "u_div_umsy")) %>%
+     pivot_wider(id_cols="year", names_from="variable", values_from="me
an")
+   stock.timeseries[[i]] <- cbind(stock=stocks$stock[i], tmp)
+ }
> stock.timeseries <- do.call(rbind, stock.timeseries)
> names(stock.timeseries) <- c("stock", "year", "bbmsy", "ffmsy")
> stock.timeseries$bbmsy.effEdepP <- stock.timeseries$bbmsy
> stock.timeseries$ffmsy.effEdepP <- stock.timeseries$ffmsy
> write.taf(stock.timeseries, "output/stock_timeseries.csv")
>
```

```
p                 List of  2
p1                List of  11
p2                List of  11
priors            1 obs. of 5 variables
stock.timeseries  30 obs. of 6 variables
stocks            2 obs. of 7 variables
tmp               15 obs. of 3 variables
```

```
current_status.csv      287 B    Jun 6, 2024, 10:27 AM
stock_tables
stock_timeseries.csv    2.9 KB   Jun 6, 2024, 10:27 AM
```

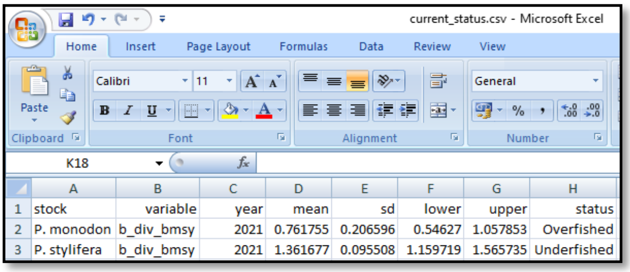
Windows view of output subfolder



stock_tables
current_status.csv
stock_timeseries.csv

Clicking the individual items inside the output subfolder will produce the following outputs.

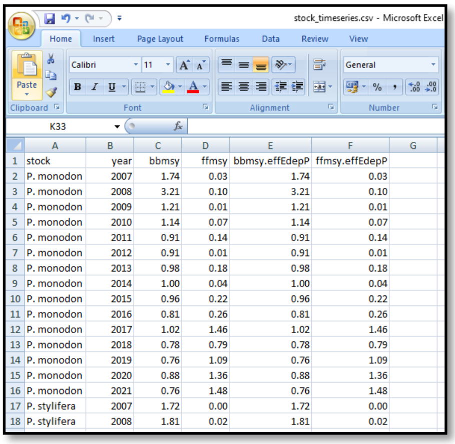
current_status.csv



	A	B	C	D	E	F	G	H
1	stock	variable	year	mean	sd	lower	upper	status
2	P. monodon	b_div_bmsy	2021	0.761755	0.206596	0.54627	1.057853	Overfished
3	P. stylifera	b_div_bmsy	2021	1.361677	0.095508	1.159719	1.565735	Underfished

The result shows the mean B/Bmsy for *P. monodon* is 0.76, which shows that the stock is in an overfished status. However, *P.stylifera* with a B/Bmsy of 1.36 is in an underfished status.

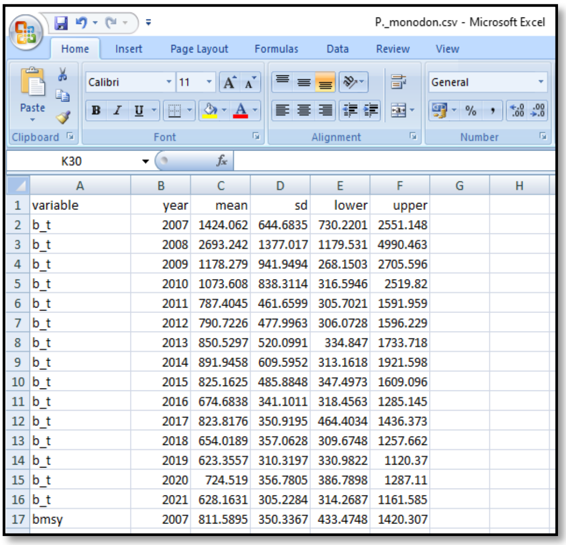
stock_timeseries.csv



	A	B	C	D	E	F	G
1	stock	year	bbmsy	ffmsy	bbmsy.effEdepP	ffmsy.effEdepP	
2	P. monodon	2007	1.74	0.03	1.74	0.03	
3	P. monodon	2008	3.21	0.10	3.21	0.10	
4	P. monodon	2009	1.21	0.01	1.21	0.01	
5	P. monodon	2010	1.14	0.07	1.14	0.07	
6	P. monodon	2011	0.91	0.14	0.91	0.14	
7	P. monodon	2012	0.91	0.01	0.91	0.01	
8	P. monodon	2013	0.98	0.18	0.98	0.18	
9	P. monodon	2014	1.00	0.04	1.00	0.04	
10	P. monodon	2015	0.96	0.22	0.96	0.22	
11	P. monodon	2016	0.81	0.26	0.81	0.26	
12	P. monodon	2017	1.02	1.46	1.02	1.46	
13	P. monodon	2018	0.78	0.79	0.78	0.79	
14	P. monodon	2019	0.76	1.09	0.76	1.09	
15	P. monodon	2020	0.88	1.36	0.88	1.36	
16	P. monodon	2021	0.76	1.48	0.76	1.48	
17	P. stylifera	2007	1.72	0.00	1.72	0.00	
18	P. stylifera	2008	1.81	0.02	1.81	0.02	

The result shows the time-series (year-wise) B/Bmsy and F/Fmsy status for the two analysed stocks of *P. monodon* and *P.stylifera*.

CSV files inside the stock_tables subfolder


P._monodon.csv - Microsoft Excel
Home
Insert
Page Layout
Formulas
Data
Review
View
Paste
Clipboard
Calibri
11
Font
Alignment
General
Number
K30
fx

variable	year	mean	sd	lower	upper
b_t	2007	1424.062	644.6835	730.2201	2551.148
b_t	2008	2693.242	1377.017	1179.531	4990.463
b_t	2009	1178.279	941.9494	268.1503	2705.596
b_t	2010	1073.608	838.3114	316.5946	2519.82
b_t	2011	787.4045	461.6599	305.7021	1591.959
b_t	2012	790.7226	477.9963	306.0728	1596.229
b_t	2013	850.5297	520.0991	334.847	1733.718
b_t	2014	891.9458	609.5952	313.1618	1921.598
b_t	2015	825.1625	485.8848	347.4973	1609.096
b_t	2016	674.6838	341.1011	318.4563	1285.145
b_t	2017	823.8176	350.9195	464.4034	1436.373
b_t	2018	654.0189	357.0628	309.6748	1257.662
b_t	2019	623.3557	310.3197	330.9822	1120.37
b_t	2020	724.519	356.7805	386.7898	1287.11
b_t	2021	628.1631	305.2284	314.2687	1161.585
bmsy	2007	811.5895	350.3367	433.4748	1420.307

Depending on the numbers of stocks in the input data, the .csv files will be created. These individual .csv files will give detailed outputs on all the parameters of the stocks in a tabular form.

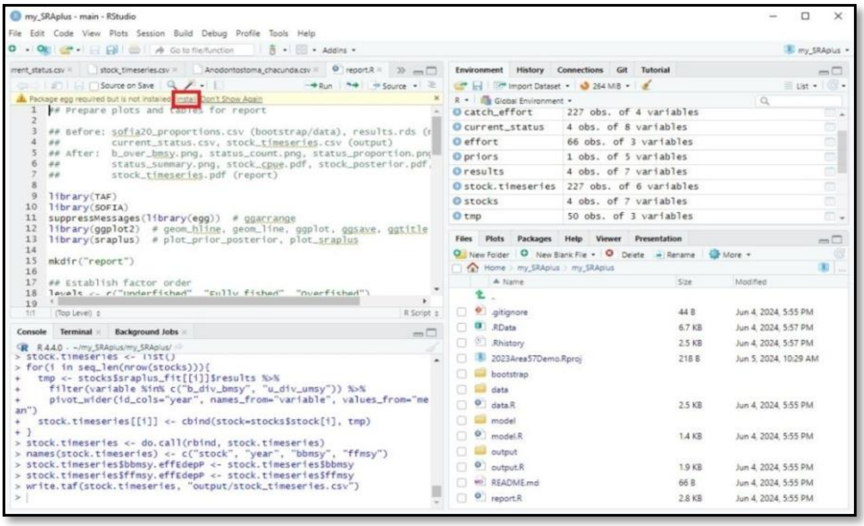
3.3.7. sraplus graphical outputs

The graphical output can be prepared by executing the report.R. However, the R script will not work if the dependent "egg" R package is not previously installed.

If the package "egg" has not been installed yet, a warning message will flag as follows:

Package egg required but is not installed, **Install**/ Don't Show Again.

Press the **Install** (shown in the red box in below image) to install the R package 'egg'.



Package egg required but is not installed. Install Don't Show Again

