LIME: An R package for simulation and estimation using length data to account for variable fishing mortality and recruitment

Merrill Rudd

2018-04-18

Contents

Introduction

This package contains functions to run the Length-based Integrated Mixed Effects (LIME) fisheries stock assessment method. The LIME package can be used in two ways:

1. as an age-structured operating model to generate simulated length composition, catch data, and abundance indices based on single or multiple fleets and seasons, and

2. as an estimation model by fitting to length composition data (at a minimum) to estimate annual fishing mortality, annual recruitment, relative spawning biomass, spawning potential ratio (SPR), and other derived outputs of an age-structured model.

Length-based

LIME was developed for data-limited fisheries, where few data are available other than a representative sample of the size structure of the vulnerable portion of the population (i.e., length composition data from the catch) and an understanding of the life history of the species.

Integrated

While length composition from the catch can hold information on fishing mortality, gear selectivity, and recruitment, with process and observation error it can be very difficult to tease apart the patterns and biological mechanisms from the noise. With additional information, such as even a few years of catch trends or an abundance index, analysts can include additional information to the LIME model to help tease apart these processes. For example, catch data is much more informative of the fishing mortality rate over time than the length composition data. Catch used alongside length composition data can help decrease bias and uncertainty in estimates of fishing mortality rates.

LIME also includes all years of data in the same analysis with a penalty on the annual fishing mortality rate to keep it from increasing or decreasing unrealistically between years. However, LIME assumes the selectivity curve is constant over time for each fleet.

Mixed effects

LIME relaxes the equilibrium assumptions of other length-based methods by estimating annual fishing mortality and recruitment variation (among other parameters), deriving annual recruitment as a random effect.

See Rudd and Thorson (2017) in the reference list for full details of the model, including simulation testing to evaluate performance across life history types, population variability scenarios, and data availability scenarios, as well as violations of the model assumptions.

Bug Reports

Please alert me to any bugs or issues by using GitHub.

Comments and suggestions for additional features are welcome and we can discuss via email at mbrudd@uw.edu.

Installation

Installing the Package

1. `devtools`: You can install the development version of the package from GitHub using the `devtools` package:

```
install.packages("devtools", repos='http://cran.us.r-project.org')
```

2. Follow the steps to install TMB:

- Windows: https://github.com/kaskr/adcomp/wiki/Windows-installation
- Linux: https://github.com/kaskr/adcomp (See README)
- Mac: should be the same as Linux, please see contact info under Bug Reports if that doesn't work.

3. Install required package `TMBhelper`:

```
devtools::install_github("kaskr/TMB_contrib_R/TMBhelper")
```

4. Install LIME:

```
devtools::install_github("merrillrudd/LIME", dependencies=TRUE, ref="multifleet")
```

The installation may produce error messages requiring other packages. You can download those from CRAN if they are not automatically downloaded in the LIME installation process, and then start from the step that produced the error. Again, please use the contact info in Bug Reports if this doesn't work, as you may have discovered a new installation error... congratulations!

Load the Package

```
library(LIME)
```

Step 1: Specify biological inputs and starting values

The LIME package simulation and estimation features require the biological inputs (e.g. growth, natural mortality, maturity) and starting values for selectivity parameters in a list. Using `create_lh_list` will make sure all required elements are included within the list and in the required format. The `create_lh_list` function requires inputs for some parameters and includes default values for others.

Minimum inputs for `create_lh_list`

The user is required to input the following parameters to `create_lh_list` for simulation and estimation:

Minimum inputs: Biology

- `linf`: von Bertalanffy asymptotic length
- `vbk`: von Bertalanffy growth coefficient
- `M`: Annual natural mortality rate
- `lwa`: Length-weight scaling parameter
- `lwb`: Length-weight allometric parameter
- `M50` Length or age at 50% maturity
- `maturity_input`: Whether `M50` input is in "length" or "age"

Minimum inputs: Exploitation

- `S50`: Length or age at 50% selectivity
- `S95`: Length or age at 95% selectivity
- `selex_input`: Whether `S50` and `S95` are in "length" or "age"

Other inputs for `create_lh_list` with default settings

Other inputs: Biology

- `M95`: Length or age at 95% maturity; default=NULL for one-parameter logistic maturity. `M95` will use two-parameter logistic maturity curve. It is assumed `M95` should be input as length if `M50` is length, same for age.
- `t0`: Length at age=0; Default=-0.01, should be adjusted based on local studies or meta-analysis.
- `R0`: Equilibrium recruitment; Default=1.0. Changing `R0` will change the scale of the population size. It is recommended to set an initial value of `R0` more appropriate to the expected population size if using catch data to estimate population size with LIME.
- `h`: Steepness; Default=1.0 such that the expected recruitment calculated in the Beverton-Holt stock-recruit curve is not affected by the level of spawning biomass. Setting `h` below 1.0 will lead to the expected annual recruitment as a function of the spawning biomass. LIME will still calculate recruitment deviates around this expected value of annual recruitment.
- `AgeMax`: Maximum age; Default=-log(0.01)/M, the age at which 1% of the population is still alive based on the specified natural mortality rate `M`. This age can be adjusted based on local evidence of maximum age.
- `rho`: Recruitment autocorrelation, used in simulation only; Default=0 (no recruitment autocorrelation). Changing the value of `rho` will add autocorrelation to the generated recruitment time series for a simulation study.

Other inputs: Exploitation

- `S95`: Length or age at 95% selectivity; Default=`NULL` for one-parameter logistic selectivity; specifying `S95` will use two-parameter logistic selectivity curve. It is assumed `S95` should be input as length if `S50` is length, same for age.
- `selex_type`: Selectivity function type; Default=`logistic` using a one-parameter logistic selectivity curve if `S95` is not specified, or a two-parameter logistic curve if `S95` is specified. Alternate option=`dome` to generate a population exploited based on a dome-shaped selectivity curve. Dome-shaped selectivity cannot currently be estimated in LIME, but specifying dome-shaped selectivity in the `create_lh_list` function can generate an assumed dome-shaped curve to be fixed when using LIME for estimation.
- `dome_sd`: Dome-shaped selectivity right-hand standard deviation; Default=`NULL` for logistic selectivity, must be specified if `selex_type="dome"`. The standard deviation of the normal distribution for the right side of the selectivity curve (after the logistic curve specified reaches 100% selectivity).
- `Fequil`: The proportion of spawning biomass relative to unfished spawning biomass at bioeconomic equilibrium, default=0.5.
- `Frate`: exponent determining the strength of coupling between effort and changes in biomass, default=0.2.
- `qcoef`: Catchability coefficient; default=1e-5. Estimated when using an abundance index, starting value should be adjusted to a reasonable number based on the abundance index and starting value for the scaling parameter `R0`.
- `theta`: Dirichlet-multinomial parameter; Default=1, a relative small value. With a larger `theta` (e.g. 10) there will be no variance inflation where the effective sample size will approach the input sample size (Thorson et al. 2017)
- `nseasons`: Number of seasons in a year; Default=1 for annual length composition data and annual growth and mortality parameters. For short-lived species, it is helpful to use a shorter-than-annual time step to account for the growth of short-lived fish during one year, if length data is collected on time steps less than one year (e.g. month, `nseasons`=12).
- `nfleets`: Number of fleets to be modeled in the fishery; default=1 for a single fleet. With `nfleets` greater than 1, the number of `S50`, `S95`, and `selex_type` values must match the value for `nfleets`. Example, `S50`=c(20,26), `S95`=c(25,35), `selex_type`=rep('logistic', `nfleets`)

Other inputs: Variation

- `CVlen`: Coefficient of variation around the growth curve; Default=0.1; Should be adjusted based on the expected variability in the age-length curve.
- `SigmaR`: Recruitment standard deviation; Default=0.737, the median across all fish species (Thorson et al. 2014); Used for generating recruitment deviates in the simulation or as a starting value for its estimation using LIME.

- `SigmaF`: Fishing mortality standard deviation; Default=0.2; Used for generating fishing mortality deviates in the simulation or as the standard deviation of the fishing mortality penalty when estimating annual fishing mortality using LIME.
- `SigmaC`: Catch standard deviation; Default=0.2; Used as the fixed standard deviation in the lognormal likelihood when fitting LIME to catch data.
- `SigmaI`: Index standard deviation; Default=0.2; Used as the fixed standard deviation in the lognormal likelihood when fitting LIME to index data.

Now let's populate the `create_lh_list` function with example values.

```
lh <- create_lh_list(vbk = 0.21, linf = 65, t0 = -0.01, lwa = 0.0245, lwb = 2.79,
    S50 = c(20), S95 = c(26), selex_input = "length", selex_type = c("logistic"),
    M50 = 34, maturity_input = "length", M = 0.27, binwidth = 1, CVlen = 0.1,
    SigmaR = 0.737, SigmaF = 0.2, SigmaC = 0.1, SigmaI = 0.1, R0 = 1, Frate = 0.1,
    Fequil = 0.25, qcoef = 1e-05, start_ages = 0, rho = 0.43, theta = 10, nseasons = 1,
    nfleets = 1)
```

Plot the biological and selectivity inputs

Now we should check out the biological parameters and selectivity we've created. Note that even though we input maturity and selectivity by length, `create_lh_list` converts to age and outputs both selectivity-at-age by fleet (`lh$S_fa`) and selectivity-at-length by fleet (`lh$S_fl`).

```
par(mfrow=c(2,2), mar=c(4,4,3,1))
plot(lh$L_a, type="l", lwd=4, xlab="Age", ylab="Length (cm)")
plot(lh$W_a, type="l", lwd=4, xlab="Age", ylab="Weight (g)")
plot(lh$Mat_l, type="l", lwd=4, xlab="Length (cm)", ylab="Proportion mature")

# plot selectivity for the first (and only) fleet (first row)
plot(lh$S_fl[1,], type="l", lwd=4, xlab="Length (cm)", ylab="Proportion selected to gear")
```

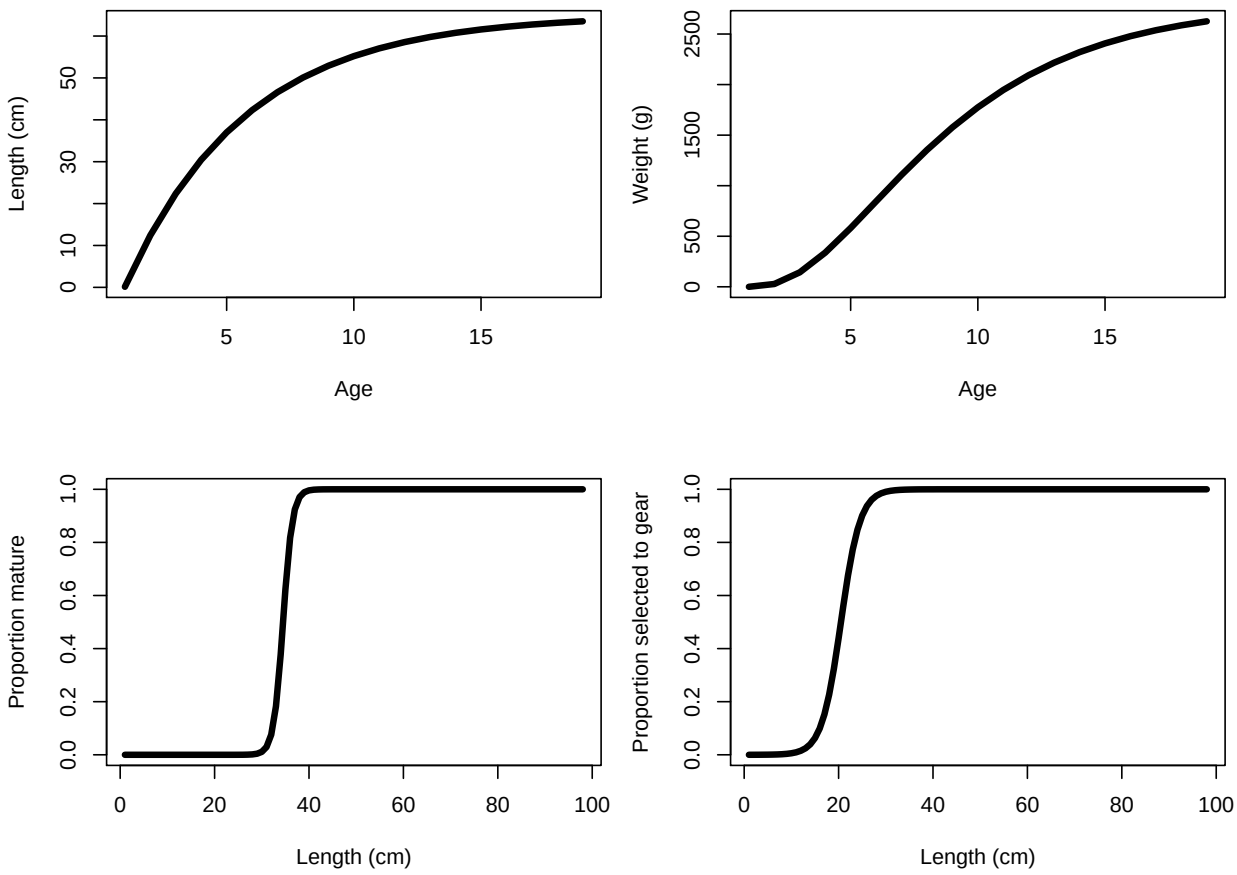


Figure 1: Example length-at-age, weight-at-age, maturity-at-length, and selectivity-at-length.

