Package: rFAMS (via r-universe)

May 16, 2026

Version 0.0.3.9000

Date 2026-3-11

Title Fisheries Analysis and Modeling Simulator

Description Simulates the dynamics of exploited fish populations using the Jones modification of the Beverton-Holt equilibrium yield equation to compute yield-per-recruit and dynamic pool models (Ricker 1975) <https://publications.gc.ca/site/eng/480738/publication.html>. Allows users to evaluate minimum, slot, and inverted length limits on exploited fisheries using specified life history parameters. Users can simulate population under a variety of conditional fishing mortality and conditional natural mortality. Calculated quantities include number of fish harvested and dying naturally, mean weight and length of fish harvested, number of fish that reach specified lengths of interest, total number of fish and biomass in the population, and stock density indices.

URL https://github.com/fishR-Core-Team/rFAMS/

BugReports https://github.com/fishR-Core-Team/rFAMS/issues

License GPL (>= 2)

Depends R (>= 4.1.0)

Imports stats, purrr, FSA

Suggests dplyr, ggplot2, metR, knitr, rmarkdown, testthat (>= 3.0.0), zipfR, quarto, FSAdata, tidyr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/Needs/website quarto

Config/pak/sysreqs cmake make libicu-dev

Repository https://fishr-core-team.r-universe.dev

Date/Publication 2026-03-17 22:48:35 UTC

RemoteUrl https://github.com/fishr-core-team/rfams

RemoteRef HEAD

RemoteSha 69e71668bcc37c851786873130f0dcdc89b48efb

Contents

`dpmBH_MinLL` *Simulate expected yield under minimum length regulations using the Dynamic Pool model for a range of input parameters*

Description

Simulate yield under minimum length regulations using the Dynamic Pool (DPM) model with (possibly) multiple values for conditional fishing mortality (`cf`) and conditional natural mortality (`cm`).

Usage

```
dpmBH_MinLL(
  minLL,
  cf,
  cm,
  rec,
  lhparms,
  simyears,
  species = NULL,
  group = NULL,
  matchRicker = FALSE
)
```

Arguments

`minLL`	A single numeric representing the minimum length limit for harvest in mm.
`cf`	A matrix of conditional fishing mortality where each row represents a year and each column represents an age (age-0 through maximum age; i.e., `tmax` in `lhparms`). All values must be between 0 and 1 (inclusive).
`cm`	A matrix of conditional natural mortality where each row represents a year and each column represents an age (age-0 through maximum age; i.e., `tmax` in `lhparms`). All values must be between 0 and 1 (inclusive).
`rec`	A numeric vector with length `simyears` that specifies the number of recruits each year. This vector is best generated using the `genRecruits`. All values must be greater than 0.
`lhparms`	A named vector or list that contains values for each `N0`, `tmax`, `Linf`, `K`, `t0`, `LWalpha`, and `LWbeta`. See `makeLH` for definitions of these life history parameters. Also see details.
`simyears`	A single numeric for the number of years to simulate. Value must be a whole number greater than 1.
`species`	A single character to specify the species used in the simulation. This will define the length for `stock`, `quality`, `preferred`, `memorable`, and `trophy` lengths from the FSA package. See the `PSDlit` documentation.
`group`	A single character to specify the sub-group name for `species` which may be required when defining the `stock`, `quality`, `preferred`, `memorable`, and `trophy` length categories from the FSA package. See the `PSDlit` documentation.
`matchRicker`	A logical that indicates whether the yield function should match that in Ricker (1975). Defaults to `FALSE`. See the FAMS vs Ricker article.

Details

Details will be filled out later.

Note that the main calculations are in the internal `dpmBH_func` (use `rFAMS:::dpmBH_func` to see that source code).

Value

A list with two data.frame object. The first list item named `sumbyAge` contains a data.frame with the following calculated values in a summary by age:

- `year` is the year number for the simulation
- ycis the year class number for the simulation
- age is the age of fish from the year class
- `length` is the length-at-age at the beginning of the year based on parameters supplied for the von Bertlanffy growth model.
- `weight` is the total weight at the beginning of the year for length-at-age based on the parameters supplied for the weight-length model.
- `N_start` is the number of fish alive at the start of the year for the given age and year class.

- `exploitation` is the exploitation rate at age based on the supplied conditional fishing mortality rate.
- `expect_nat_death` is the expectation of natural death based on the supplied conditional natural mortality rate.
- `cf` is the supplied conditional fishing mortality rate.
- `cm` is the supplied conditional natural mortality rate.
- `F` is the instantaneous rate of fishing mortality.
- `M` is the instantaneous rate of natural mortality.
- `Z` is the instantaneous rate of total mortality.
- `S` is the (total) annual rate of survival.
- `biomass` is the total biomass of fish at age and year.
- `N_harvest` is the total number of fish harvested at age and year.
- `N_die` is the total number of fish that die at age and year.
- `yield` is the estimated yield (in g).
- `minLL` is the minimum length limit specified in the simulation.