```
## Prepare plots and tables for report

## Before: sofia20_proportions.csv (bootstrap/data), results.rds (r
##         current_status.csv, stock_timeseries.csv (output)
## After:  b_over_bmsy.png, status_count.png, status_proportion.pnc
##         status_summary.png, stock_cpue.pdf, stock_posterior.pdf,
##         stock_timeseries.pdf (report)

library(TAF)
library(SOFIA)
suppressMessages(library(egg))  # ggarrange
library(ggplot2)  # geom_hline, geom_line, ggplot, ggsave, ggtitle
library(sraplus)  # plot_prior_posterior, plot_sraplus

mkdir("report")

## Establish factor order
levels <- c("underfished", "Fully fished", "Overfished")
```

```
R 4.4.0 · ~/my_SRAplus/my_SRAplus/
> stock.timeseries <- list()
> for(i in seq_len(nrow(stocks))){
+   tmp <- stocks$sraplus_fit[[i]]$results %>%
+     filter(variable %in% c("b_div_bmsy", "u_div_umsy")) %>%
+     pivot_wider(id_cols="year", names_from="variable", values_from="me
an")
+   stock.timeseries[[i]] <- cbind(stock=stocks$stock[i], tmp)
+ }
> stock.timeseries <- do.call(rbind, stock.timeseries)
> names(stock.timeseries) <- c("stock", "year", "bbmsy", "ffmsy")
> stock.timeseries$bbmsy.effEdepP <- stock.timeseries$bbmsy
> stock.timeseries$ffmsy.effEdepP <- stock.timeseries$ffmsy
> write.taf(stock.timeseries, "output/stock_timeseries.csv")
> |
```

catch_effort	227 obs. of 4 variables
current_status	4 obs. of 8 variables
effort	66 obs. of 3 variables
priors	1 obs. of 5 variables
results	4 obs. of 7 variables
stock.timeseries	227 obs. of 6 variables
stocks	4 obs. of 7 variables
tmp	50 obs. of 3 variables

Name	Size	Modified
..		
.gitignore	44 B	Jun 4, 2024, 5:55 PM
.RData	6.7 KB	Jun 4, 2024, 5:57 PM
.Rhistory	2.5 KB	Jun 4, 2024, 5:57 PM
2023Area57Demo.Rproj	218 B	Jun 5, 2024, 10:29 AM
bootstrap		
data		
data.R	2.5 KB	Jun 4, 2024, 5:55 PM
model		
model.R	1.4 KB	Jun 4, 2024, 5:55 PM
output		
output.R	1.9 KB	Jun 4, 2024, 5:55 PM
README.md	66 B	Jun 4, 2024, 5:55 PM
report.R	2.8 KB	Jun 4, 2024, 5:55 PM

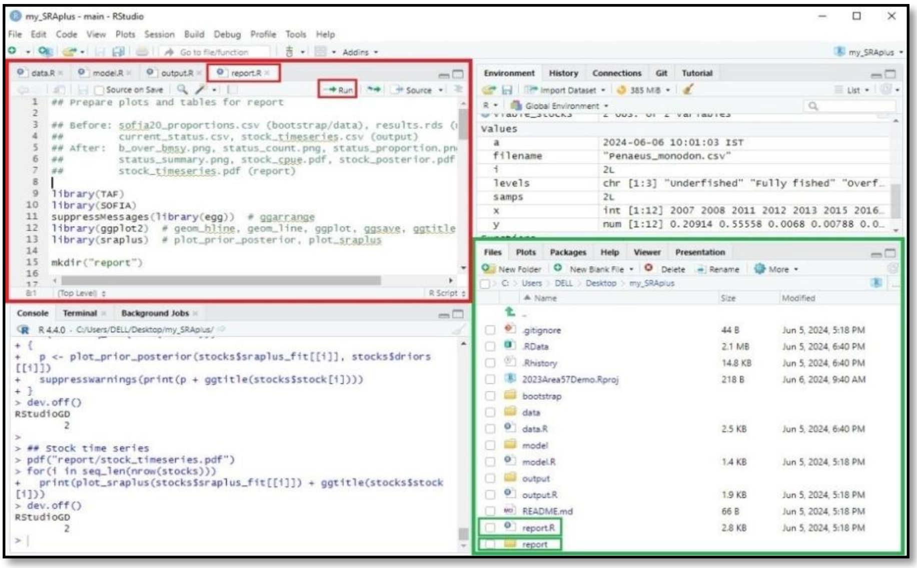
If the report.R is executed without installing the 'egg' R package, then the following error will pop up: Error in library(egg): there is no package called 'egg'

The error can be circumvented by installing the 'egg' R package using the following code:

```
install.packages("egg")
```

Click the report.R (right side bottom window shown inside the green box).

The R codes in the report.R will be visible on the left side top window (shown inside the red box). Select all the codes (click anywhere on the code and press control+A) and Run.



```
## Prepare plots and tables for report

## Before: sofia20_proportions.csv (bootstrap/data), results.rds (
##         current_status.csv, stock_timeseries.csv (output)
## After:  b_over_bmsy.png, status_count.png, status_proportion.pn
##         status_summary.png, stock_cpue.pdf, stock_posterior.pdf
##         stock_timeseries.pdf (report)

library(TAF)
library(SOFIA)
suppressMessages(library(egg))  # ggarrange
library(ggplot2)  # geom_hline, geom_line, ggplot, ggsave, ggtitle
library(sraplus)  # plot_prior_posterior, plot_sraplus

mkdir("report")
```

```
R 4.4.0 · C:/Users/DELL/Desktop/my_SRAplus/
+ {
+   p <- plot_prior_posterior(stocks$sraplus_fit[[i]], stocks$driors
[[i]])
+   suppresswarnings(print(p + ggtitle(stocks$stock[i])))
+ }
> dev.off()
RStudioGD
        2
>
> ## Stock time series
> pdf("report/stock_timeseries.pdf")
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]]) + ggtitle(stocks$stock
[i]))
> dev.off()
RStudioGD
        2
> |
```

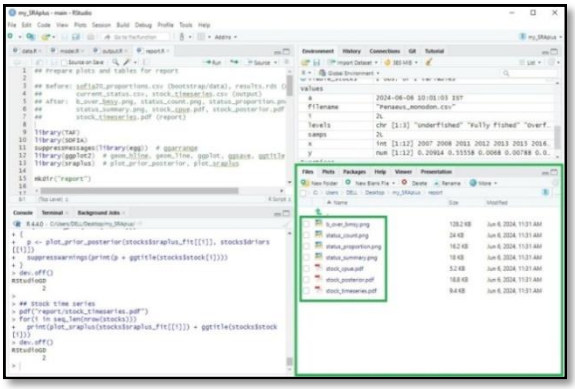
Values	
a	2024-06-06 10:01:03 IST
filename	"Penaeus_monodon.csv"
i	2L
levels	chr [1:3] "Underfished" "Fully fished" "Overf…