Check time step regarding predicted growth

Particularly for short-lived fish, it is possible that an annual time step is too coarse of a time scale to capture individual growth between years. For example, if a fish grows rapidly between ages 1 and 2, it is possible the probability of being a length given age will result in a negligible probability of being certain lengths. However, it is likely those lengths will be observed, and in this case LIME will not be able to fit the data well.

To check for this issue, we recommend plotting the probability of being in a length bin given age to make sure there is greater than negligible probability of observing all lengths. In this case, there is a small overlap between the distributions for age 1 and 2 fish, which should be enough to have an adequate model fit if fish are observed at approximately 17 cm.

For tips on what to do when there is a negligible probability of observing some lengths, see the "Format data" section in this document.

```
plba <- with(lh, age_length(highs, lows, L_a, CVlen))
```

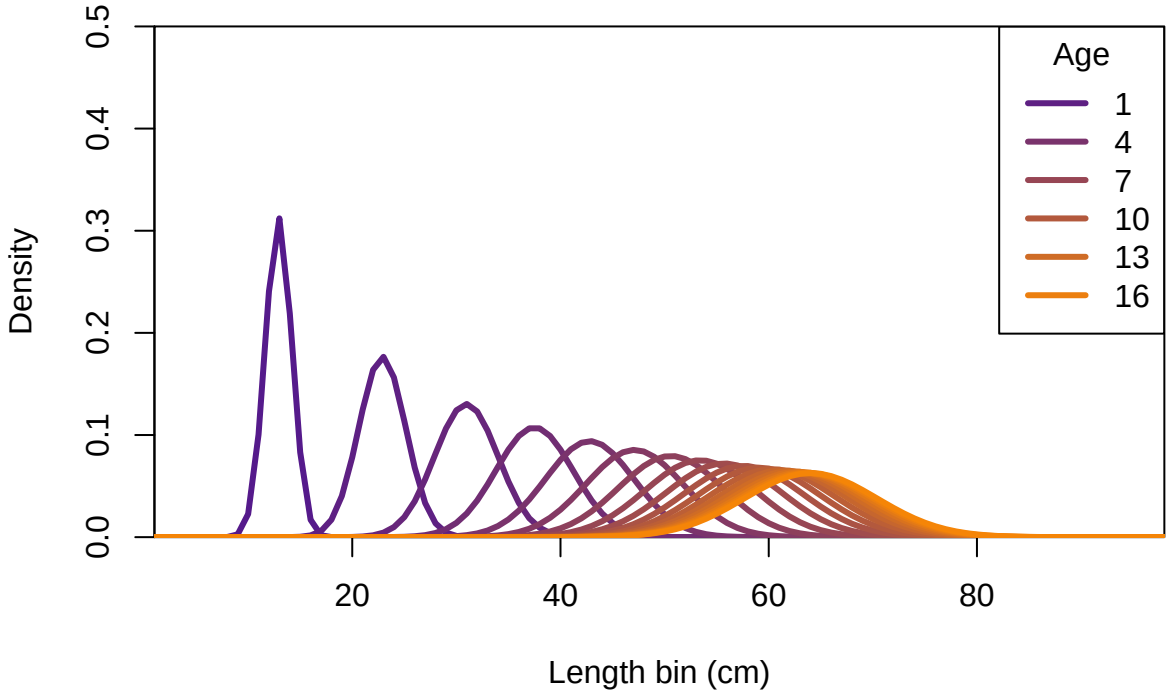


Figure 2: Probability of being in a length bin given age (years).

Step 2: Data inputs to LIME

The first subsection on data formatting provides an overview of the simulation functions within the LIME package. To learn how to format data from your fishery, skip to the "Format data" section.

Simulating data

Using the life history list output from `create_lh_list`, we can simulate a population and generate data.

The `generate_data` function has several settings for which the user is required to input a value:

- `modpath`: model path for saving simulated populations; can set as `NULL` to run within the R environment only without saving locally.
- `itervec`: vector of iterations of simulated data; can be 1 to run only one iteration, or 1:100 to run 100 iterations of simulated populations.
- `lh`: life history list, output from `create_lh_list`.
- `Fdynamics`: Pattern for fishing mortality dynamics; options include `Constant`, `Oneway`, `Endogenous`, or `None`.
- `Rdynamics`: Pattern for recruitment dynamics; options include `Constant`, or `Pulsed`.
- `Nyears`: number of years for your simulated population.
- `Nyears_comp`: number of years to generate length data; e.g. if 10 years and `Nyears` is 20 years, will be the last 10 years in the 20 year time series. Must correspond to the number of fleets, e.g. c(20,10)

would mean 20 years of length data for fleet 1 and 10 years of length data for fleet 2.

- `comp_sample`: nominal sample size of length data annually; e.g. 200 length measurements collected annually. If there is more than 1 fleet and only 1 number is specified, that number will apply to all fleets (e.g. 200 samples in each fleet).
- `init_depl`: initial depletion, or the proportion of the unfished population biomass in the first year of the population to be modeled. Specifying a single value indicates the initial depletion (e.g. 0.5) or two values indicates the lower and upper bounds of a uniform distribution for which the population simulation will randomly draw a depletion value (e.g. c(0.1,0.9)).
- `seed`: set a seed for random numbers for generating process deviations.

There are also several other settings not required but may be useful:

- `rewrite`: default=TRUE to always re-simulated data. `rewrite`=FALSE may be useful to only simulate a new population if the `True.rds` output file already exists in the folder.
- `pool`: default=TRUE, meaning that if the number of seasons per year as specified in the `create_lh_list` function is more than 1, the length composition data collected in each time set is pooled together annually. `pool`=FALSE would mean the generated length composition data is on a time step shorter than 1 year (e.g. monthly if `nseasons`=12). The total sample size for the year will still be equal to `comp_sample`.

Now let's use `generate_data` to simulate 1 example population (itervec=1) to generate length, catch, and an abundance index. We won't specify a `modpath` so the simulated population will just be used locally in this vignette.

```
true <- generate_data(modpath = NULL, itervec = 1, Fdynamics = c("Endogenous"),
    Rdynamics = "Constant", lh = lh, Nyears = 20, Nyears_comp = c(20), comp_sample = 200,
    init_depl = 0.7, seed = 123, fleet_proportions = 1)
```

```
par(mfrow = c(3, 2))
plot(true$F_ft[1, ], type = "l", lwd = 4, xlab = "Year", ylab = "Fishing mortality",
    ylim = c(0, max(true$F_ft) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5)
plot(true$R_t, type = "l", lwd = 4, xlab = "Year", ylab = "Recruitment", ylim = c(0,
    max(true$R_t) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5, cex.lab = 1.5)
plot(true$SPR_t, type = "l", lwd = 4, xlab = "Year", ylab = "SPR", ylim = c(0,
    1), xaxs = "i", yaxs = "i", cex.axis = 1.5, cex.lab = 1.5)
plot(true$D_t, type = "l", lwd = 4, xlab = "Year", ylab = "Relative spawning biomass",
    ylim = c(0, max(true$D_t) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5)
plot(true$Cw_ft[1, ], type = "l", lwd = 4, xlab = "Year", ylab = "Catch", ylim = c(0,
    max(true$Cw_ft) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5, cex.lab = 1.5)
plot(true$I_ft[1, ], type = "l", lwd = 4, xlab = "Year", ylab = "Abundance index",
    ylim = c(0, max(true$I_ft) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5)
```

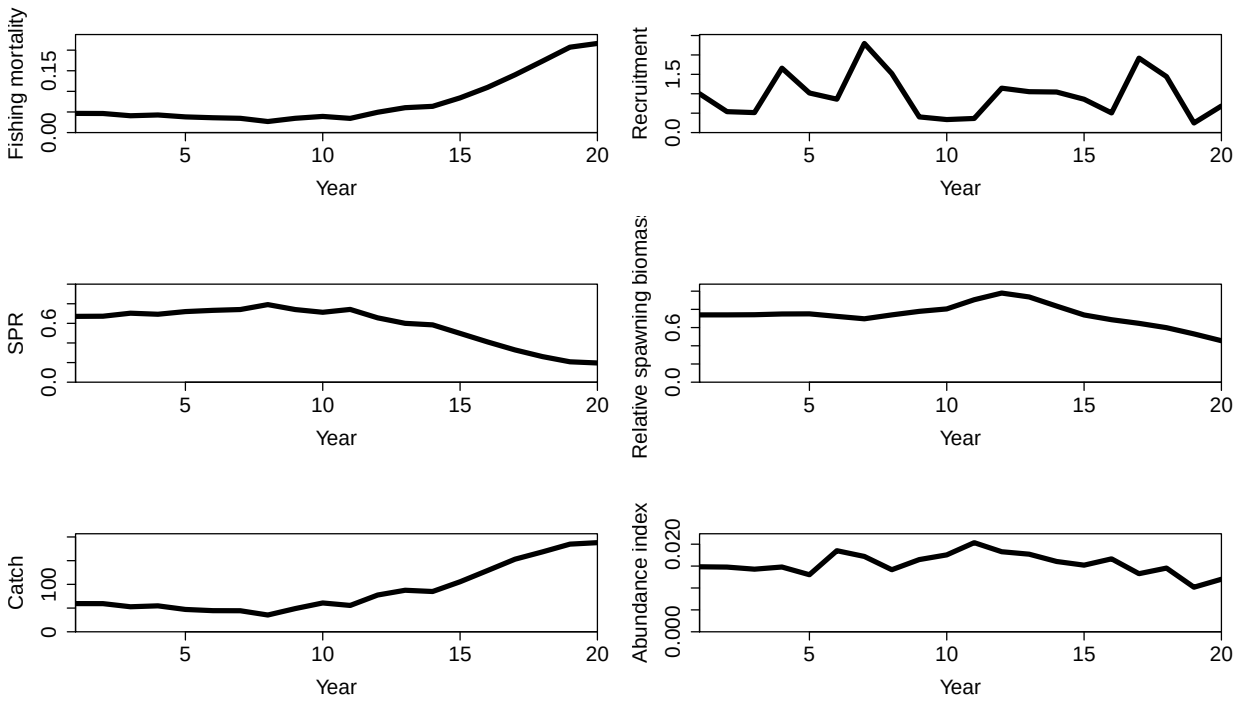


Figure 3: True population: fishing mortality mimicking an open access fishery, where effort is a function of spawning biomass, autocorrelated recruitment, spawning potential ratio (SPR), mean length, spawning biomass, and relative spawning biomass.

The `LIME` package includes the `plot_LCfits` function which can first be used to plot the length frequency data, and later used to plot the model fits to the length frequency data. The `plot_LCfits` function requires the length frequency data in list form, where each element of the list is the length frequency data for a single fleet, structured as a matrix with years along the rows and length bins along the column.

The following code converts the length frequency array `true_LF` output from the `generate_data` function into a list to use in the `plot_LCfits` function:

```
plot_LCfits(LFlist = list(LF = true$LF[, , 1]))
```

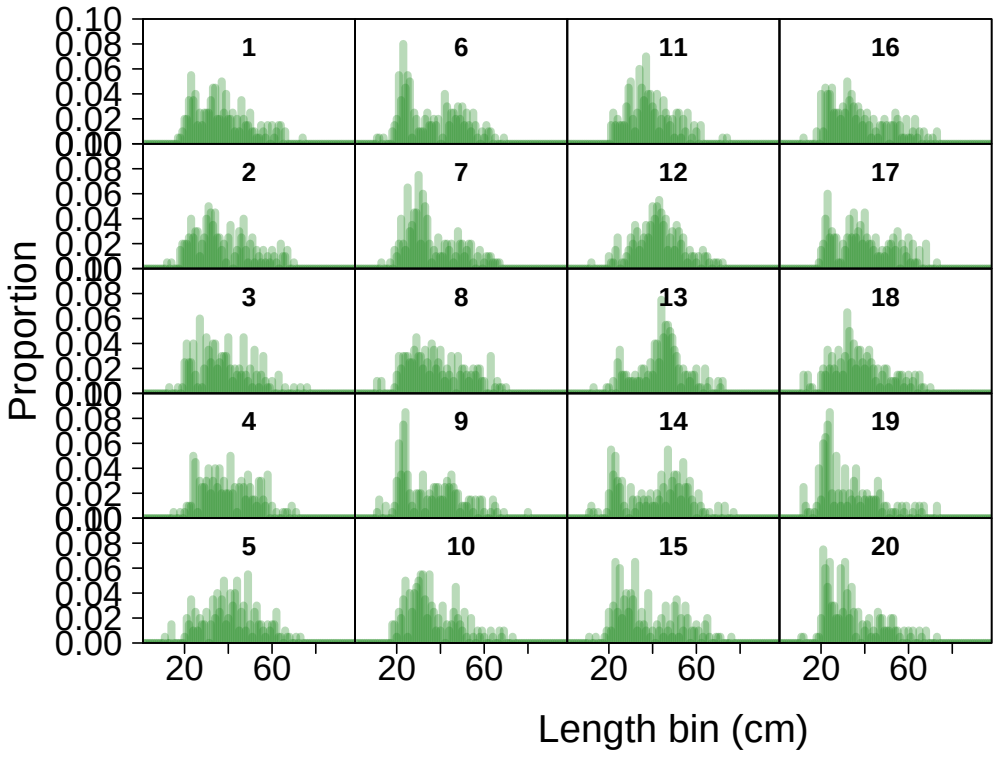


Figure 4: Generated length composition data, labeled by years 1-20.