For convenience the data.frame also contains the model input values (`N0`, `Linf`, `K`, `t0`, `LWalpha`, `LWbeta`, and `tmax`).

The second list item named `sumbyYear` contains a data.frame with the following calculated values in a summary by year:

- `year` is the year number for the simulation
- `Age_1plus` is the total number of fish age-1 plus per year.
- `Yield_Age_1plus` is the total year of age-1 plus fish per year.
- `Total_biomass` is the total biomass of age-1 plus fish per year.
- `N_harvest_Age_1plus` is the number of age-1 plus fish that are harvested per year.
- `N_die_Age_1plus` is the number of age-1 plus fish that die per year.
- `substock` is the number of substock sized fish at age and year at the beginning of the year.
- `stock` is the number of stock sized fish at age and year at the beginning of the year.
- `quality` is the number of quality sized fish at age and year at the beginning of the year.
- `preferred` is the number of preferred sized fish at age and year at the beginning of the year.
- `memorable` is the number of memorable sized fish at age and year at the beginning of the year.
- `trophy` is the number of trophy sized fish at age and year at the beginning of the year.
- `PSD` is the number of quality sized fish divided by the number of stock sized multiplied by 100.
- `PSD_P` is the number of preferred sized fish divided by the number of stock sized multiplied by 100.
- `PSD_M` is the number of memorable sized fish divided by the number of stock sized multiplied by 100.
- `PSD_T` is the number of trophy sized fish divided by the number of stock sized multiplied by 100.

PSD-X are calculated based on the number of fish in each category (`stock`, `quality`, `preferred`, `memorable`, and `trophy`) at the beginning of the year. That is, the length-at-age during the start of the year is used to assign PSD-X categories at age. For example, if Quality size is 300mm, an age-1 fish at 275mm at the start of the year would not be counted as a quality-sized fish, but an age-2 fish at 325mm at the start of the year would be counted as a quality-sized fish.

Author(s)

Jason C. Doll, <`jason.doll@fmarion.edu`>

See Also

`yprBH_MinLL` for estimating yield with a yield-per-recruit model using a minimum length limit and `yprBH_SlotLL` for estimating yield with the yield-per-recruit model and a slot limit.

See this demonstration page for more examples of this function.

Examples

```
#load required library
library(dplyr)
library(ggplot2)

# Example of simulating yield with the dynamic pool model,

lhparms <- makeLH(N0=100,tmax=30,Linf=1349.5,K=0.111,t0=0.065,
            LWalpha=-5.2147,LWbeta=3.153)
simyears <- 50
minLL <- 400
rec <- genRecruits(method = "fixed", nR = 100, simyears = simyears)
cm <- matrix(rep(c(rep(0,1), rep(0.18,(lhparms$tmax))), simyears),nrow=simyears,byrow=TRUE)
cf <- matrix(rep(c(rep(0,1), rep(0.33,(lhparms$tmax))), simyears),nrow=simyears,byrow=TRUE)

out<-dpmBH_MinLL(simyears = simyears, minLL = minLL, cf = cf,
                 cm = cm, rec = rec, lhparms = lhparms,
                 matchRicker=FALSE,species="Striped Bass",group="landlocked")

#Use summary by year data frame to plot yield vs year
ggplot(data=out[[2]],mapping=aes(x=year,y=Yield_age_1plus)) +
  geom_point() +
  geom_line() +
  labs(y="Total yield (g)",x="Year") +
  theme_bw()

#Plot date using summary by age
#filter for year class = 1
plotdat<- out[[1]] |> filter(yc==1)
#Plot yield vs age
ggplot(data=plotdat,mapping=aes(x=age,y=yield)) +
  geom_point() +
  geom_line() +
  labs(y="Total yield (g)",x="Age") +
  theme_bw()
```

```
#Recruitment based on a normal distribution
rec <- genRecruits(method = "normal", simyears = simyears,
                   meanR = 1000, sdR = 500, minR = 100, maxR =2500)
cm <- matrix(rep(c(rep(0,1), rep(0.18,(lhparms$tmax))), simyears),nrow=simyears,byrow=TRUE)
cf <- matrix(rep(c(rep(0,1), rep(0.33,(lhparms$tmax))), simyears),nrow=simyears,byrow=TRUE)

out_2<-dpmBH_MinLL(minLL = minLL, cf = cf, cm = cm,
                   rec = rec, lhparms = lhparms,simyears = simyears,
                   species="Striped Bass",group="landlocked",matchRicker=FALSE)

#Use summary by year data frame to plot yield vs year
ggplot(data=out_2[[2]],mapping=aes(x=year,y=PSD)) +
  geom_point() +
  geom_line() +
  labs(y="PSD",x="Year") +
  theme_bw()

#Plot date using summary by age
#Plot yield vs age for each year class
ggplot(data=out_2[[1]],mapping=aes(x=age,y=yield,group=yc,color=yc)) +
  geom_point() +
  geom_line() +
  labs(y="Total yield (g)",x="Age") +
  theme_bw()
```

`est_natmort` *Compute meta-analytic estimates of instantaneous and conditional natural mortality*

Description

Several methods may be used to estimate instantaneous (M) and conditional natural mortality (cm) from other types of data, especially those saved in the life history parameters vector/list from `makeLH`.