samps	2L
x	int [1:12] 2007 2008 2011 2012 2013 2015 2016…
y	num [1:12] 0.20914 0.55558 0.0068 0.00788 0.0…

C: › Users › DELL › Desktop › my_SRAplus

Name	Size	Modified
.gitignore	44 B	Jun 5, 2024, 5:18 PM
.RData	2.1 MB	Jun 5, 2024, 6:40 PM
.Rhistory	14.8 KB	Jun 5, 2024, 6:40 PM
2023Area57Demo.Rproj	218 B	Jun 6, 2024, 9:40 AM
bootstrap		
data		
data.R	2.5 KB	Jun 5, 2024, 6:40 PM
model		
model.R	1.4 KB	Jun 5, 2024, 5:18 PM
output		
output.R	1.9 KB	Jun 5, 2024, 5:18 PM
README.md	66 B	Jun 5, 2024, 5:18 PM
report.R	2.8 KB	Jun 5, 2024, 5:18 PM
report		

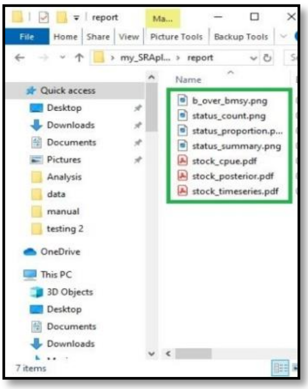
A report folder will be created (right side bottom window shown inside the green box). Click it to view the graphical outputs (bbmsy.png, status_by_year.png, status_sofia.png, status_sraplus.png, stock_cpue.pdf, stock_posterior.pdf, stock_timeseries.pdf) in the RStudio. These outputs can also be seen inside the report subfolder inside the my_SRAplus folder created on the Desktop.

RStudio view of report



b_over_bmsy.png
status_count.png
status_proportion.png
status_summary.png
stock_cpue.pdf
stock_posterior.pdf
stock_timeseries.pdf

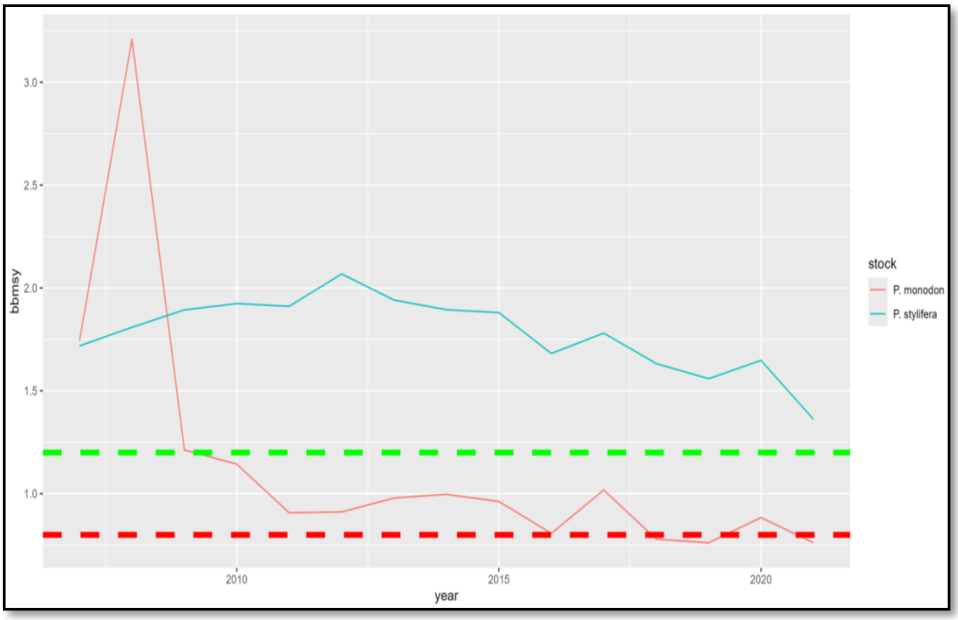
Windows view of report



Name
b_over_bmsy.png
status_count.png
status_proportion.p...
status_summary.png
stock_cpue.pdf
stock_posterior.pdf
stock_timeseries.pdf

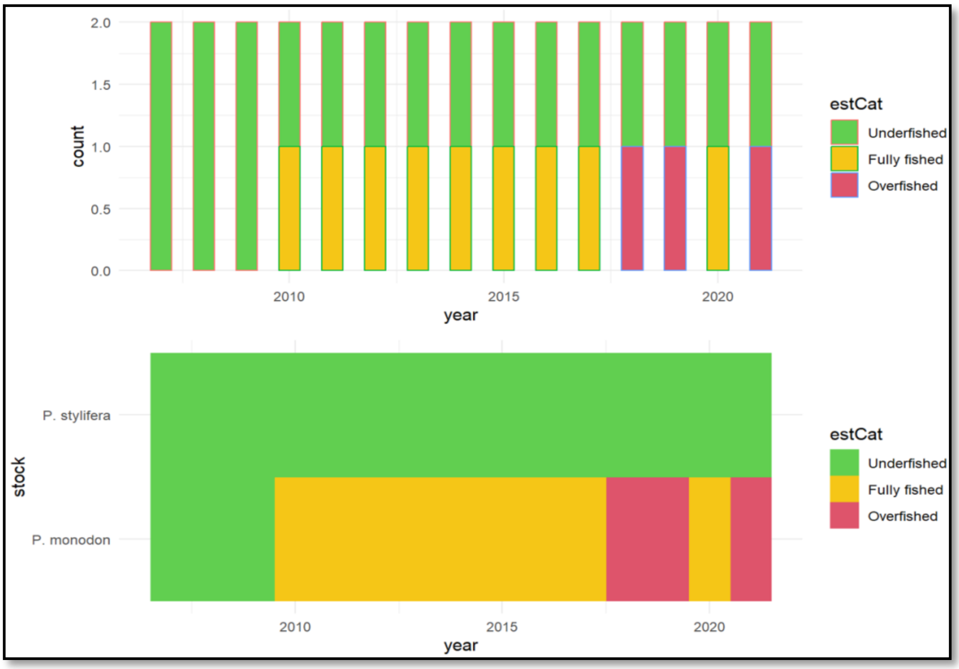
Clicking the individual items inside the report subfolder will produce the following graphical outputs.