Format data

LIME requires data input in a specific format:

Length composition data

Option 1: Matrix

With only one fleet, length composition data can be in a length frequency matrix, with years along the rows and length bins along the columns. Here are 5 years as an example:

```
##    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  1  2  4  4  7 11  7  8  3  5
## 2  0 0 0 0 0 0 0 0 0  0  0  1  0  1  0  0  0  3  4  4  4  5  8  5  6  6  2
## 3  0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  1  0  1  5  8  5  5  8  3  1 12
## 4  0 0 0 0 0 0 0 0 0  0  0  0  0  0  1  0  0  1  0  1  2  2  2 10  9  1  6
## 5  0 0 0 0 0 0 0 0 0  0  1  0  0  3  0  0  0  0  0  1  4  3  7  2  5  2  4
##    28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1   3  5  5  5  7  9  9  4  4 10  5  8  4  4  5  2  4  2  7  4  3  3  5  2
## 2   4  5  8 10  9  7  9  5  5  3  4  1  5  7  0  2  3  6  5  8  3  1  4  5
## 3   1  4  9  7  5  8  8  4  5  6  6  6  9  4  2  4  3  4  3  9  3  2  4  1
```

```
## 4  5  5  6  8  4  6  8  5  8  4  6  4  4 10  4  5  1  5  6  5  3  7  5  3
## 5  3  3  5  0  3  7  4  8  5  7 10  3  4  8  6  8 10  2  5  6  2 11  3  1
##    52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## 1  2  1  1  3  2  1  3  0  3  1  2  3  3  0  2  0  0  0  0  0  0  0  1  0
## 2  1  3  2  1  3  1  2  1  3  2  1  0  4  2  2  3  0  0  1  0  0  0  0  0
## 3  7  3  1  3  6  2  1  1  2  0  0  3  0  0  1  0  0  0  1  0  0  1  0  0
## 4  3  2  6  6  3  2  7  0  2  0  1  0  0  1  1  1  0  2  0  1  0  0  0  0
## 5  5  6  3  3  2  1  3  2  3  3  5  1  1  0  2  1  1  0  0  1  0  1  0  0
##    76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
## 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Option 2: Array

The length composition data can be an array of matrices (one for each fleet). Columns should be named with the upper end of the length bin and rows should be named with the year.

One fleet, first five years:

```
## , , 1
##
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  1  2  4  4  7 11  7  8  3  5
## 2 0 0 0 0 0 0 0 0 0  0  0  1  0  1  0  0  0  3  4  4  4  5  8  5  6  6  2
## 3 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  1  0  1  5  8  5  5  8  3  1 12
## 4 0 0 0 0 0 0 0 0 0  0  0  0  0  0  1  0  0  1  0  1  2  2  2 10  9  1  6
## 5 0 0 0 0 0 0 0 0 0  0  1  0  0  3  0  0  0  0  0  1  4  3  7  2  5  2  4
##   28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1  3  5  5  5  7  9  9  4  4 10  5  8  4  4  5  2  4  2  7  4  3  3  5  2
## 2  4  5  8 10  9  7  9  5  5  3  4  1  5  7  0  2  3  6  5  8  3  1  4  5
## 3  1  4  9  7  5  8  8  4  5  6  6  6  9  4  2  4  3  4  3  9  3  2  4  1
## 4  5  5  6  8  4  6  8  5  8  4  6  4  4 10  4  5  1  5  6  5  3  7  5  3
## 5  3  3  5  0  3  7  4  8  5  7 10  3  4  8  6  8 10  2  5  6  2 11  3  1
##   52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## 1  2  1  1  3  2  1  3  0  3  1  2  3  3  0  2  0  0  0  0  0  0  0  1  0
## 2  1  3  2  1  3  1  2  1  3  2  1  0  4  2  2  3  0  0  1  0  0  0  0  0
## 3  7  3  1  3  6  2  1  1  2  0  0  3  0  0  1  0  0  0  1  0  0  1  0  0
## 4  3  2  6  6  3  2  7  0  2  0  1  0  0  1  1  1  0  2  0  1  0  0  0  0
## 5  5  6  3  3  2  1  3  2  3  3  5  1  1  0  2  1  1  0  0  1  0  1  0  0
##   76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
## 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

With two fleets, we add a second matrix to the array. If data are not available in the same years for both fleets, include inclusive years for all fleets with no observations in the years with no data.

Two fleets, first five years, with no data the first two years for the second fleet:

```
## , , 1
##
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
```

```
## 1 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  6  7  9 14  4  3  6  5
## 2 0 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  0  3  2  1  6  8  7  5  7  4
## 3 0 0 0 0 0 0 0 0 0 0  0  0  0  1  1  0  0  2  0  2  1  5  4  5  6  6  6  2
## 4 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  3  1  3  2  7  4  5  2  3
## 5 0 0 0 0 0 0 0 0 0 0  0  0  1  3  1  0  0  0  0  2  3  5  3  7  2  2  3  6
##   28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1  0  6  7  7  6  4  1  5  5  4  5  6  2  4  7  7  3  4  7  6  3  2  5  2
## 2  4  9  7  4  5 11  6  7  4  6  5  8  3  3  2  3  9  3  5  5  2  2  3  2
## 3  7  6  8  4  6  7  6  5  4  2  4  8  7  7  3  2  9  4  4  2  5  3  3  5
## 4  6  2  9  6  6  6  3  6  7  9  2 12  4  9  6  5  7  1  6  4  6  4  3  5
## 5  3  3  1  9  2  2  8  5  4  9  2  5  7  4  5  4  8  3  8  4  5  6  2  1
##   52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## 1  1 11  0  2  5  3  1  3  3  1  1  2  1  1  2  0  0  0  0  0  0  1  0  0
## 2  2  9  7  1  4  2  1  1  2  0  1  2  1  2  0  1  0  0  2  0  0  0  0  0
## 3  3  5  1  5  3  4  3  2  2  0  1  2  0  2  1  1  2  0  0  0  0  1  0  0
## 4  1  5  2  4  6  1  4  3  1  1  0  2  2  1  1  1  0  0  0  0  1  0  0  0
## 5 11  5  5  6  4  6  0  4  3  2  1  0  1  0  2  0  0  1  1  0  0  0  0  0
##   76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
## 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
## , , 2
##
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
## 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Option 2: List

The length composition data could also be a list of matrices (one for each fleet).

Two fleets, first five years:

```
## [[1]]
##    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  6  7  9 14  4  3  6  5
## 2 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  0  3  2  1  6  8  7  5  7  4
## 3 0 0 0 0 0 0 0 0 0  0  0  0  1  1  0  0  2  0  2  1  5  4  5  6  6  6  2
## 4 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  3  1  3  2  7  4  5  2  3
## 5 0 0 0 0 0 0 0 0 0  0  0  1  3  1  0  0  0  0  2  3  5  3  7  2  2  3  6
##    28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1   0  6  7  7  6  4  1  5  5  4  5  6  2  4  7  7  3  4  7  6  3  2  5  2
## 2   4  9  7  4  5 11  6  7  4  6  5  8  3  3  2  3  9  3  5  5  2  2  3  2
## 3   7  6  8  4  6  7  6  5  4  2  4  8  7  7  3  2  9  4  4  2  5  3  3  5
## 4   6  2  9  6  6  6  3  6  7  9  2 12  4  9  6  5  7  1  6  4  6  4  3  5
## 5   3  3  1  9  2  2  8  5  4  9  2  5  7  4  5  4  8  3  8  4  5  6  2  1
##    52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## 1   1 11  0  2  5  3  1  3  3  1  1  2  1  1  2  0  0  0  0  0  0  1  0  0
## 2   2  9  7  1  4  2  1  1  2  0  1  2  1  2  0  1  0  0  2  0  0  0  0  0
## 3   3  5  1  5  3  4  3  2  2  0  1  2  0  2  1  1  2  0  0  0  0  1  0  0
## 4   1  5  2  4  6  1  4  3  1  1  0  2  2  1  1  1  0  0  0  0  1  0  0  0
## 5  11  5  5  6  4  6  0  4  3  2  1  0  1  0  2  0  0  1  1  0  0  0  0  0
##    76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
## 1   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
## [[2]]
##    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##    28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##    52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## 1   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##    76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
## 1   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 4   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Catch and abundance index

Time series data (catch and abundance index) should me in matrix form with fleets along the rows and years along the columns. Rows and columns do not need to be explicitly named with fleet number or year, but the number of years should match the total number of years modeled. If data are not available in any given year there should be a negative place-holder.

With one fleet, the catch or abundance index time series should still be in matrix form rather than a vector (the model will be looking for the first row, for the one and only fleet).

Catch, one fleet:

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 59.46262 59.36848 52.76642 54.73389 47.12109 44.56865 44.38514
##          [,8]     [,9]    [,10]   [,11]    [,12]    [,13]    [,14]   [,15]
## [1,] 35.38133 48.96205 60.77217 55.5668 77.54452 87.57765 84.94026 105.492
##         [,16]    [,17]    [,18]    [,19]   [,20]
## [1,] 129.1808 153.0853 168.5263 184.8633 187.971
```

Abundance index, one fleet:

```
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.01483963 0.01477266 0.01429386 0.01478802 0.01302629 0.01851097
##           [,7]       [,8]      [,9]      [,10]      [,11]      [,12]
## [1,] 0.0172011 0.01417072 0.0164843 0.01755709 0.02034781 0.01827671
##           [,13]      [,14]      [,15]      [,16]      [,17]      [,18]
## [1,] 0.01771503 0.01604935 0.01521179 0.01664973 0.01326307 0.01454048
##          [,19]      [,20]
## [1,] 0.0101772 0.01199936
```

With multiple fleets, the catch and abundance index will have multiple rows. Any missing data will have a negative place-holder.

Catch, two fleets:

```
##            [,1]      [,2]       [,3]       [,4]       [,5]       [,6]
## [1,]   36.84685   33.0563   29.62745   34.84104   28.65523   30.31546
## [2,] -999.00000 -999.0000 -999.00000 -999.00000 -999.00000 -999.00000
##            [,7]      [,8]       [,9]      [,10]    [,11]    [,12]    [,13]
## [1,]   31.93961   26.3029   45.87026   43.48086 34.96187 54.97020 43.18992
## [2,] -999.00000 -999.0000 -999.00000 -999.00000 25.36398 31.62263 30.40232
##         [,14]    [,15]    [,16]    [,17]    [,18]    [,19]    [,20]
## [1,] 34.92335 42.60325 41.58836 40.62811 38.26569 37.50054 34.94907
## [2,] 30.10680 28.74993 28.30972 37.48081 35.93396 47.42923 35.91556
```

Abundance index, two fleets:

```
##               [,1]          [,2]         [,3]          [,4]          [,5]
## [1,]    0.01658904    0.01589921    0.0160789    0.01613434    0.01446922
## [2,] -999.00000000 -999.00000000 -999.0000000 -999.00000000 -999.00000000
##               [,6]          [,7]          [,8]          [,9]         [,10]
## [1,]    0.01480824    0.01411623    0.01448008    0.01772401    0.01904159
## [2,] -999.00000000 -999.00000000 -999.00000000 -999.00000000 -999.00000000
##           [,11]      [,12]      [,13]      [,14]      [,15]      [,16]
## [1,] 0.01869168 0.01995806 0.02087256 0.01526838 0.01229024 0.01673507
## [2,] 0.01941986 0.01761610 0.01907365 0.01771191 0.01635546 0.01549947
##           [,17]      [,18]      [,19]      [,20]
## [1,] 0.01382445 0.01359398 0.01647612 0.01485966
## [2,] 0.01393730 0.01668471 0.01400380 0.01902703
```

Data input

Observations

LIME requires inputing observed data in a list form, including:

- `years`: inclusive years to be modeled
- `LF`: length frequency data in matrix, array, or list form
- `neff_ft`: effective sample size to use in case of multinomial distribution (otherwise use Dirichlet-multinomial to estimate effective sample size, or assume nominal sample size = effective sample size if not included)
- `I_ft`: Index of abundance by fleet and time (if applicable)
- `C_ft`: Catch by fleet and time (if applicable) in numbers or biomass (to be specified later)

```
data_all <- list(years = 1:true$Nyears, LF = LF_array, I_ft = true$I_ft, C_ft = true$Cw_ft,
    neff_ft = true$obs_per_year)
```

Life history and starting values

LIME includes a function `create_inputs` which checks the length frequency data and includes all input data, life history information, and starting values in a list together to input into LIME.

`create_inputs` requires:

- `lh`: life history and starting values list output from `create_lh_inputs`
- `input_data`: input data list described in Data input section above.