Usage

```
est_natmort(lhparms = NULL, method = "rFAMS", incl.avg = FALSE, ...)
```

Arguments

`lhparms`	A named vector or string returned by `lhparms`.
`method`	A string that indicates what methods to use to estimate M (see `metaM`).
`incl.avg`	A logical that indicates whether the average cm should be computed from the estimated M of all methods.
`...`	Option arguments for parameter values required by methods using parameters other than those in `lhparms`. See examples.

Details

The default `methods` to use are all of those listed in `Mmethods` that use some of the life history parameters required by `makeLH`. These methods are not all equally useful or robust, so the user may want to select a subset of them for use after learning more about them. See references in `metaM`.

Other methods that require parameters other than those required by `makeLH` can be used by providing the name of the method in `method` and the required parameters as arguments, as defined in `metaM`. See `metaM` for more details and the examples below for an example.

Value

A data.frame with the following items:

- `method`: The name for the method within the function (as given in `method`).
- `M`: The estimated instantaneous natural mortality rate (from `metaM`)
- `cm`: The estimated conditional natural mortality rate (computed directly from `M`).
- `givens`: A string that contains the input values required by the method to estimate M.

Author(s)

Derek Ogle

Examples

```
# An example lhparm as would be returned from makeLH
tmp <- list(N0=100,tmax=15,Linf=500,K=0.3,t0=-0.5,LWalpha=-5.16,LWbeta=3.1)

# All methods in metaM() that use those life history parameters
est_natmort(tmp)

# Same but including the average in the last row
est_natmort(tmp,incl.avg=TRUE)

# Selecting just one method
est_natmort(tmp,method="HoenigNLS")

# Selecting several methods
est_natmort(tmp,method=c("HoenigNLS","HoenigO","HoenigO2","HoenigLM"))

# A method that uses a parameter not usually in lhparms
est_natmort(tmp,method="QuinnDeriso",PS=0.05)

# Selecting all Hoenig methods using Mmethods from FSA
est_natmort(tmp,method=FSA::Mmethods("Hoenig"))

# Over-riding the Linf param in parameters list, but others from tmp
est_natmort(tmp,method="PaulyLNoT")              # Linf from tmp
est_natmort(tmp,Linf=1000/10,method="PaulyLNoT") # Linf from Linf= arg
```

---

genRecruits *Generate a vector of number of recruits for the dynamic pool model.*

Description

This function is used to generate number of recruits across multiple years using different random functions.

Usage

```
genRecruits(
  simyears,
  method = c("fixed", "uniform", "normal", "StrYC_Nth", "StrYC_randInt"),
  nR = NULL,
  minR = NULL,
  maxR = NULL,
  meanR = NULL,
  sdR = NULL,
  nStr = NULL,
  sizeStr = NULL,
  avgFreq = NULL
)

## S3 method for class 'GENREC'
print(x, ...)
```

Arguments

`simyears`	A single numeric that sets the number of years to simulate recruitment
`method`	A single string to call the method of generating a vector of recruits. `fixed` generates recruitment based on a fixed value for each year of simyears, `uniform` generates recruitment based on random values from a uniform distribution for each year of simyears, `normal` generates recruitment based on random values from a normal distribution for each year of simyears, `StrYC_Nth` generates recruitment based on a strong year class every Nth year, and `StrYC_randInt` generates recruitment based on a strong year classes at random intervals.
`nR`	A single numeric that sets the fixed number of recruitment. Used when `method="fixed"` or `method="StrYC_Nth"`.
`minR`	A single numeric that sets the minimum number of recruits during simulations. Used when `method="uniform"` or `method="normal"`.
`maxR`	A single numeric that sets the maximum number of recruits during simulations. Used when `method="uniform"` or `method="normal"`.
`meanR`	A single numeric that sets the mean number of recruits. Used when `method="normal"` or `method="StrYC_randInt"`.
`sdR`	A single numeric that sets the standard deviation of number of recruits. Used when `method="normal"`.

`nStr`	A single numeric that sets the Nth year that a strong year class will occur. Used when `method="StrYC_Nth"`.
`sizeStr`	A single numeric that sets the multiplier for the strong year class relative to meanR. Used when `method="StrYC_Nth"`, or `method="StrYC_randInt"`.
`avgFreq`	A single numeric that sets the average frequency of a strong year class. Used when `method="StrYC_randInt"`.
`x`	Object saved from `genRecruits`.
`...`	Optional arguments for `print`.

Value

A vector that contains the number of recruits for each simulation that can be used directly in the dynamic pool model (e.g., `dpmBH_MinLL`).

