Package ‘optistock’

August 24, 2023

Type Package

Title Determine Optimum Stocking Times Used in Fishery Enhancements

Version 0.0.2

Maintainer Paul Frater <`paul.frater@wisconsin.gov`>

Description A collection of functions that aid in calculating the optimum time to stock hatchery reared fish into a body of water given the growth, mortality and cost of raising a particular number of individuals to a certain length.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0), knitr, rmarkdown, tidyverse, shiny, shinydashboard

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Paul Frater [aut, cre] (<`https://orcid.org/0000-0002-7237-6563`>)

Repository CRAN

Date/Publication 2023-08-24 14:40:05 UTC

R topics documented:

`cost_parameters` *Cost parameters for species used in examples*

Description

This data.frame contains the cost parameters used in `spp_examples`. These parameters correspond with the `total_cost` and `linear_total_cost` functions.

Usage

```
cost_parameters
```

Format

A `tibble` of variables and 16 records

spp Species common name

source The source of the data. WDNR is from Wisconsin Dep't of Natural Resources Hatchery cost data. AFS is the American Fisheries Society Special Publication 35 on Fishkill Replacement costs

cost_fun_type Either "exp" for exponential – corresponds to the `total_cost` cost function, or "linear" – corresponds to the `linear_total_cost` cost function.

cost_fun The cost function – either `total_cost` or `linear_total_cost`

cost_fun_params A list-col of the parameters necessary for the respective cost function found in `cost_fun` `cost_per_fish` *Compute the per-cost fish based on stocking time, time to recruitment, growth, and mortality*

Description

Compute the per-cost fish based on stocking time, time to recruitment, growth, and mortality

Usage

```
cost_per_fish(
  time_at_stocking,
  time_at_rec,
  n_recruits_desired,
  cost_fun = total_cost,
  cost_fun_args,
  mort_fun = constant_mort,
  mort_fun_args
)
```

Arguments

`time_at_stocking`	The time at which fish are stocked (i.e. synonymous with the amount of time that fish are raised in a hatchery)
`time_at_rec`	The time at which a fish enters the fishery (i.e. the amount of time it takes a fish to grow to a desired length). Use `inv_vb` to calculate this.
`n_recruits_desired`	The number of recruits desired at `time_at_rec`
`cost_fun`	The cost function. Defaults to `total_cost`
`cost_fun_args`	Arguments for `cost_fun`
`mort_fun`	The mortality function, see ?mort_funs
`mort_fun_args`	List. Named arguments to be passed to `mort_fun`

Value

The per-fish cost fish that lives until `time_at_rec` based on `time_at_stocking`, the cost function and mortality functions.

Examples

```
cost_args <- list(
    init_cost = 0.05,
    time_slope = 0.01, time_exp = 1.2,
    rec_exp = 1
)
```

```
mort_args <- list(m_init = (1 / 365))
# the cost-per-fish to stock across a range of times given cost and mortality
# assumes fish recruit into the fishery at day 1000
curve(cost_per_fish(
    x, 1000, 1000,
    cost_fun_args = cost_args,
    mort_fun_args = mort_args),
  xlab = "Days", ylab = "$ per fish stocked",
  10, 1200
)
```

daily_cost_fun — *Compute the instantaneous cost of raising hatchery fish*

Description

This is a multivariable function of both time and number of recruits raised. Cost-per-time and cost-per-recruit can be calculated as a quadratic where the slope and exponent can be specified.

Usage

```
daily_cost_fun(
  time,
  recruits,
  daily_cost,
  time_slope = 0,
  time_exp = 1,
  rec_slope = 1,
  rec_exp = 1,
  type = "multiplicative"
)
```

Arguments

`time`	The time at which fish are raised in hatchery
`recruits`	The number of recruits raised
`daily_cost`	Baseline daily cost to raise a single fish
`time_slope`	The slope term on the amount of time (see details)
`time_exp`	The exponent on the amount of time
`rec_slope`	The slope term on the number of recruits
`rec_exp`	The exponent on the number of recruits
`type`	Either multiply the number of recruits times the cost-at-time or add to it (see Details).