b_over_bmsy.png



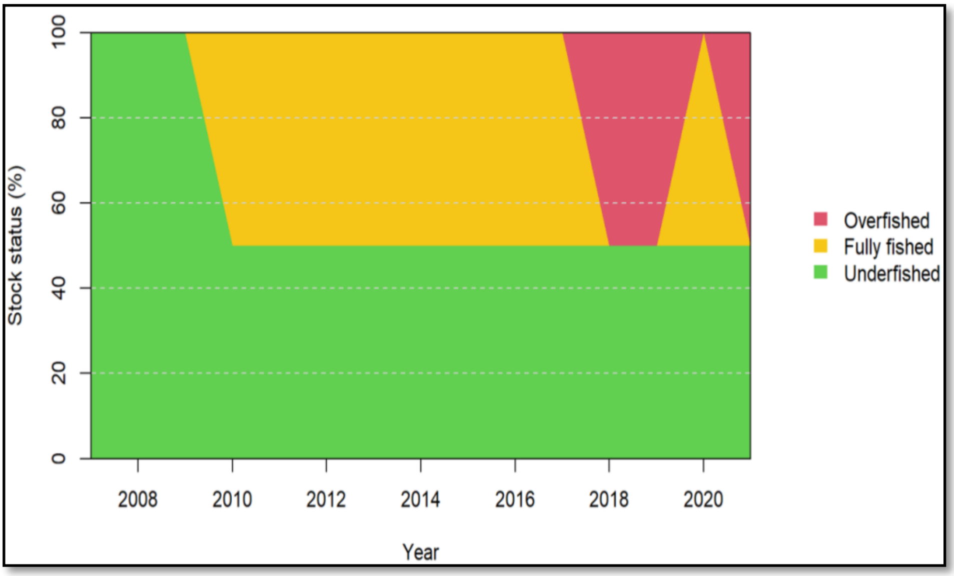
A times-series of B/Bmsy status for the analyzed stocks.

status_count.png



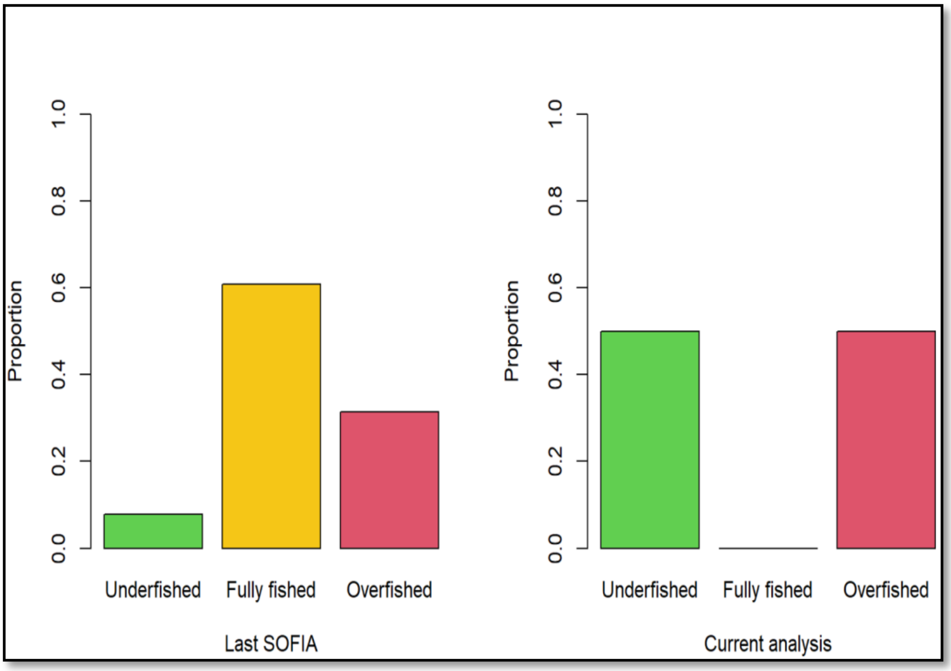
A times-series of stock health status (underfished or fully fished or overfished) in terms of count for the analyzed stocks.

status_proportion.png



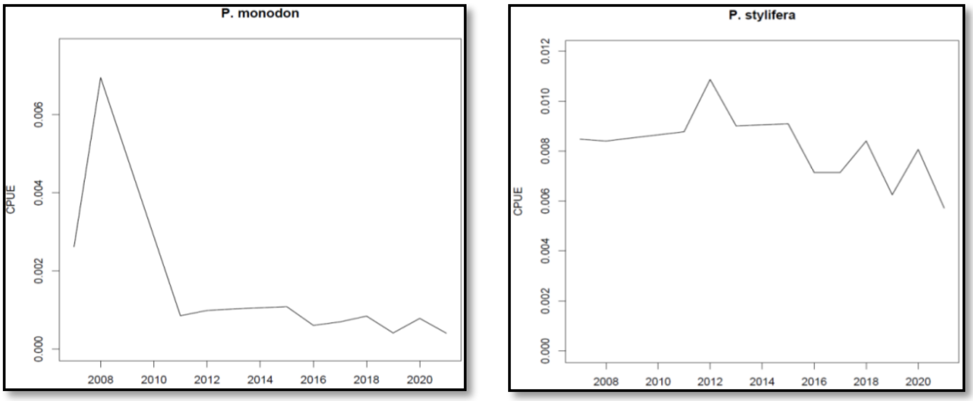
A times-series of stock health status (underfished or fully fished or overfished) in terms of proportion for the analyzed stocks.

status_summary.png



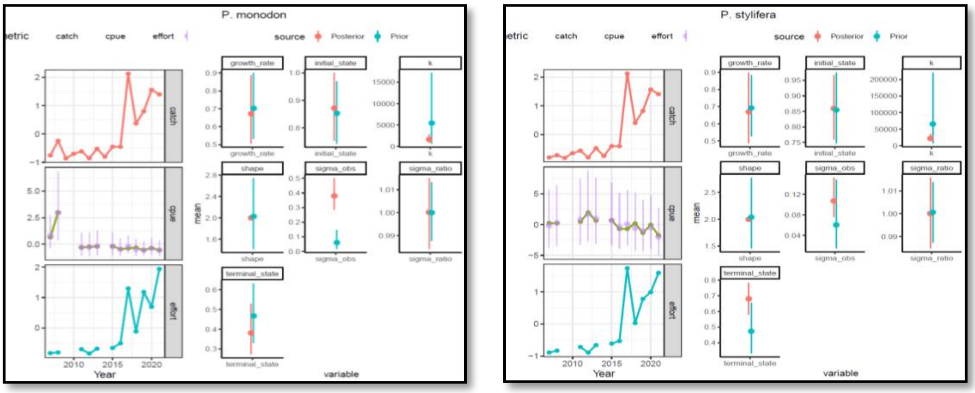
A comparison of stock health status (last assessment vs. current assessment) of the analyzed stocks in terms of proportion.

stock_cpue.pdf



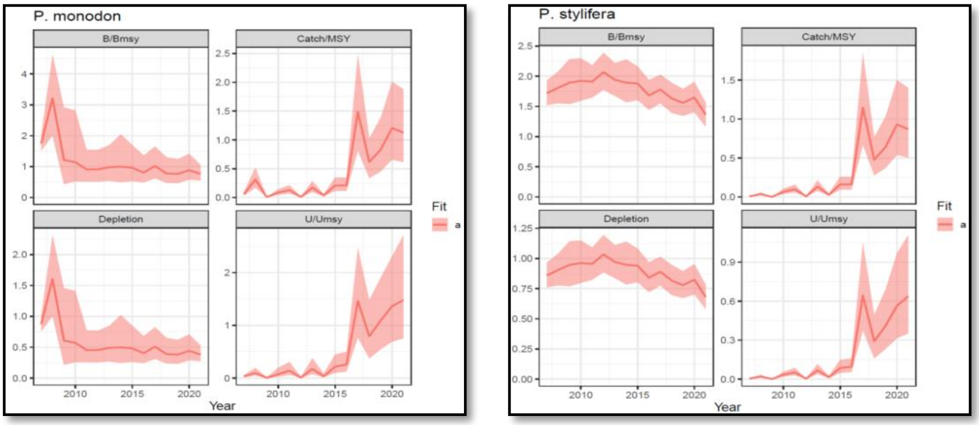
A pictorial demonstration of time-series of CPUE for the analyzed stocks.

stock_posterior.pdf



A comparison between the priors and posteriors of variables used in the sraplus modeling of the analyzed stocks.

stock_timeseries.pdf



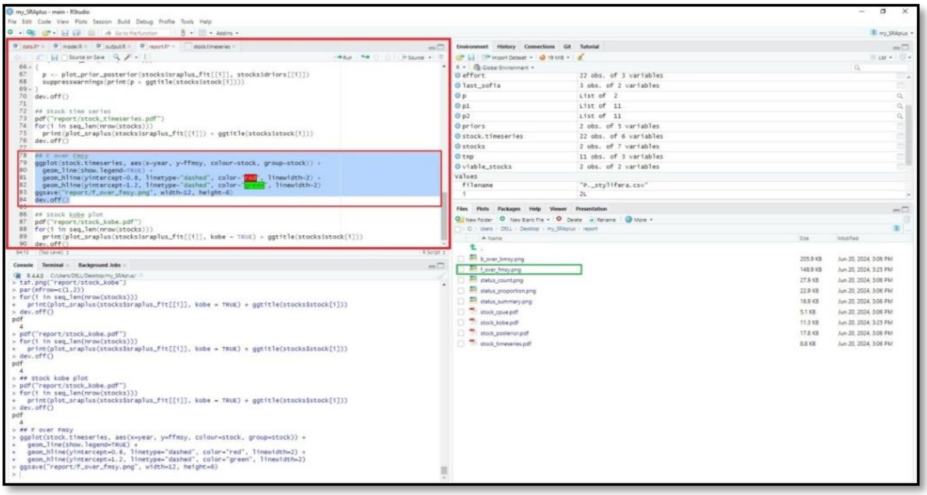
A times-series of B/Bmsy, F/Fmsy (U/Umsy), Catch/MSY and Depletion status for the analyzed stocks.

Additional outputs

F/F$_{msy}$ plot

Additional outputs, such as F/Fmsy plot, can be plotted using the following codes. Copy and paste the following codes below the last line of codes in the report.R (R-script file) to produce an additional graph on F/Fmsy:

```
## F over Fmsy
ggplot(stock.timeseries, aes(x=year, y=ffmsy, colour=stock, group=stock)) +
  geom_line(show.legend=TRUE) +
  geom_hline(yintercept=0.8, linetype="dashed", color="red", linewidth=2) +
  geom_hline(yintercept=1.2, linetype="dashed", color="green", linewidth=2)
ggsave("report/f_over_fmsy.png", width=12, height=6)
dev.off()
```


my_SRAplus - main - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
data.R*
model.R
output.R
report.R*
stock.timeseries
Source on Save
Run
Source
Console
Terminal
Background Jobs
R 4.4.0 · C:/Users/DELL/Desktop/my_SRAplus/
Environment History Connections Git Tutorial
Import Dataset
19 MiB
Global Environment
List
Files Plots Packages Help Viewer Presentation
New Folder
New Blank File
Delete
Rename
More
C > Users > DELL > Desktop > my_SRAplus > report
my_SRAplus

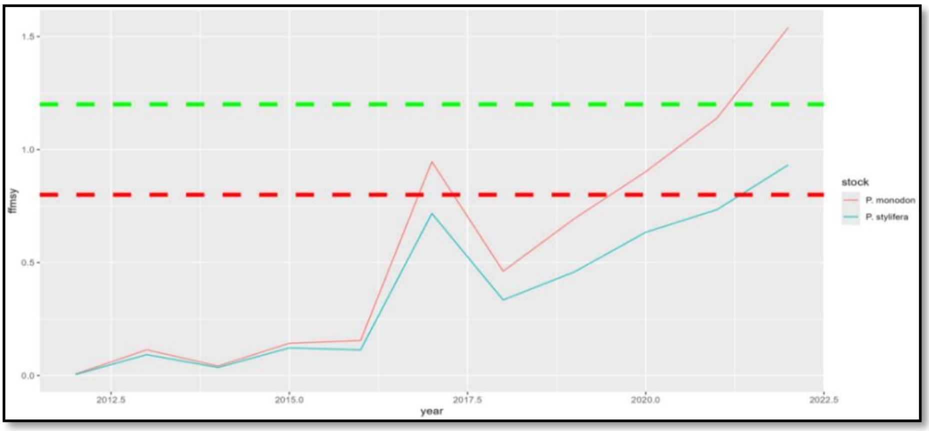
```
{
  p <- plot_prior_posterior(stocks$sraplus_fit[[i]], stocks$driors[[i]])
  suppressWarnings(print(p + ggtitle(stocks$stock[i])))
}
dev.off()