Author(s)

Jason C. Doll, <`jason.doll@fmarion.edu`>

Examples

```
# Generate recruits for 20 years based on a fixed number
rec <- genRecruits(simyears=20,method="fixed",nR=50)
rec

# Generate recruits for 20 years from a uniform distribution bound
# by 25 and 75
rec <- genRecruits(simyears=20,method="uniform",minR=25,maxR=75)
rec

# Generate recruits for 20 years based on a normal distribution with a mean
# of 50, standard deviation of 10, and trucated to be between 25 and 75
rec <- genRecruits(simyears=20,method="normal",minR=25,maxR=75,meanR=50,sdR=10)
rec

# Geneate recruits for 20 years based on a fixed number of recruits at 50 and
# a strong year class every 5 years with recruits 2 times the mean recruits
rec <- genRecruits(simyears=20,method="StrYC_Nth",nR=50,sizeStr=2,nStr=5)
rec

# Generate recruits for 20 years based on a fixed number of recruits at 50
# and a strong year class at random intervals of size 2 times the mean
# recruitswith the random interval averaging every 5 years.
rec <- genRecruits(simyears=20,method="StrYC_randInt",nR=50,sizeStr=2,avgFreq=5)
rec
```

`makeLH` *Make a list or vector of life history parameters for yield-per-recruit analyses.*

Description

Efficiently construct either a vector or list that contains the seven life history parameters required for Beverton-Holt yield-per-recruit analyses. The parameters can be given by the user through function arguments. Alternativvely, the von Bertalanffy parameters (`Linf`, `K`, and `t0`) may be extracted from an `nls` object created from fitting the von Bertalanffy equation to length-at-age data (object created outside this function). Similarly the log10-transformed weight-length model coefficients may be extracted from an `lm` object created from fitting the model to transformed weight-length data (object created outside this function). All parameter values are checked for sanity (e.g., Linf>0).

Usage

```
makeLH(N0, tmax, Linf, K, t0, LWalpha, LWbeta, restype = c("list", "vector"))

## S3 method for class 'MAKELH'
print(x, ...)
```

Arguments

`N0`	A single numeric that represents the number of fish in the population at the hypothetical age of `t0`.
`tmax`	A single whole number that represents maximum age in the population in years.
`Linf`	A single numeric that represents the point estimate of asymptotic mean length from the von Bertalanffy growth model OR an `nls` object created from fitting the von Bertalanffy equation to length-at-age data.
`K`	A single numeric that represents the point estimate of the Brody growth coefficient from the von Bertalanffy growth model.
`t0`	A single numeric that represents the point estimate of the x-intercept (i.e., theoretical age at a mean length of 0) from the von Bertalanffy growth model.
`LWalpha`	A single numeric that represents the point estimate of alpha from the length-weight regression on the log10 scale OR an `lm` object created from fitting the model to log10-transformed weight-length data.
`LWbeta`	A single numeric that represents the point estimate of beta from the length-weight regression on the log10 scale.
`restype`	A character that indicates the type of output (list or vector) returned by the function.
`x`	An object created by `makeLH`.
`...`	Optional arguments to be passed to `print`.

Details

Use of this function for putting life history parameters into a list or vector is recommended as (i) values for `Linf`, `K`, `t0`, `LWalpha`, and `LWbeta` can be extracted from objects from appropriate model fitting and (ii) checks for impossible or improbable values for each parameter are performed; i.e.,

```
# Best practice for entering life history parameter values
LH <- makeLH(N0=100,tmax=15,Linf=600,K=0.30,t0=-0.6,
             LWalpha=-5.453,LWbeta=3.10)

# Works but no checks on the values
LH <- list(N0=100,tmax=15,Linf=600,K=0.30,t0=-0.6,
           LWalpha=-5.453,LWbeta=3.10)
```

If a list is returned then values will be displayed with the number of decimals provided by the user. If a vector is returned then the number of decimals displayed will be the same for each value and will match the value supplied by the user with the most decimals. Thus, a list is preferred as it will be easier to match what was given to what was expected to be given.

Value

A named list or vector (depending on `restype`) that contains the given (or extracted) life history parameters values that can be used directly in the yield-per-recruit calculation functions (e.g., `yprBH_SlotLL`).

Author(s)

Derek Ogle

See Also

This demonstration page for further examples.

Examples

```
library(FSA)
library(FSAdata)
library(dplyr)

# ----- Simple examples with explicit arguments for each --------------------
makeLH(N0=100,tmax=15,Linf=500,K=0.3,t0=-0.5,LWalpha=-5.613,LWbeta=3.1)
makeLH(N0=100,tmax=15,Linf=500,K=0.3,t0=-0.5,LWalpha=-5.613,LWbeta=3.1,
       restype="vector")

# ----- Example of extracting values from model fits ------------------------
# N0 and tmax provided as arguments ... Linf, K, and t0 extracted from nls
#   output and LWalpha and LWbeta extracted from lm output. Note that nls
#   and lm output here are just examples of the function, they should be
#   calculated for the same species from the same waterbody, etc.

# Load data from FSAdata package, restrict to one location and year,
```

```
# create log10 values of weight and length
data(WalleyeErie2,package="FSAdata")
tmp <- WalleyeErie2 |>
  dplyr::filter(loc==2,year==2010) |>
  dplyr::mutate(logW=log10(w),
                logL=log10(tl))

# Generate LVB results
vb1 <- FSA::makeGrowthFun(type="von Bertalanffy")
fit1 <- nls(tl~vb1(age,Linf,K,t0),data=tmp,
            start=FSA::findGrowthStarts(tl~age,data=tmp))

# Generate length-weight regression results
fit2 <- lm(logW~logL,data=tmp)

# Make life-history list with those results
waeLH <- makeLH(N0=100,tmax=15,Linf=fit1,LWalpha=fit2)
waeLH
```