```
inputs_all <- create_inputs(lh = lh, input_data = data_all)
```

Step 3: Run LIME

Estimated parameters

LIME is set up to estimate certain parameters and fix others by default.

LIME estimates by default:

- `log_F_ft`: matrix of fishing mortality estimates in log-space by fleet (rows) over time (columns)
- `log_sigma_R`: recruitment standard deviation in log-space
- `log_S50_f`: length at 50% selectivity in log space, one for each fleet
- `log_Sdelta_f`: difference between length at 95% selectivity and length at 50% selectivity in log-space, so that length at 95% selectivity can never be less than length at 50% selectivity, one for each fleet
- `Nu_input`: temporal variation in recruitment, treated as random effect

Parameters estimated by default under certain conditions:

- `log_theta`: Dirichlet-multinomial parameter relating to effective sample size (only estimated if using Dirichlet-multinomial to fit to length composition data, where `run_LIME` argument `LFdist=1`)
- `beta`: equilibrium recruitment in log-space, serves as population scaling parameter (only estimated if fitting to catch data, `run_LIME` argument `data_avail` must include "Catch")
- `log_q_f`: catchability coefficient in log-space, one for each fleet (only estimated if fitting to an index, `run_LIME` argument `data_avail` must include "Index")

Other parameters that could be estimated but are fixed by default:

- `log_sigma_C`: observation error on catch, fixed at log(0.2)
- `log_sigma_I`: observation error on abundance index, fixed at log(0.2)
- `log_CV_L`: coefficient of variation on predicted age-length curve, fixed at log(0.1)

run_LIME function

The function `run_LIME` used to run LIME models has many settings, but hopefully most models will keep the defaults and use only a few arguments.

Basic inputs to `run_LIME`:

- `modpath`: model path to save results and flags; default=NULL to save in R environment locally
- `input`: list output from `create_inputs`
- `data_avail`: what data types will LIME fit to? "LC"= length composition only, "Index_LC"= index and length comps, "Catch_LC"= catch and length comps, and "Index_Catch_LC" = index, catch, and length comps.

If fitting to catch data, the user must specify:

- `C_type`: default=0 (no catch data), 1 for catch in numbers, 2 for catch in biomass

Additional settings in `run_LIME`:

- `LFdist`: which distribution to fit to length frequency data? default=1 to use Dirichlet-multinomial and estimate additional parameter related to effective sample size, multinomial = 0
- `est_more`: additional parameters to estimate that are fixed by default (see Estimated parameters section)
- `fix_more`: additional parameters to fix that are estimated by default (see Estimated parameters section)
- `est_F_ft`: which F parameters to estimate? default=TRUE to estimate all. Otherwise, a matrix with fleets as rows and years along columns, with a 1 where the F parameter should be estimated and a 0 where the F parameter should be fixed at the starting value.
- `f_startval_ft`: starting values for fishing mortality, default=NULL to start at 0.2 for all years and fleets. Otherwise, matrix of starting values for F where fleets are along the rows and years along the columns (e.g. can input the estimated F values from the previous run with Report$F_ft).
- `rdev_startval_t`: starting values for recruitment deviations over time. default=NULL to start at 0 for all years. Otherwise, specify vector of recruitment deviations for all years to be modeled (e.g. can start at simulated truth and turn off estimation for debugging)
- `est_selex_f`: estimate selectivity parameters? default=TRUE to estimate selectivity parameters for all fleets. Turn off selectivity estimation for all or a single fleet with FALSE (e.g. estimate selectivity parameters for fleet 1 but not fleet 2 with c(TRUE, FALSE))
- `vals_selex_ft`: input selectivity-at-length (columns) by fleet (rows). default= matrix of negative values to estimate selectivity. Otherwise, input selectivity and set `est_selex_f`=FALSE. This is how one would go about fixing the selectivity curve to a dome shape.
- `randomR`: default=TRUE treats recruitment deviations as a random effect. FALSE turns the random effect off.
- `newtonsteps`: number of extra Newton steps to take after optimization to help convergence, default=3; FALSE to turn off. Note: it seems with TMB that bounding parameters does not work when using newtonsteps. If a parameter is estimated outside of a reasonable bound, I recommend setting newtonsteps=FALSE.
- `F_up`: upper bound on fishing mortality estimate
- `S50_up`: upper bound on length at 50% selectivity
- `Fpen`: penalty on fishing mortality to avoid fishing mortality being estimated at drastically different values between years; default=1 on, otherwise 0=off.
- `SigRpen`: penalty on sigmaR, default=1 on, otherwise off=0.
- `SigRprior`: vector with prior info for normally distributed sigmaR penalty, first term is the mean and second term is the standard deviation; default =c(0.737, 0.3)
- `derive_quants`: if TRUE, derive MSYbased reference points, default=FALSE
- `est_totalF`: TRUE will estimate total fishing mortality across all fleets (number of parameters = number of years), not by fleet (where number of parameters would = number of years * number of fleets). default=FALSE estimates F by year and fleet.

- `prop_f`: proportion of catch from each fleet, must sum to 1.0. When `est_totalF`=TRUE, must assume how much of the catch comes from each fleet (e.g. c(0.5,0.5), c(0.6,0.4), etc.)
- `mirror`: vector of parameter names matching those in Estimated parameters section to mirror between fleets (estimating one parameter instead of the same number of fleets)
- `rewrite`: default=TRUE to rewrite LIME output (LIME_output.rds) in the directory `modpath`. FALSE is useful for simulation testing if the process was stopped midway through 100 iterations, so a function can skip iterations with output.

Settings in `run_LIME` associated with simulation-testing:

- `simulation`: default=FALSE, but if TRUE, sets working directory in the iteration directory, not the general `modpath`
- `itervec`: vector of iterations for simulation, e.g. 1:100. default=NULL for stock assessment application with real data (1 iteration)

Example: Data-rich model run

Here's the best-case scenario for running LIME: 20 years of length composition, catch, and an abundance index, with a typical 200 samples of length measurements per year, assuming the life history information is known without error.

Here we specify `C_type=2` because the catch data specified in `data_all` in the Data input section was catch in weight, not numbers.

```
rich <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "Index_Catch_LC",
    C_type = 2)
```

```
## Time difference of 1.065552 mins
```

The time difference above shows the run time of the data-rich case on the machine that compiled this document. The run time will vary depending on the machine, but hopefully this gives a ballpark of how long it takes to run a model with approximately 20 years and three data types.

The object `rich` contains several elements of LIME output:

- `Inputs`: List of Data, Parameters, Map, and Random elements that are input directly to the LIME.cpp program. The goal is to help the user check the exact values and structure that went in to the LIME optimization using TMB.
- `Report`: Report file of maximum likelihood estimates and derived values of population parameters from the LIME model run.
- `Sdreport`: Standard errors of all estimated parameters and derived values of population parameters which are specified in the LIME.cpp file.
- `obj`: the TMB object
- `opt`: the optimization results from TMBhelper::Optimize
- `df`: a data frame of final gradients for each estimated parameter
- `input`: input data list created using `create_inputs` function. This is used when re-running the model using the function `get_converged`.
- `data_avail`: the data types available. This is used when re-running the model using the function `get_converged`.

Check convergence

General convergence criteria for LIME models includes:

1. Final gradient of all estimated parameters is within +/- 0.001
2. Hessian matrix is positive definite

We can check this for any model run using the following code:

```
gradient <- rich$opt$max_gradient <= 0.001
hessian <- rich$Sdreport$pdHess
hessian == TRUE & gradient == TRUE
```

```
## [1] TRUE
```

Because both the hessian and gradient checks passed, we assume that the program found the global minimum of the negative log likelihood surface and can move on to examine results.

Plot results

The `LIME` package includes functions to plot length composition model fits and other results.

LIME model fits to length frequency data:

```
plot_LCfits(Inputs = rich$Inputs, Report = rich$Report)
```

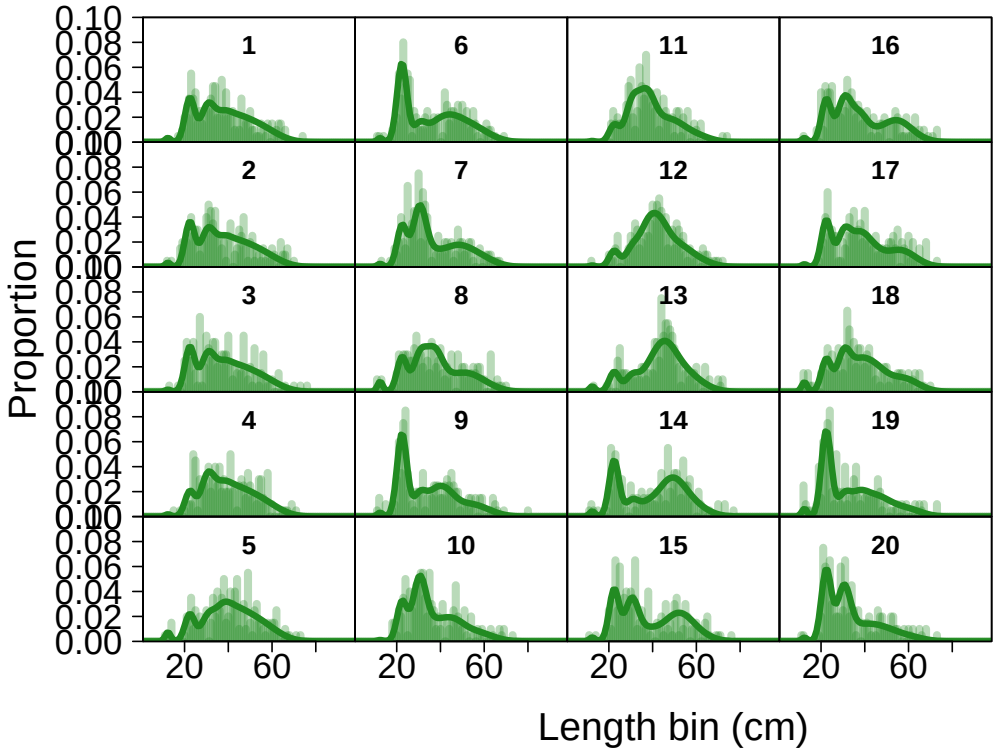


Figure 5: Fits to length composition data for the data-rich LIME model run.

LIME estimates of key population parameters:

```
plot_output(Inputs = rich$Inputs, Report = rich$Report, Sdreport = rich$Sdreport,
    lh = lh, True = true, plot = c("Fish", "Rec", "SPR", "ML", "SB", "Selex"),
    set_ylim = list(Fish = c(0, 1), SPR = c(0, 1)))
```

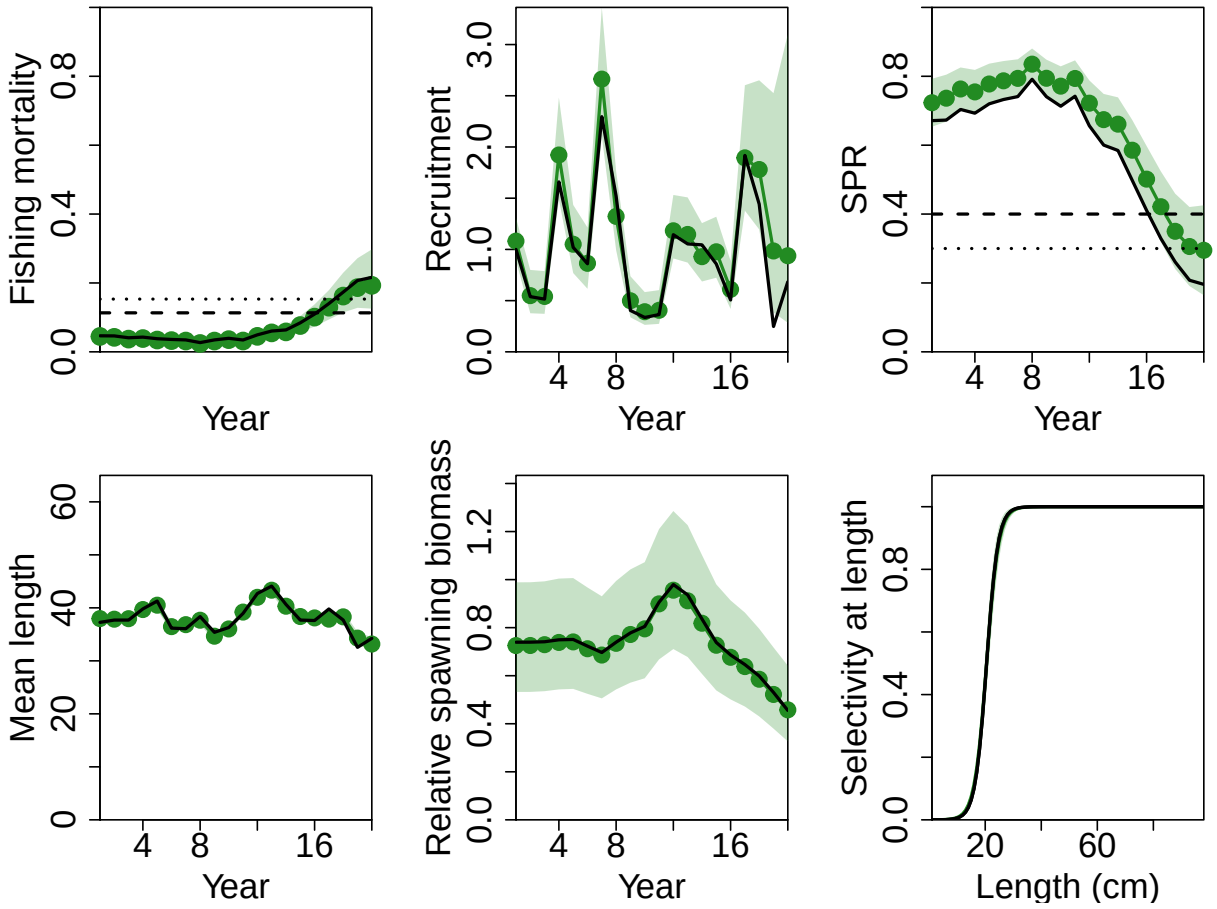


Figure 6: LIME estimates of key population parameters (green) compared to the truth (black) for the data-rich case.

Example: Length data only

This is the scenario for which LIME was developed: integrating multiple years of length data to account for time-varying fishing mortality and recruitment. The big change from the data-rich case in the code is to make sure `data_avail="LC"` and `C_type` is set to zero or removed from the function call (because the default is zero). We can still use the `inputs_all` list that includes the index and catch data, but as long as `data_avail="LC"` the model will not include the other data types.

```
lc_only <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "LC")
```

```
## Time difference of 47.75294 secs
```

Note the time difference is shorter than the data-rich case.

Check convergence

```
gradient <- lc_only$opt$max_gradient <= 0.001
hessian <- lc_only$Sdreport$pdHess
hessian == TRUE & gradient == TRUE
```

```
## [1] TRUE
```

Plot results

We can use the same LF_list as was calculated in the data-rich case, but compare the LIME model fits from the model that excluded catch and abundance index:

```
plot_LCfits(LFlist = LF_list, Inputs = lc_only$Inputs, Report = lc_only$Report)
```

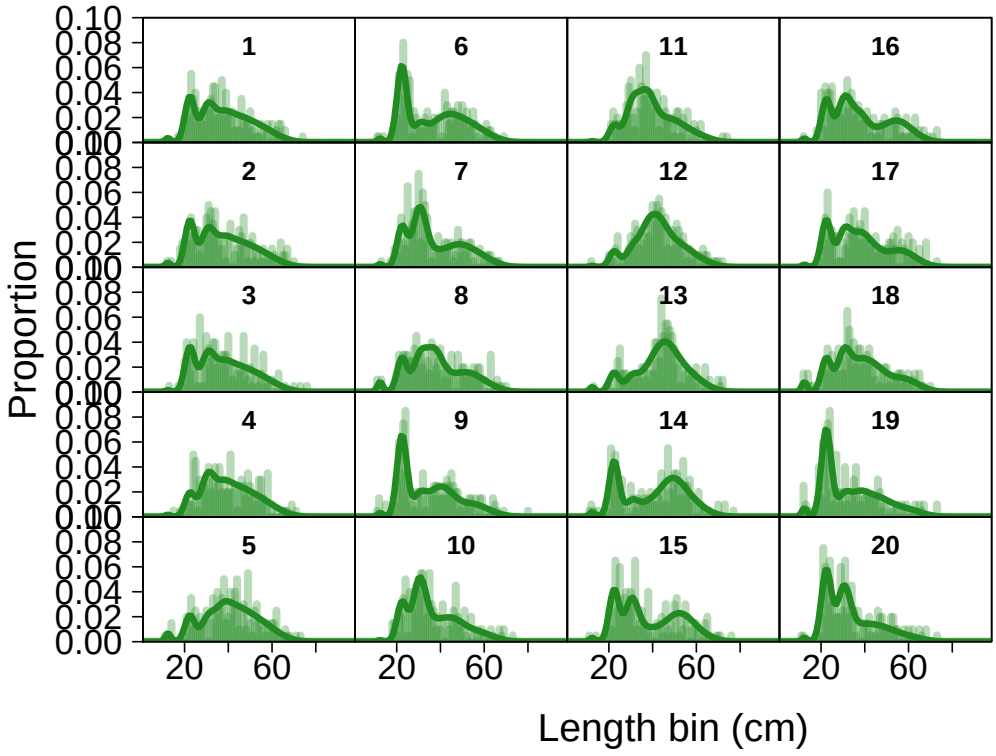


Figure 7: Fits to length composition data for the LIME model run with length composition data only.

Model fits with length data only are of course much less accurate and with higher uncertainty than the data-rich case:

```
plot_output(Inputs = lc_only$Inputs, Report = lc_only$Report, Sdreport = lc_only$Sdreport,
    lh = lh, True = true, plot = c("Fish", "Rec", "SPR", "ML", "SB", "Selex"),
    set_ylim = list(Fish = c(0, 0.5), SPR = c(0, 1)))
```

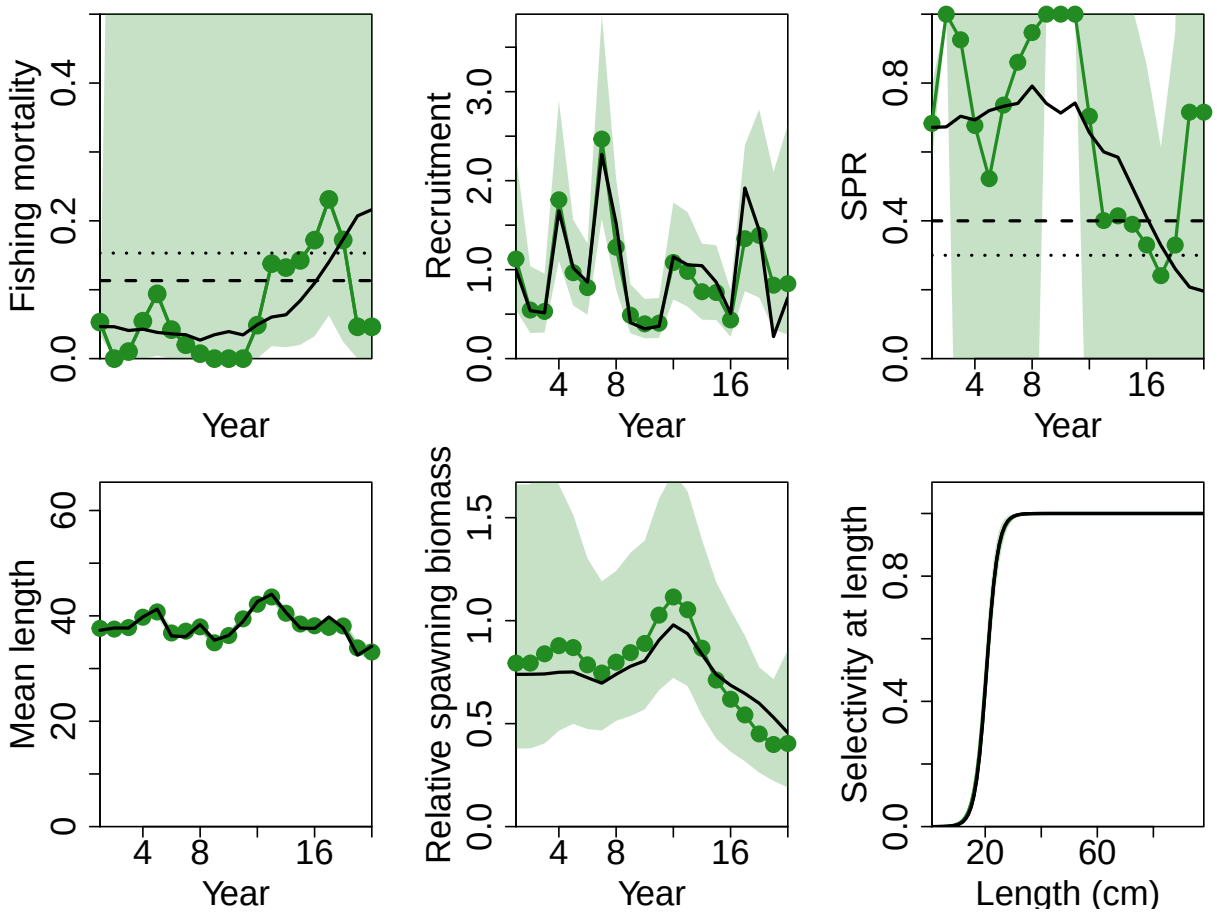


Figure 8: LIME estimates of key population parameters (green) compared to the truth (black) with length composition data only.

The model meets convergence criteria but the fishing mortality rate estimates jump around between years more than is realistic. To attempt to curb this behavior, it is possible to decrease the standard deviation for fishing mortality which will serve to penalize estimates of fishing mortality that stray too far from the previous year.

```
inputs_all$SigmaF <- 0.1
lc_only2 <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "LC")

gradient <- lc_only2$opt$max_gradient <= 0.001
hessian <- lc_only2$Sdreport$pdHess
hessian == TRUE & gradient == TRUE
```

```
plot_output(Inputs = lc_only2$Inputs, Report = lc_only2$Report, Sdreport = lc_only2$Sdreport,
    lh = lh, True = true, plot = c("Fish", "Rec", "SPR", "ML", "SB", "Selex"),
    set_ylim = list(Fish = c(0, 0.5), SPR = c(0, 1)))
```

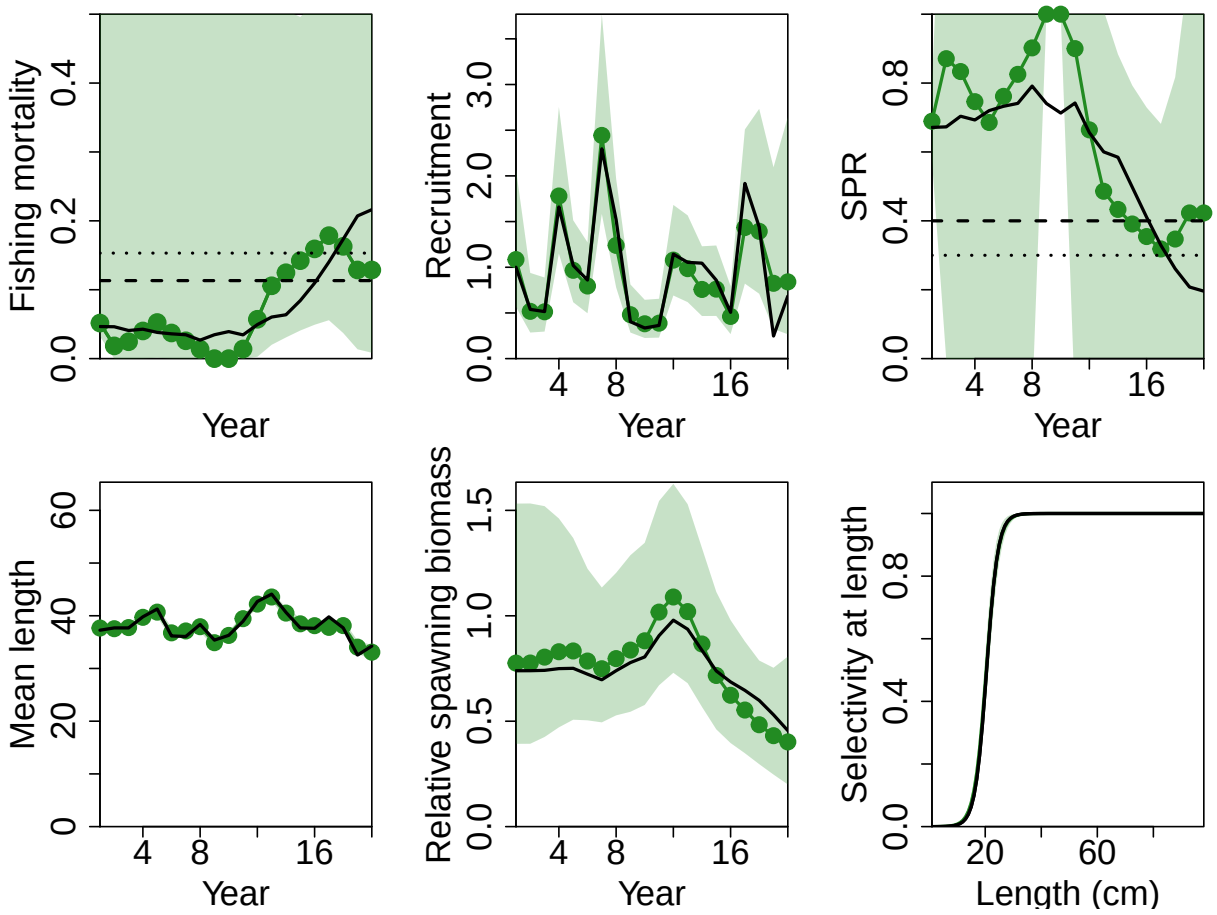


Figure 9: LIME estimates of key population parameters (green) compared to the truth (black) with length composition data only, with lower SigmaF.

Example: Length and catch data

This is the scenario most similar to the ideas behind catch-curve stock reduction analysis (CCSRA). CCSRA used catch and age composition data to estimate stock status. LIME can use catch and length composition, which is much easier to collect than ages. With accurate assumptions about life history information, the combination of catch and length data can be very informative of stock status.