## stock time series
pdf("report/stock_timeseries.pdf")
for(i in seq_len(nrow(stocks)))
  print(plot_sraplus(stocks$sraplus_fit[[i]]) + ggtitle(stocks$stock[i]))
dev.off()

## F over Fmsy
ggplot(stock.timeseries, aes(x=year, y=ffmsy, colour=stock, group=stock)) +
  geom_line(show.legend=TRUE) +
  geom_hline(yintercept=0.8, linetype="dashed", color="red", linewidth=2) +
  geom_hline(yintercept=1.2, linetype="dashed", color="green", linewidth=2)
ggsave("report/f_over_fmsy.png", width=12, height=6)
dev.off()

## stock kobe plot
pdf("report/stock_kobe.pdf")
for(i in seq_len(nrow(stocks)))
  print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
dev.off()
```

```
> taf.png("report/stock_kobe")
> par(mfrow=c(1,2))
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
> dev.off()
pdf
  4
> pdf("report/stock_kobe.pdf")
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
> dev.off()
pdf
  4
> ## stock kobe plot
> pdf("report/stock_kobe.pdf")
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
> dev.off()
pdf
  4
> ## F over Fmsy
> ggplot(stock.timeseries, aes(x=year, y=ffmsy, colour=stock, group=stock)) +
+   geom_line(show.legend=TRUE) +
+   geom_hline(yintercept=0.8, linetype="dashed", color="red", linewidth=2) +
+   geom_hline(yintercept=1.2, linetype="dashed", color="green", linewidth=2)
> ggsave("report/f_over_fmsy.png", width=12, height=6)
> 
```

effort	22 obs. of 3 variables
last_sofia	3 obs. of 2 variables
p	List of 2
p1	List of 11
p2	List of 11
priors	2 obs. of 5 variables
stock.timeseries	22 obs. of 6 variables
stocks	2 obs. of 7 variables
tmp	11 obs. of 3 variables
viable_stocks	2 obs. of 2 variables
Values	
filename	"P._stylifera.csv"
i	2L

Name	Size	Modified
..		
b_over_bmsy.png	205.9 KB	Jun 20, 2024, 3:06 PM
f_over_fmsy.png	148.9 KB	Jun 20, 2024, 3:25 PM
status_count.png	27.9 KB	Jun 20, 2024, 3:06 PM
status_proportion.png	22.9 KB	Jun 20, 2024, 3:06 PM
status_summary.png	16.9 KB	Jun 20, 2024, 3:06 PM
stock_cpue.pdf	5.1 KB	Jun 20, 2024, 3:06 PM
stock_kobe.pdf	11.3 KB	Jun 20, 2024, 3:25 PM
stock_posterior.pdf	17.8 KB	Jun 20, 2024, 3:06 PM
stock_timeseries.pdf	8.8 KB	Jun 20, 2024, 3:06 PM

f_over_fmsy.png

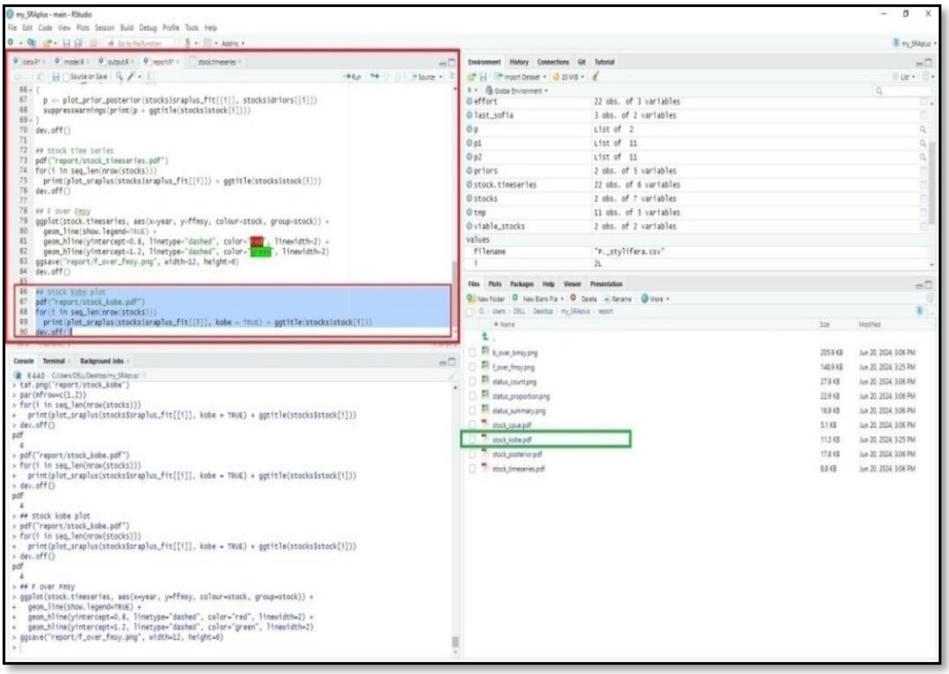


A times-series of F/Fmsy status for the analysed stocks.

Kobe plot

Additional outputs, such as Kobe plot, can be plotted using the following codes. Copy and paste the following codes below the last line of codes in the report.R (R-script file) to produce an additional Kobe plot.