`seeMorts` *Convert vectors of conditional fishing and natural mortality rates to other mortality rates.*

Description

Convert vectors of conditional fishing (cf) and natural (cm) mortality rates to instantaneous total (Z), fishing (F), and natural (M) mortality rates, total annual mortality rate (A), the annual exploitation rate (u), and the expectation of natural death (v). The primary purpose of this function is to provide a data.frame from which the user can explore the relationships between these rates and understand how choices of cf and cm effect the other rates, especially A and u.

Usage

```
seeMorts(cf, cm, type = 2, verbose = TRUE)

## S3 method for class 'SEEMORTS'
summary(object, verbose = TRUE, ...)
```

Arguments

`cf`	A numeric vector (could be of length 1) representing conditional fishing mortality. See details.
`cm`	A numeric vector (could be of length 1) representing conditional natural mortality. See details.
`type`	A single numeric that identifies whether the annual exploitation rate (u) and the expectation of natural death (v) should be computed for a type-2 (DEFAULT) or type-1 fishery (as defined by Ricker (1975); see details).

`verbose`	A logical that indicates whether a brief note should be printed to the console. Defaults to `TRUE`.
`object`	An object returned by `seeMorts`.
`...`	Arguments to be forwarded to `summary`.

Details

Numeric values in the `cf` and `cm` vectors can be entered as a single value (e.g., `cf=0.3`), a sequence of values created with `seq` (e.g., `cf=seq(0.1,0.5,0.05)`, or as unique values with `c` (e.g., `cf=c(0.1,0.4,0.5)` depending on the user's needs. Values of `cf` and `cm` will be repeated as necessary (via `expand.grid`) to form all combinations of the two sets of given values. Thus, neither `cf` and `cm` should contain repeated values.

Equations for computing the other mortality rates (F, M, Z, A, u, and v) from cf and cm are in Ricker (1975). Note that n and m in Ricker (1975) are cf and cm here.

The formulae for u and v differ depending on whether a Type-1 or a Type-2 fishery is being considered (see `type`). A Type-1 fishery is where fishing mortality occurs in a very narrow part of the annual period such that it is reasonable to assume that fishing and natural mortality do not both occur (or overlap) in that portion (e.g., a fishery where the open harvest season is only a few days). A Type-2 fishery is where natural and fishing mortality substantially overlap throughout the annual period (e.g., a fishery where the open harvest season is much of the annual period).

Value

The main function returns a data.frame with the following values:

- `cm` is the given conditional natural mortality rates.
- `cf` is the given conditional fishing mortality rates.
- `M` is the calculated instantaneous rate of natural mortality.
- `F` is the calculated instantaneous rate of fishing mortality.
- `Z` is the calculated instantaneous rate of total mortality.
- `A` is the calculated total annual rate of mortality.
- `u` is the calculated annual exploitation rate.
- `v` is the calculated expectation of natural death.

The `summary` function returns a data.frame with the following values for each of the mortality rates:

- `type` is the "type" of mortality rate (cm, cf, M, F, Z, A, u, or v).
- `unique` is the number of unique values.
- `min` is the minimum value (rounded to 3 decimal places).
- `max` is the maximum value (rounded to 3 decimal places).

References

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. Was (is?) from `https://waves-vagues.dfo-mpo.gc.ca/library-bibliotheque/1485.pdf`.

See Also

`yprBH_MinLL`, `yprBH_SlotLL`, and `dpmBH_MinLL` for functions that require the user to provide reasonable values of cf and cm.

Examples

```
# == Simple examples ===================================================
seeMorts(cf=0.3,cm=0.2)
seeMorts(cf=0.3,cm=0.2,type=1)

# == More realistic example ============================================
test <- seeMorts(cf=seq(0,0.5,0.05),cm=c(0.2,0.3,0.4,0.5))
head(test)
tail(test)
summary(test)

#-- Optional plotting examples ---------------------------------------------
if (require(ggplot2)) {
  ggplot(data=test,mapping=aes(x=cf,y=u,color=as.factor(cm))) +
    geom_line(linewidth=1) +
    theme_bw()

  ggplot(data=test,mapping=aes(x=Z,y=A)) +
    geom_line(linewidth=1) +
    theme_bw()

  ggplot(data=test,mapping=aes(x=cf,y=cm,z=A)) +
    geom_contour_filled(bins=9) +
    scale_fill_discrete(name="A",palette="OrRd") +
    theme_bw()
}
```

`yprBH_MinLL` *Simulate expected yield under minimum length regulations using the Beverton-Holt Yield-per-Recruit model for a range of input parameters*

Description

Simulate yield under minimum length regulations using the Beverton-Holt Yield-per-Recruit (YPR) model with (possibly) multiple values for minimum length limits for harvest (`minLL`), conditional fishing mortality (`cf`), and conditional natural mortality (`cm`).