```
catch_lc <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "Catch_LC",
    C_type = 2)
```

```
## Time difference of 1.17716 mins
```

Check convergence

```
gradient <- catch_lc$opt$max_gradient <= 0.001
hessian <- catch_lc$Sdreport$pdHess
hessian == TRUE & gradient == TRUE
```

```
## [1] TRUE
```

Plot results

Model fits including catch data with length composition data:

```
plot_LCfits(LFlist = LF_list, Inputs = catch_lc$Inputs, Report = catch_lc$Report)
```

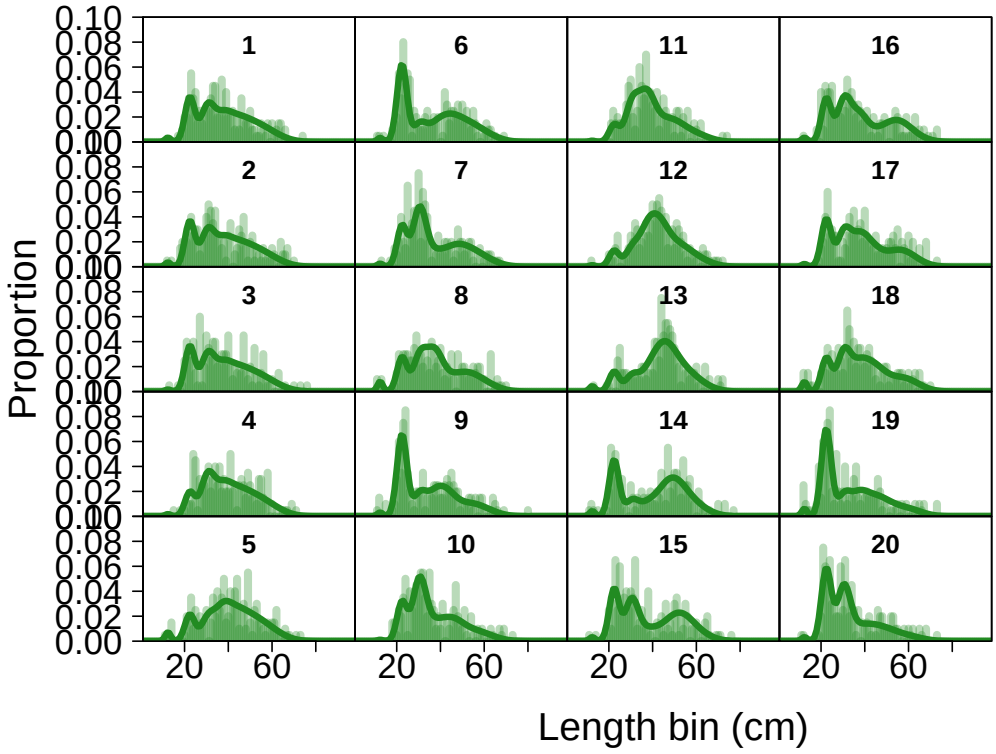


Figure 10: Fits to length composition data for the LIME model run with catch and length data.

Including catch data improves estimates of population parameters over length data only.

```
plot_output(Inputs = catch_lc$Inputs, Report = catch_lc$Report, Sdreport = catch_lc$Sdreport,
    lh = lh, True = true, plot = c("Fish", "Rec", "SPR", "ML", "SB", "Selex"),
    set_ylim = list(Fish = c(0, 0.5), SPR = c(0, 1)))
```

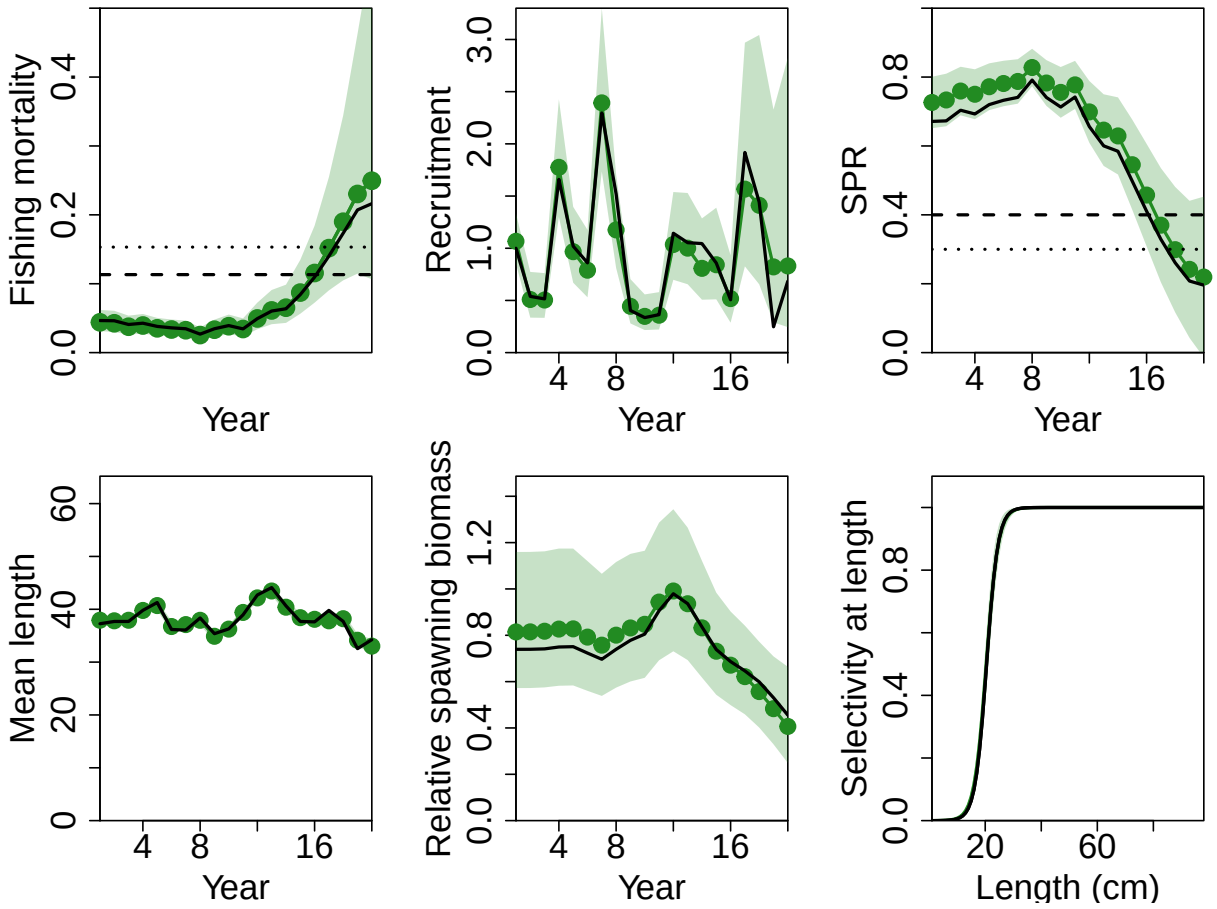


Figure 11: LIME estimates of key population parameters (green) compared to the truth (black) with catch and length data.

Turning off parameter estimation

There may be non-convergence issues (apparent from either a high final gradient or non-positive definite Hessian matrix). Some first checks in these cases are to 1) see if the Dirichlet-multinomial theta parameter is estimated too high, in which case it can be fixed to a high value and not estimated:

```
inputs_all$theta <- 50
check1 <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "LC", fix_more = "log_theta")
```

Another option is to check if the recruitment standard deviation is estimated too high or too low, in which case it can be fixed at a reasonable value:

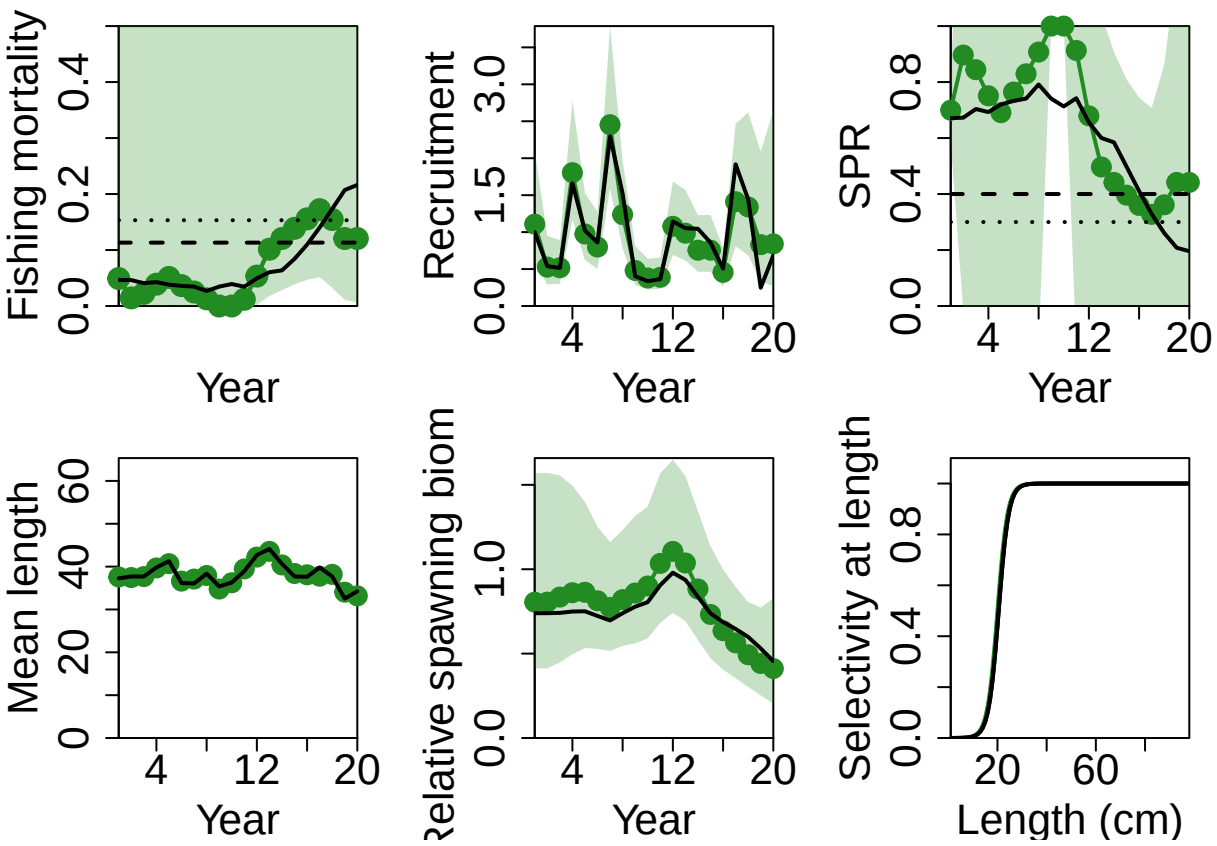
```
inputs_all$SigmaR <- 0.3
check2 <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "LC", fix_more = "log_sigma_R")
```

Some length datasets may not be informative enough to tease apart fishing mortality, recruitment deviations, and selectivity. In these cases it may be convenient or helpful to fix selectivity at the starting values:

```
check3 <- run_LIME(modpath = NULL, input = inputs_all, data_avail = "LC", est_selex_f = FALSE)

plot_output(Inputs = check3$Inputs, Report = check3$Report, Sdreport = check3$Sdreport,
```