```
## Stock kobe plot
pdf("report/stock_kobe.pdf")
for(i in seq_len(nrow(stocks)))
  print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
dev.off()
```



```
{
  p <- plot_prior_posterior(stocks$sraplus_fit[[i]], stocks$priors[[i]])
  suppressWarnings(print(p + ggtitle(stocks$stock[i])))
}
dev.off()

## stock time series
pdf("report/stock_timeseries.pdf")
for(i in seq_len(nrow(stocks)))
  print(plot_sraplus(stocks$sraplus_fit[[i]]) + ggtitle(stocks$stock[i]))
dev.off()

## F over Fmsy
ggplot(stock.timeseries, aes(x=year, y=ffmsy, colour=stock, group=stock)) +
  geom_line(show.legend=TRUE) +
  geom_hline(yintercept=0.8, linetype="dashed", color="red", linewidth=2) +
  geom_hline(yintercept=1.2, linetype="dashed", color="green", linewidth=2)
ggsave("report/f_over_fmsy.png", width=12, height=6)
dev.off()

## stock kobe plot
pdf("report/stock_kobe.pdf")
for(i in seq_len(nrow(stocks)))
  print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
dev.off()
```

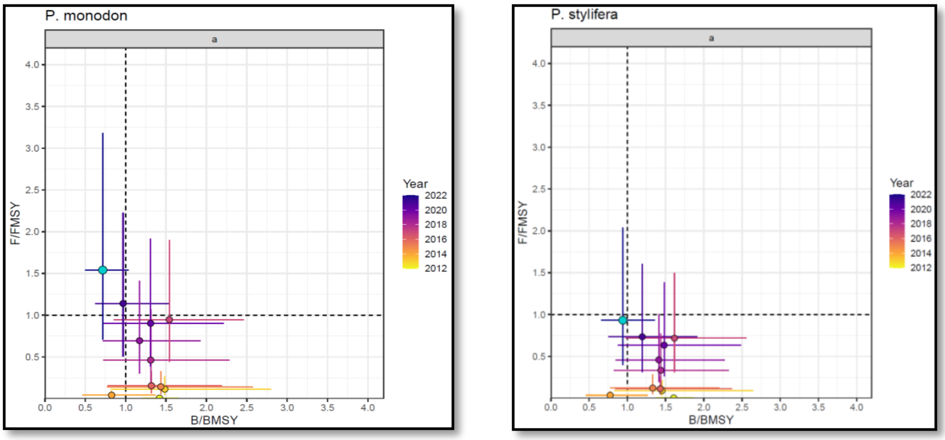
```
R 4.4.0 · C:/Users/DELL/Desktop/my_SRAplus/
> taf.png("report/stock_kobe")
> par(mfrow=c(1,2))
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
> dev.off()
pdf 
  4 
> pdf("report/stock_kobe.pdf")
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
> dev.off()
pdf 
  4 
> ## stock kobe plot
> pdf("report/stock_kobe.pdf")
> for(i in seq_len(nrow(stocks)))
+   print(plot_sraplus(stocks$sraplus_fit[[i]], kobe = TRUE) + ggtitle(stocks$stock[i]))
> dev.off()
pdf 
  4 
> ## F over FMSY
> ggplot(stock.timeseries, aes(x=year, y=ffmsy, colour=stock, group=stock)) +
+   geom_line(show.legend=TRUE) +
+   geom_hline(yintercept=0.8, linetype="dashed", color="red", linewidth=2) +
+   geom_hline(yintercept=1.2, linetype="dashed", color="green", linewidth=2)
> ggsave("report/f_over_fmsy.png", width=12, height=6)
> 
```

effort	22 obs. of 3 variables
last_sofia	3 obs. of 2 variables
p	List of 2
p1	List of 11
p2	List of 11
priors	2 obs. of 5 variables
stock.timeseries	22 obs. of 6 variables
stocks	2 obs. of 7 variables
tmp	11 obs. of 3 variables
viable_stocks	2 obs. of 2 variables
Values	
filename	"P._stylifera.csv"
i	2L

C: › Users › DELL › Desktop › my_SRAplus › report

Name	Size	Modified
..		
b_over_bmsy.png	205.9 KB	Jun 20, 2024, 3:06 PM
f_over_fmsy.png	148.9 KB	Jun 20, 2024, 3:25 PM
status_count.png	27.9 KB	Jun 20, 2024, 3:06 PM
status_proportion.png	22.9 KB	Jun 20, 2024, 3:06 PM
status_summary.png	16.9 KB	Jun 20, 2024, 3:06 PM
stock_cpue.pdf	5.1 KB	Jun 20, 2024, 3:06 PM
stock_kobe.pdf	11.3 KB	Jun 20, 2024, 3:25 PM
stock_posterior.pdf	17.8 KB	Jun 20, 2024, 3:06 PM
stock_timeseries.pdf	8.8 KB	Jun 20, 2024, 3:06 PM

Kobe plot



Note: *If one species or more than one species are there for which the effort levels are the same (Refer to the Catch and effort file types), then the same.effort=TRUE should be kept unchanged and TRUE in Line 60 of data.R (shown inside the blue box). If more than one species are there for which the efforts are different (Refer to Catch and effort file types), then the same.effort=TRUE should be changed to same.effort=FALSE in Line 60 of data.R (shown inside the blue box).*

References

Journal articles

Ovando, D., Hilborn, R., Monnahan, C., Rudd, M., Sharma, R., Thorson, J. T., Rousseau, Y., & Ye, Y. (2021). Improving estimates of the state of global fisheries depends on better data. Fish and Fisheries, 22(6), 1377–1391. https://doi.org/10.1111/faf.12593

Online resources

GitHub repository: Ovando, D. (n.d.). sraplus [GitHub repository]. GitHub. https://github.com/DanOvando/sraplus

Example data file download link

Use the following Google Drive link to download the example data file. The user may work with this file and modify it according to the data at hand.

https://drive.google.com/drive/folders/1cNNiT5t3vr1MRpK69vygENsQdqwn-OJD?usp=sharing

भाकृअनुप
ICAR

सी एम एफ आर आइ
CMFRI

ISBN 978-93-82263-67-8

Indian Council of Agricultural Research
Central Marine Fisheries Research Institute
Ernakulam North PO, PB No. 1603, Kochi, India