Usage

```
yprBH_MinLL(minLL, cf, cm, lhparms, loi = NULL, matchRicker = FALSE)
```

Arguments

`minLL`	A numeric vector of minimum length limits (in mm). All values must be less than `Linf` in `lhparms`.
`cf`	A numeric vector of conditional fishing mortality. All values must be between 0 and 1 (inclusive).
`cm`	A numeric vector of conditional natural mortality. All values must be between 0 and 1 (inclusive).
`lhparms`	A named vector or list that contains values for each `N0`, `tmax`, `Linf`, `K`, `t0`, `LWalpha`, and `LWbeta`. See `makeLH` for definitions of these life history parameters. Also see details.
`loi`	A numeric vector of lengths (in mm) of interest. Used to determine number of fish that reach these lengths. All must be less than `Linf` in `lhparms`.
`matchRicker`	A logical that indicates whether the yield function should match that in Ricker (1975). Defaults to `FALSE`. See the FAMS vs Ricker article.

Details

Details will be filled out later.

Note that the main calculations are in the internal `yprBH_func` (use `rFAMS:::yprBH_func` to see that source code).

Value

A data.frame with the following calculated values:

- `yield` is the estimated yield (in g).
- `exploitation` is the exploitation rate.
- `Nharvest` is the number of harvested fish.
- `Ndie` is the number of fish that die of natural deaths.
- `Nt` is the number of fish at time tr (time they become harvestable size).
- `avgwt` is the average weight of fish harvested.
- `avglen` is the average length of fish harvested.
- `tr` is the time for a fish to recruit to a minimum length limit (i.e., time to enter fishery).
- `nAtxxx` is the number that reach the length of interest supplied. There will be one column for each length of interest.
- `F` is the instantaneous rate of fishing mortality.
- `M` is the instantaneous rate of natural mortality.
- `Z` is the instantaneous rate of total mortality.
- `S` is the (total) annual rate of survival.

For convenience the data.frame also contains the model input values (`minLL`, `cf`, and`cm` from input vectors; `N0`; `Linf`; `K`; `t0`; `LWalpha`; `LWbeta`; and `tmax` from `lhparms`).

The data.frame also contains a `notes` value which may contain abbreviations for "issues" that occurred when computing the results and were adjusted for. The possible abbreviates are as follows:

- `minLL>=Linf`: The minimum length limit (`minLL`) being explored was greater than the given asymptotic mean length (`Linf`). For the purpose (only) of computing the time at recruitment to the fishery (`tr`) the `Linf` was set to `minLL`+0.1.
- `tr<t0`: The age at recruitment to the fishery (`tr`) was less than the hypothetical time when the mean length is zero (`t0`). The fish can't recruit to the fishery prior to having length 0 so `tr` was set to `t0`. This also assures that the time it takes to recruit to the fishery is greater than 0.
- `Nt<0`: The number of fish recruiting to the fishery was less than 0. There cannot be negative fish, so `Nt` was then set to 0.
- `Nt>N0`: The number of fish recruiting to the fishery was more than the number of fish recruited to the populations. Fish cannot be added to the cohort, so `Nt` was set to `N0`.
- `Y=Infinite`: The calculated yield (`Y`) was inifinity, which is impossible and suggests some other problem. Yield was set to `NA`.
- `Y<0`: The calculated yield (`Y`) was negative, which is impossible. Yield was set to 0.
- `Nharv<0`: The calculated number of fish harvested (`Nharv`) was negative, which is not possible. Number harvested was set to 0.
- `Nharv>Nt`: The calculated number of fish harvested (`Nharv`) was greater than the number of fish recruiting to the fishery, which is impossible. The number harvested was set to the number recruiting to the fishery.
- `Ndie<0`: The calculated number of fish recruiting to the fishery that died naturally (`Ndie`) was negative, which is not possible. Number that died was set to 0.
- `Ndie>Nt`: The calculated number of fish recruiting to the fishery that died naturally (`Ndie`) was greater than the number of fish recruiting to the fishery, which is impossible. The number that died was set to the number recruiting to the fishery.
- `agvglen<minLL`: The average length of harvested fish was less than the given minimum length limit being explored, which is not possible (with only legal harvest). The average length was set to the minimum length limit.

Author(s)

Jason C. Doll, `<jason.doll@fmarion.edu>`

See Also

`yprBH_SlotLL` for estimating yield with the yield-per-recruit model and a slot limit, or `dpmBH_MinLL` for estimating yield with a dynamic pool model using a minimum length limit.