```
    lh = lh, True = true, plot = c("Fish", "Rec", "SPR", "ML", "SB", "Selex"),
    set_ylim = list(Fish = c(0, 0.5), SPR = c(0, 1)))
```



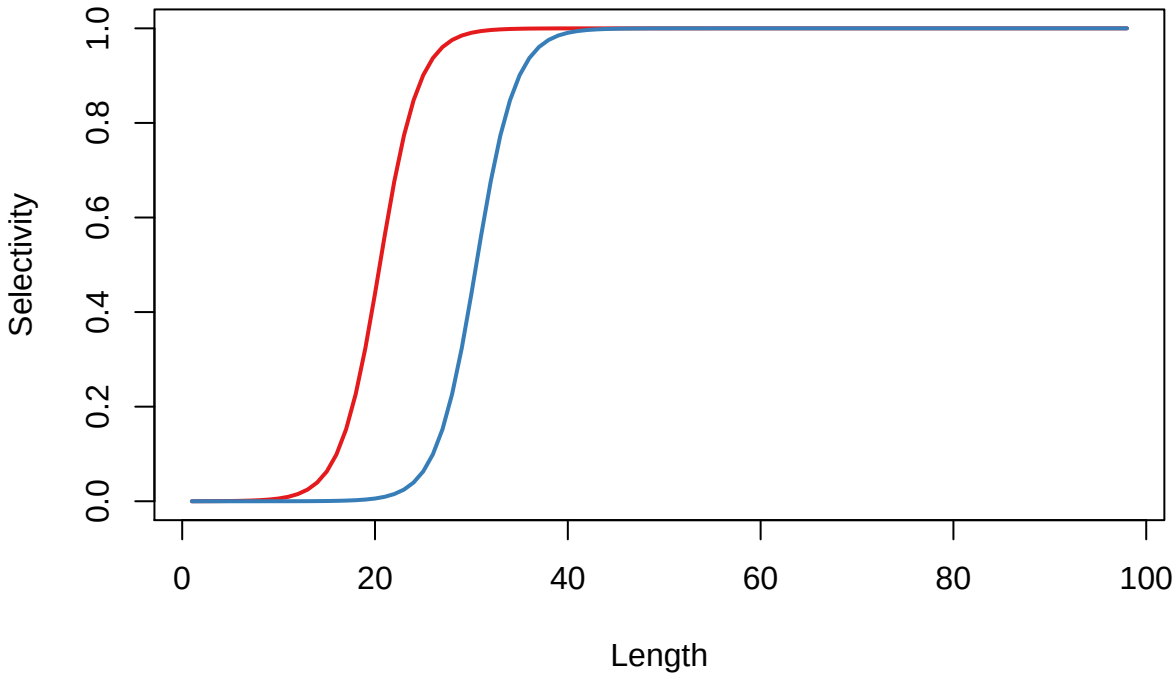
Multiple fleets

With multiple fleets, all biological inputs are the same as with a single fleet. However, we must specify selectivity for each fleet. In this case, both selectivities are logistic.

```
lh_mf1 <- with(lh, create_lh_list(vbk = vbk, linf = linf, t0 = t0, lwa = lwa,
    lwb = lwb, S50 = c(20, 30), S95 = c(26, 36), selex_input = "length", selex_type = c("logistic",
        "logistic"), M50 = ML50, M95 = NULL, maturity_input = "length", M = M,
    h = h, binwidth = binwidth, CVlen = CVlen, SigmaR = SigmaR, SigmaF = SigmaF,
    SigmaC = SigmaC, SigmaI = SigmaI, R0 = R0, Frate = Frate, qcoef = qcoef,
    start_ages = 0, rho = rho, nseasons = 1, nfleets = 2))
```

Using the simulation function for multiple fleets, we need to specify the fleet dynamics for each (`Fdynamics`), the number of years of length composition data for each (`Nyears_comp`), and the proportions of each fleet (`fleet_proportions`).

```
true_mf1 <- generate_data(modpath = NULL, itervec = 1, Fdynamics = c("Constant",
    "Endogenous"), Rdynamics = "Constant", lh = lh_mf1, Nyears = 20, Nyears_comp = c(20,
    10), comp_sample = 200, init_depl = 0.7, seed = 123, fleet_proportions = c(0.6,
    0.4))
```




```
par(mfrow = c(3, 2))
plot(true_mf1$F_y, type = "l", lwd = 4, xlab = "Year", ylab = "Fishing mortality",
    ylim = c(0, max(true_mf1$F_y) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5)
for (i in 1:lh_mf1$nfleets) {
    lines(true_mf1$F_ft[i, ], col = cols[i])
}
plot(true_mf1$R_t, type = "l", lwd = 4, xlab = "Year", ylab = "Recruitment",
    ylim = c(0, max(true_mf1$R_t) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5)
plot(true_mf1$SPR_t, type = "l", lwd = 4, xlab = "Year", ylab = "SPR", ylim = c(0,
    1), xaxs = "i", yaxs = "i", cex.axis = 1.5, cex.lab = 1.5)
plot(true_mf1$D_t, type = "l", lwd = 4, xlab = "Year", ylab = "Relative spawning biomass",
    ylim = c(0, max(c(1, max(true_mf1$D_t)))), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5)
plot(colSums(true_mf1$Cw_ft), type = "l", lwd = 4, xlab = "Year", ylab = "Catch",
    ylim = c(0, max(colSums(true_mf1$Cw_ft)) * 1.1), xaxs = "i", yaxs = "i",
    cex.axis = 1.5, cex.lab = 1.5)
for (i in 1:lh_mf1$nfleets) {
    lines(true_mf1$Cw_ft[i, ], col = cols[i])
}
plot(true_mf1$I_ft[1, ], type = "l", xlab = "Year", ylab = "Abundance index",
    ylim = c(0, max(true_mf1$I_ft) * 1.1), xaxs = "i", yaxs = "i", cex.axis = 1.5,
    cex.lab = 1.5, col = cols[1])
lines(true_mf1$I_ft[2, ], col = cols[2])
```

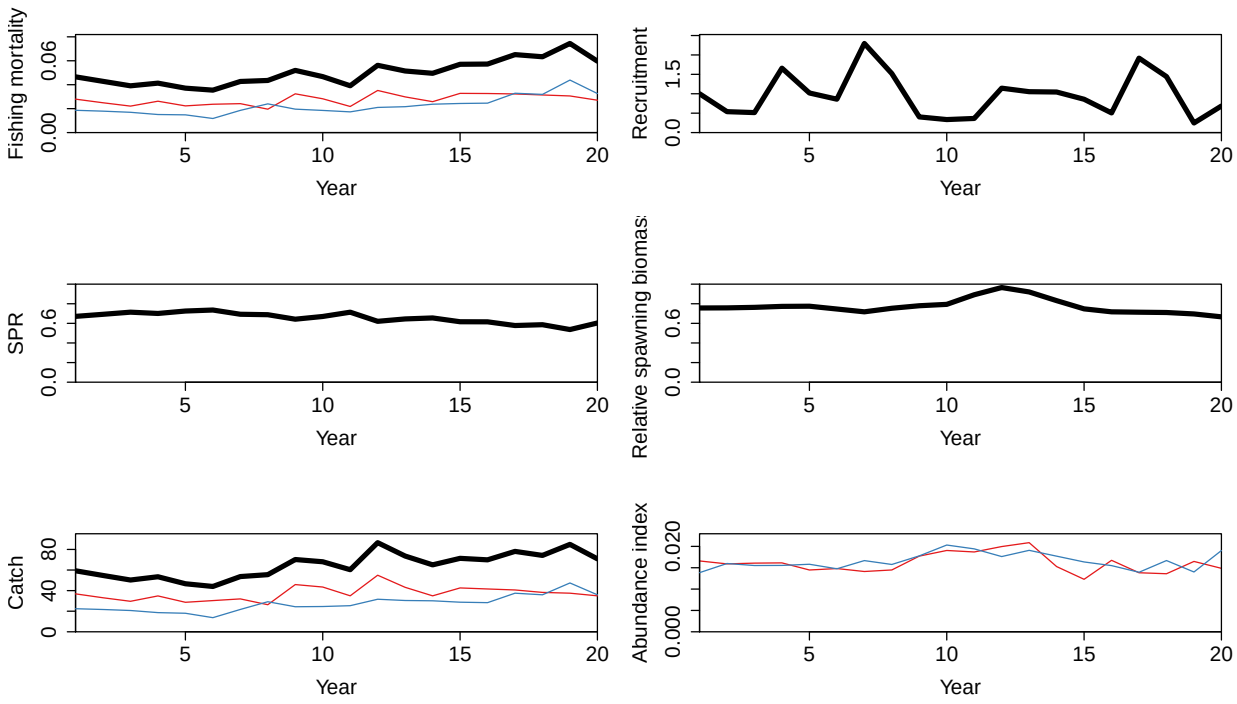


Figure 12: True population with two fleets, one with constant fishing mortality, and one with open access fishing.

Move the generated data from an array to a list to plot the data.

```
plot_LCfits(LFlist = LF_list)
```

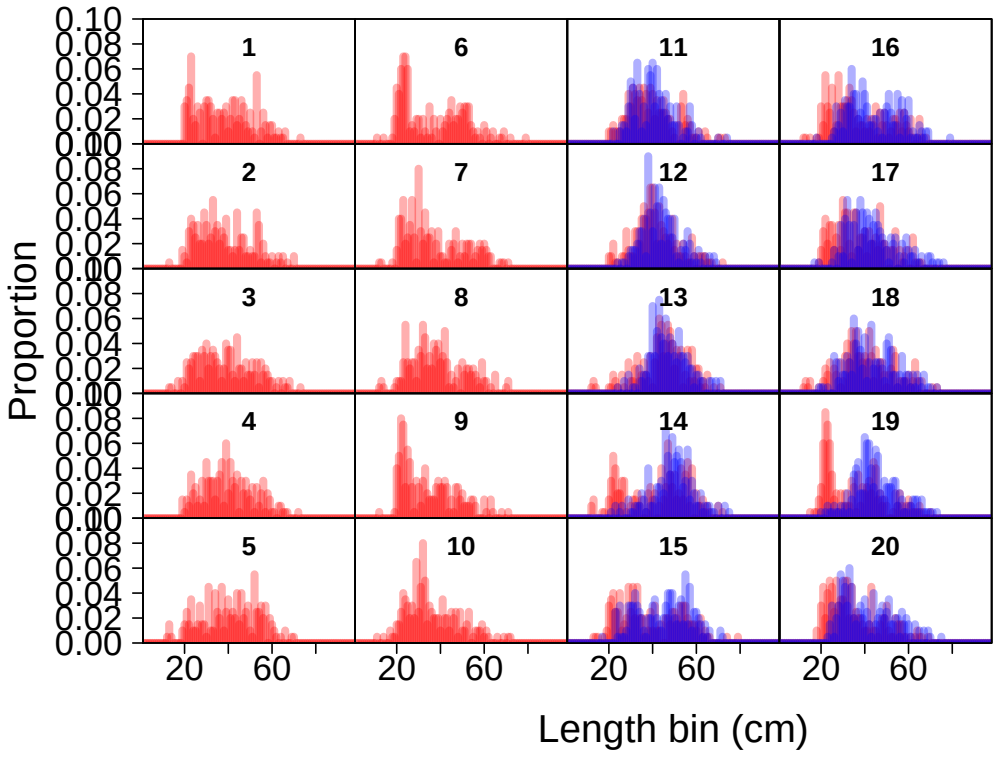


Figure 13: Generated length composition data, labeled by years 1-20.

```
data_mf1 <- list(years = 1:true_mf1$Nyears, LF = true_mf1$LF, I_ft = true_mf1$I_ft,
    C_ft = true_mf1$Cw_ft, neff_ft = true_mf1$obs_per_year)
inputs_mf1 <- create_inputs(lh = lh_mf1, input_data = data_mf1)
```

For now, use the multinomial distribution for multiple fleets, specifying `LFdist=0`.