See this demonstration page for more examples of this function.

Examples

```
# Load other required packages for organizing output and plotting
library(dplyr)    ## for filter
library(ggplot2)  ## for ggplot et al.
library(metR)     ## geom_contour2

# Life history parameters to be used below
LH <- makeLH(N0=100,tmax=15,Linf=592,K=0.20,t0=-0.3,LWalpha=-5.528,LWbeta=3.273)
```

```
# Estimate yield for multiple values of minLL, cf, and cm
# # This is a minimal example, increments for minLL, cf, and cm would likely be smaller
# #   to produce finer-scaled results.

minLL <- seq(from = 200, to = 550, by = 50)
cf <- seq(from = 0.1, to = 0.9, by = 0.1)
cm <- seq(from = 0.1, to = 0.9, by = 0.1)
loi <- c(400,450,500,550)

Res_1 <- yprBH_MinLL(minLL = minLL, cf = cf, cm = cm,
                     lhparms=LH, loi=loi)

# Yield curves (yield vs exploitation) by varying minimum lengths,
# using cm=40
plot_dat <- Res_1 |> filter(cm==0.40)

ggplot(data=plot_dat,mapping=aes(y=yield,x=exploitation,
                                 group=minLL,color=minLL)) +
  geom_line(linewidth=1) +
  scale_color_gradient2(high="black") +
  xlab("Exploitation (u)")+
  ylab("Yield (g)")+
  labs(color="Min Length Limit") +
  theme_bw()

# Yield isopleths for varying minLL and exploitation with cm=0.40
# # Using same data as previous example
ggplot(data=plot_dat,mapping=aes(x=exploitation,y=minLL,z=yield)) +
  geom_contour2(aes(label = after_stat(level))) +
  xlab("Exploitation (u)") +
  ylab("Minimum length limit (mm)") +
  theme_bw()
```

`yprBH_SlotLL` *Simulate expected yield using below slot limit regulations using the Beverton-Holt Yield-per-Recruit model*

Description

Simulate yield below slot length regulations using the Beverton-Holt Yield-per-Recruit (YPR) model with (possibly) multiple values for conditional natural mortality (`cm`) and chosen values for the lower and upper lengths of the slot (i.e,. `lowerSL` and `upperSL`); conditional fishing mortality below (`cfBelow`), in (`cfIn`), and above (`cfAbove`) the slot; and length when fish recruit to the fishery (`recruitmentTL`).

Usage

```
yprBH_SlotLL(
```

```
  lowerSL,
  upperSL,
  cfBelow,
  cfIn,
  cfAbove,
  cm,
  lhparms,
  recruitmentTL = NULL,
  loi = NULL,
  matchRicker = FALSE,
  label = NULL
)
```

Arguments

`lowerSL`	A single numeric representing the length of the lower slot limit in mm. See details. Must be less than `upperSL`.
`upperSL`	A single numeric representing the length of the upper slot limit in mm. See details. Must be less than `Linf` in `lhparms`.
`cfBelow`	A single numeric representing conditional fishing mortality below the lower slot limit length. Must be between 0 and 1 (inclusive).
`cfIn`	A single numeric representing conditional fishing mortality between the lower and upper slot limit lengths (i.e., "in the slot"). Must be between 0 and 1 (inclusive).
`cfAbove`	A single numeric representing conditional fishing mortality above the upper slot limit length. Must be between 0 and 1 (inclusive).
`cm`	A numeric vector of conditional natural mortality values. All values must be between 0 and 1 (inclusive).
`lhparms`	A named vector or list that contains values for each `N0`, `tmax`, `Linf`, `K`, `t0`, `LWalpha`, and `LWbeta`. See `makeLH` for definitions of these life history parameters. Also see details.
`recruitmentTL`	A single numeric that represents the minimum length (in mm) for recruiting to the fishery. Cannot be greater than `lowerSL`.
`loi`	A numeric vector of lengths (in mm) of interest. Used to determine number of fish that reach these lengths. All must be less than `Linf` in `lhparms`.
`matchRicker`	A logical that indicates whether the yield function should match that in Ricker (1975). Defaults to `TRUE`. The only reason to changed to `FALSE` is to try to match output from FAMS. See the FAMS vs Ricker article.
`label`	An optional string to label the type of slot limit being simulated.

Details

Details will be filled out later.

Note that the main calculations are in the internal `yprBH_slot_func` (use `rFAMS:::yprBH_slot_func` to see that source code).

Value

A data.frame with the following calculated values:

- `yieldTotal` is the calculated total yield
- `yieldBelow` is the calculated yield below the slot limit
- `yieldIn` is the calculated yield within the slot limit
- `yieldAbove` is the calculated yield above the slot limit
- `nharvTotal` is the calculated total number of harvested fish
- `ndieTotal` is the calculated total number of fish that die of natural death
- `nharvestBelow` is the number of harvested fish below the slot limit
- `nharvestIn` is the number of harvested fish within the slot limit
- `nharvestAbove` is the number of harvested fish above the slot limit
- `n0die` is the number of fish that die of natural death before entering the fishery at a minimum length
- `ndieBelow` is the number of fish that die of natural death between entering the fishery and the lower slot limit
- `ndieIn` is the number of fish that die of natural deaths within the slot limit
- `ndieAbove` is the number of fish that die of natural deaths above the slot limit
- `nrBelow` is the number of fish at time trBelow (time they become harvestable size below the slot limit)
- `nrIn` is the number of fish at time trIn (time they reach the lower slot limit size)
- `nrAbove` is the number of fish at time trAbove (time they reach the upper slot limit size)
- `trBelow` is the time for a fish to recruit to a minimum length limit (i.e., time to enter fishery)
- `trIn` is the time for a fish to recruit to a lower length limit of the slot limit
- `trOver` is the time for a fish to recruit to a upper length limit of the slot limit
- `avglenBelow` is the average length of fish harvested below the slot limit
- `avglenIn` is the average length of fish harvested within the slot limit
- `avglenAbove` is the average length of fish harvested above the slot limit
- `avgwtBelow` is the average weight of fish harvested below the slot limit
- `avgwtIn` is the average weight of fish harvested within the slot limit
- `avgwtAbove` is the average weight of fish harvested above the slot limit
- `nAtxxx` is the number that reach the length of interest supplied. There will be one column for each length of interest.
- `cm` A numeric representing conditional natural mortality
- `expBelow` is the exploitation rate below the slot limit
- `expIn` is the exploitation rate within the slot limit
- `expAbove` is the exploitation rate above the slot limit
- `FBelow` is the estimated instantaneous rate of fishing mortality below the slot limit
- `FIn` is the estimated instantaneous rate of fishing mortality within the slot limit

- `FAbove` is the estimated instantaneous rate of fishing mortality above the slot limit
- `MBelow` is the estimated instantaneous rate of natural mortality below the slot limit
- `MIn` is the estimated instantaneous rate of natural mortality within the slot limit
- `MAbove` is the estimated instantaneous rate of natural mortality above the slot limit
- `ZBelow` is the estimated instantaneous rate of total mortality below the slot limit
- `ZIn` is the estimated instantaneous rate of total mortality within the slot limit
- `ZAbove` is the estimated instantaneous rate of total mortality above the slot limit
- `SBelow` is the estimated total survival below the slot limit
- `SIn` is the estimated total survival within the slot limit
- `SAbove` is the estimated total survival above the slot limit

For convenience the data.frame also contains the model input values (`lowerSL`, `upperSL`, `cfBelow`, `cfIn`, `cfOver`, `cm` from input vectors; `N0`; `Linf`; `K`; `t0`; `LWalpha`; `LWbeta`; and `tmax` from `lhparms`) and, optionally, the string provided in `label`.

Author(s)

Jason C. Doll, <`jason.doll@fmarion.edu`>

See Also

`yprBH_MinLL` for estimating yield with the yield-per-recruit model using a minimum length limits, or `dpmBH_MinLL` for estimating yield with a dynamic pool model using a minimum length limit.

See this demonstration page for more examples of this function.

Examples

```
#Load other required packages for organizing output and plotting
library(ggplot2)  #for plotting
library(dplyr)    #for select
library(tidyr)    #for pivot_longer

# Life history parameters to be used below
LH <- makeLH(N0=100,tmax=15,Linf=592,K=0.20,t0=-0.3,LWalpha=-5.528,LWbeta=3.273)
# conditional natural mortality vector
cm <- seq(from = 0.1, to = 0.9, by = 0.1)

# Estimate yield based on a protected slot limit
Res_1 <- yprBH_SlotLL(lowerSL=250,upperSL=325,
                      cfBelow=0.25,cfIn=0.0,cfAbove=0.15,cm=cm,
                      lhparms=LH,recruitmentTL=200,
                      loi=c(200,250,300,325,350),label="250-325")

Res_1

# Plot results
# Total Yield vs Conditional Natural Mortality (cm)
ggplot(data=Res_1,mapping=aes(x=cm,y=yieldTotal)) +
```

```
  geom_point() +
  geom_line() +
  labs(y="Total Yield (g)",x="Conditional Natural Mortality (cm)") +
  theme_bw()


# Yield below, in, and above the slot limit vs Conditional Natural Mortality (cm)
# Select columns for plotting
plot_data <- Res_1 |>
  select(cm, yieldBelow, yieldIn, yieldAbove) |>
  pivot_longer(!cm, names_to="YieldCat",values_to="Yield")

# Generate plot
ggplot(data=plot_data,mapping=aes(x=cm,y=Yield,group=YieldCat,color=YieldCat)) +
  geom_point() +
  scale_color_discrete(name="Yield",labels=c("Above SL","In SL","Below SL"))+
  geom_line() +
  labs(y="Total Yield (g)",x="Conditional Natural Mortality (cm)") +
  theme_bw() +
  theme(legend.position = "top")+
  guides(color=guide_legend(title="Yield"))
```

Index