```
rich_mf1 <- run_LIME(modpath = NULL, input = inputs_mf1, data_avail = "Index_Catch_LC",
    C_type = 2, LFdist = 0)
```

Check that the data-rich model converges:

```
gradient <- rich_mf1$opt$max_gradient <= 0.001
hessian <- rich_mf1$Sdreport$pdHess
hessian == TRUE & gradient == TRUE
```

```
## [1] TRUE
```

```
plot_LCfits(Inputs = rich_mf1$Inputs, Report = rich_mf1$Report)
```

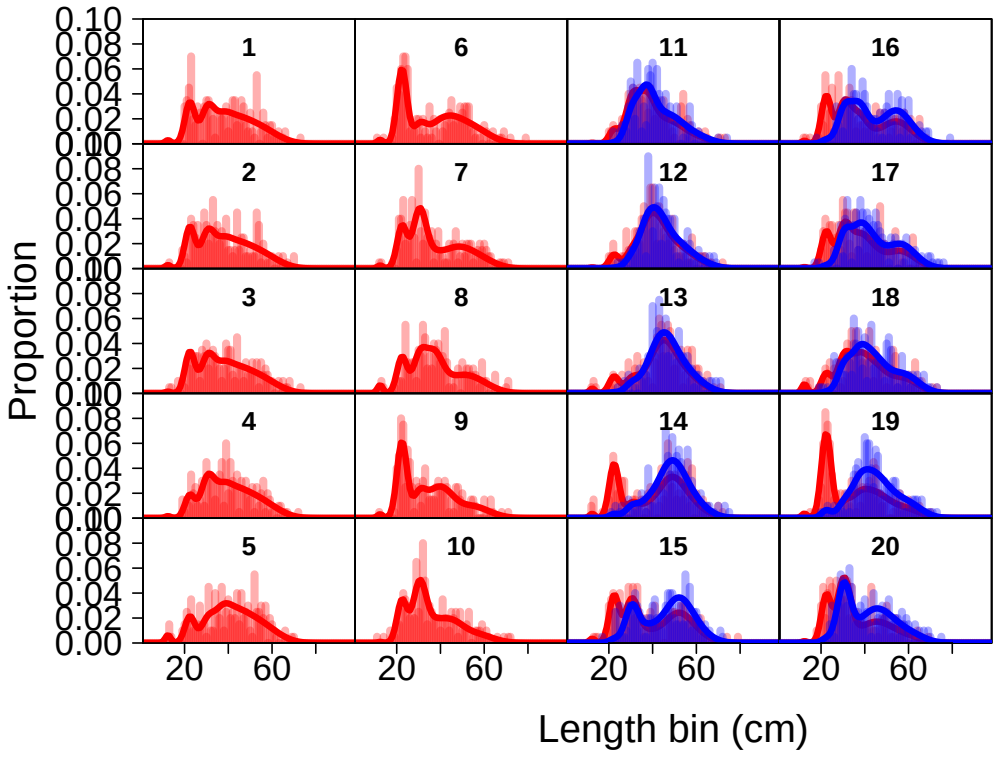


Figure 14: Fits to length composition data for the data-rich LIME model run with two fleets, both with logistic selectivity.

```
plot_output(Inputs = rich_mf1$Inputs, Report = rich_mf1$Report, Sdreport = rich_mf1$Sdreport,
    lh = lh_mf1, True = true_mf1, plot = c("Fish", "Rec", "SPR", "ML", "SB",
        "Selex"), set_ylim = list(SPR = c(0, 1)))
```

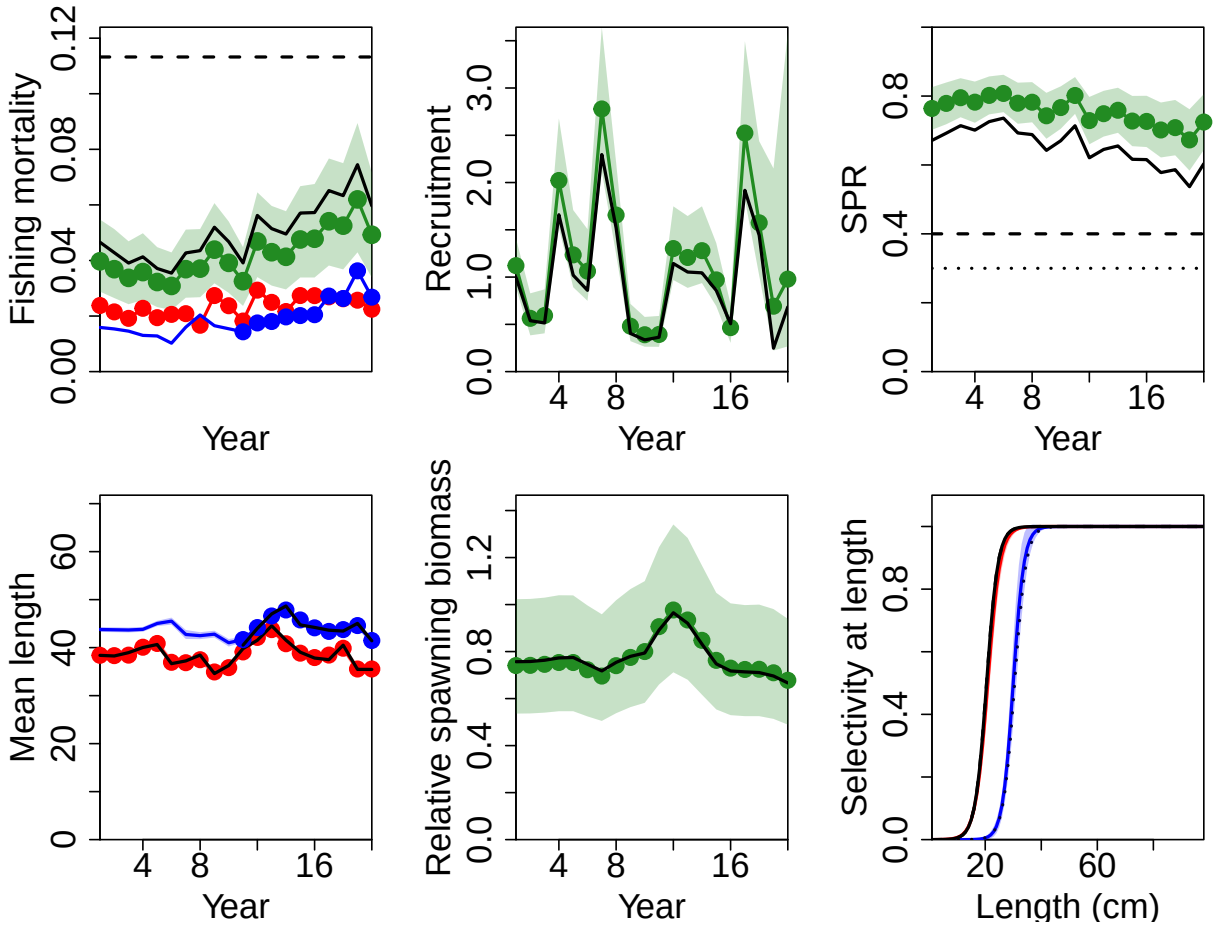


Figure 15: LIME estimates of key population parameters (green) compared to the truth (black) for the data-rich case with two fleets each with logistic selectivity.

With length data only, we need to estimate the total fishing mortality (as opposed to annual fishing mortality for each fleet). It's also useful to set the fishing mortality penalty to something relatively small to help the estimation.

```
inputs_mf1$SigmaF <- 0.1
lc_mf1 <- run_LIME(modpath = NULL, input = inputs_mf1, data_avail = "LC", LFdist = 0,
    est_totalF = TRUE)

gradient <- lc_mf1$opt$max_gradient <= 0.001
hessian <- lc_mf1$Sdreport$pdHess
hessian == TRUE & gradient == TRUE
```

```
plot_LCfits(Inputs = lc_mf1$Inputs, Report = lc_mf1$Report)
```

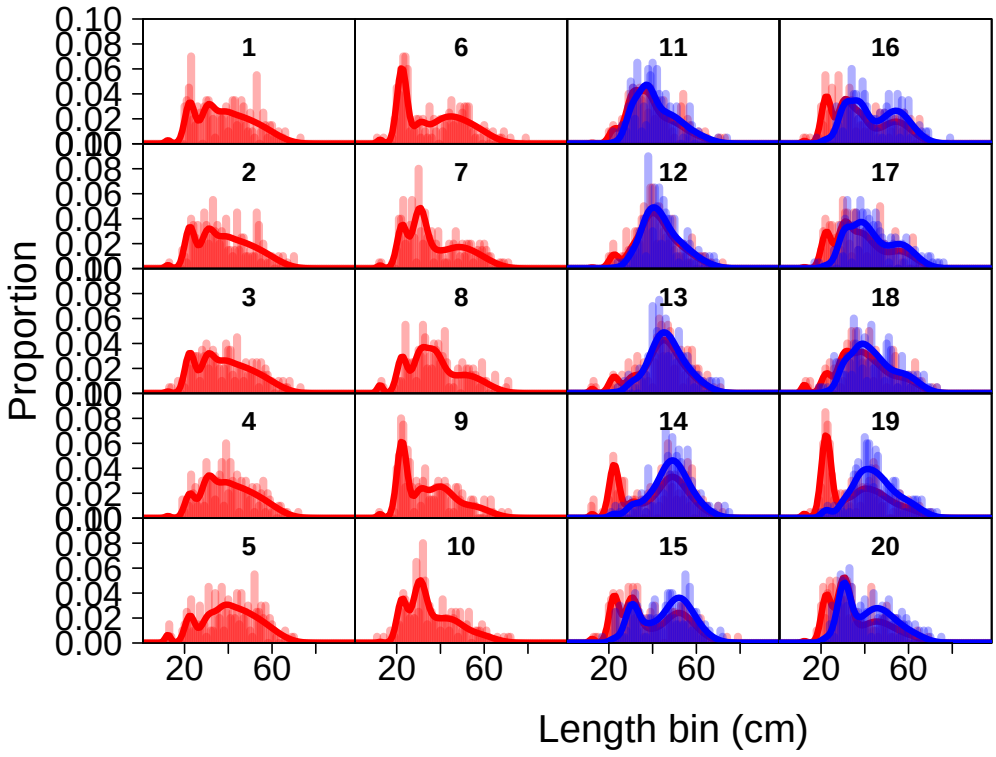


Figure 16: Fits to length composition data for LIME model with length data only for two fleets, both with logistic selectivity.

```
plot_output(Inputs = lc_mf1$Inputs, Report = lc_mf1$Report, Sdreport = lc_mf1$Sdreport,
    lh = lh_mf1, True = true_mf1, plot = c("Fish", "Rec", "SPR", "ML", "SB",
        "Selex"), set_ylim = list(SPR = c(0, 1)))
```

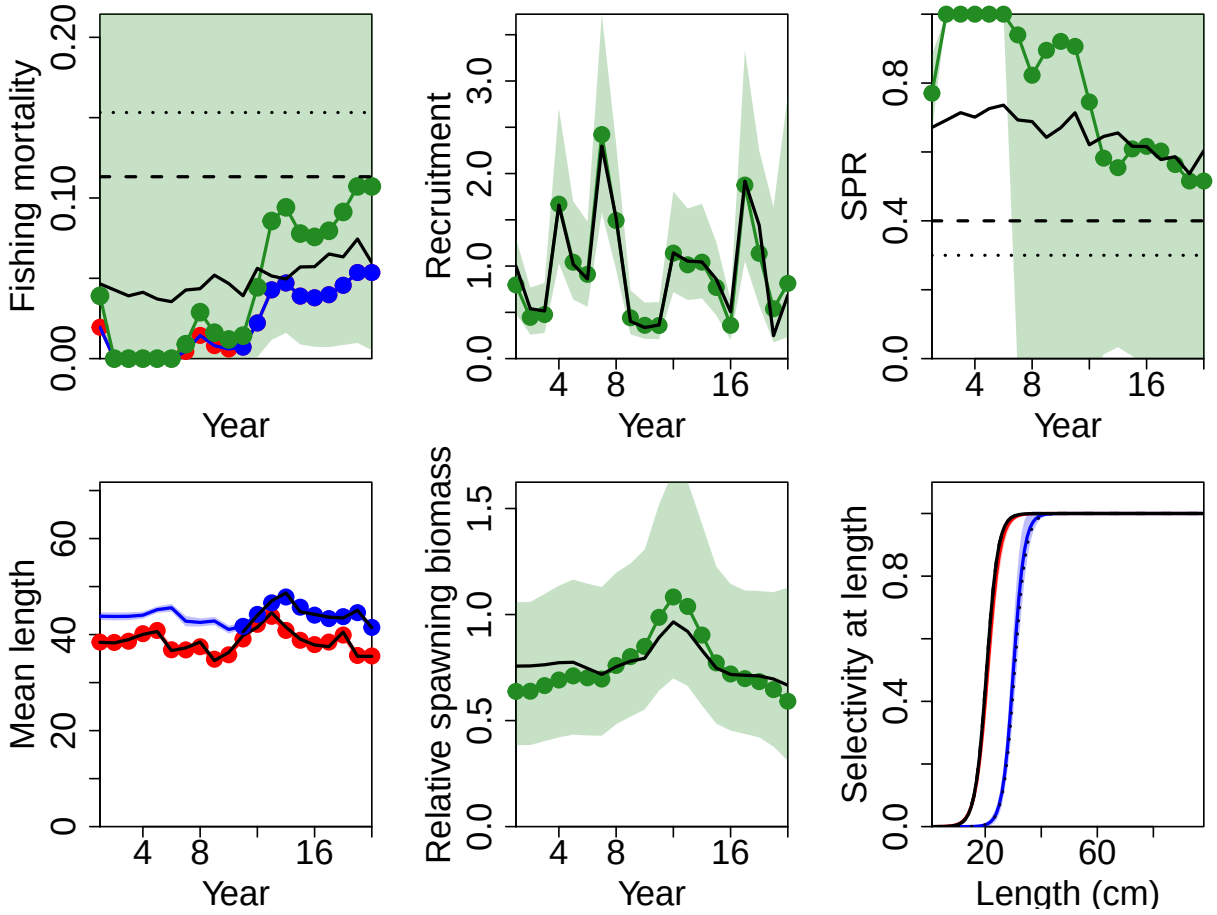


Figure 17: LIME estimates of key population parameters (green) compared to the truth (black) for the length-only case with two fleets each with logistic selectivity.