Details

The cost-per-fish based on time and number of recruits uses the function:

$$C = s_1 T^{\alpha} \cdot s_2 R^{\beta} + b$$

if type = "multiplicative". Otherwise it uses:

$$C = s_1 T^{\alpha} + s_2 R^{\beta} + b$$

if type = "additive"

where C = the cost to rear R number of recruits at time T, the s values are the slopes, α and β, are the exponents on time (T) and recruits (R), respectively, and b is the intercept. The instantaneous cost is really what is of interest, and the number of recruits essentially adjusts the intercept on that dimension of the equation.

Increasing the exponent will dramatically increase the cost of raising hatchery fish as time goes on. Increasing the exponent dramatically increases the cost of raising a greater number of fish. Integrating this equation across time will compute the total cost to raise the number of `recruits` to time T. Use the `total_daily_cost` function to do this automatically.

Value

A numeric value representing the cost of rearing the number of `recruits` at a given time and given the number of recruits raised

Examples

```
# compute the instantaneous cost of raising 1000 fish on day 100
daily_cost_fun(time = 100, recruits = 1000, daily_cost = 0.05,
         time_slope = 0, time_exp = 1,
         rec_slope = 0.01, rec_exp = 1)
# plot a curve of instantaneous cost against time
curve(daily_cost_fun(x, 1000, 0.05, 0.01, 1.2, 0.05, 1), 0, 1000,
      xlab = "Time", ylab = "$")
## Not run:
# 3d plot of costs by time and recruit
emdbook::curve3d(daily_cost_fun(x, y, 0.05, 0.01, 1.2, 0.05, 1),
                 from = c(0, 0),
                 to = c(1000, 1000),
                 xlab = "Time", ylab = "Recruits",
                 zlab = "$", sys3d = "wireframe")

## End(Not run)
```

growth_parameters *Growth parameters for species used in examples*

Description

This data.frame contains the growth parameters used in spp_examples. The growth parameters correspond with the von Bertalanfy growth curve (VBGF – see vbgf)

Usage

```
growth_parameters
```

Format

A data.frame with 7 fields and 6 records:

spp Species common name

latin Scientific name for the species

linf The L_infinity parameters for the VBGF

k The k parameter for the VBGF

t0 The t_0 parameter for the VBGF

n Number of samples used. For WDNR data this is the number of paired length-at-age data points (WDNR, 2021). For FishBase it is the number of submitted entries.

source Where data was retrieved from. WDNR is the Wisconsin Dep't of Natural Resources Fisheries Management Database. FishBase (Froese and Pauly, 2010) is FishBase.

inv_vb *The inverse von Bertalanffy function (iVBGF)*

Description

This function calculates the inverse of the VBGF, or, time it takes to grow to a particular length

Usage

```
inv_vb(len, linf, k, t0)
```

Arguments

len	Numeric. A length at which to determine how long it takes to grow
linf	The L_∞ parameter of the VBGF
k	The k parameter of the VBGF
t0	The t_0

Value

A numeric vector of how long it takes to grow to length `len`

Examples

```
time <- 365
len_at_age <- vbgf(time, 30, (0.2 / 365), -0.2)
inv_vb(len_at_age, 30, (0.2/365), -0.2)
```

`linear_total_cost` *Compute total cost as a linear function of time*

Description

This function returns the total cost of raising n `recruits` to `time`. The curve across `time` can only be linear with parameters `int` and `beta`, but can be non-linear with respect to `recruits`

Usage

```
linear_total_cost(time, recruits, int, beta, rec_exp = 1)
```

Arguments

`time`	The amount of time that fish are raised in hatchery
`recruits`	The number of recruits raised
`int`	Intercept for the linear total cost curve
`beta`	Slope for the linear total cost curve
`rec_exp`	The exponent on the number of recruits

Value

A vector the same length as `time` with the total cost to raise n `recruits` to `time`

Examples

```
curve(linear_total_cost(x, 0.5, 0.001, 100), 0, 1000)
```

mort_funs *Functions to produce mortality curves*

Description

This family of functions produce different shapes of mortality curves across time

Usage

```
exp_mort(time, m_init, m_inf, alpha, t_scale = NULL)

decreasing_mort(time, m_init, m_inf, alpha)

constant_mort(time, m_init)

inv_mort(time, m_init, m_inf)

gaussian_mort(time, m_init, m_max, t_scale, alpha)

half_gaussian_mort(time, m_init, m_max, m_inf, t_scale, alpha)

linear_mort(time, alpha, m_init)

parabolic_mort(time, m_min, alpha, t_scale, beta)
```

Arguments

`time`	The time to calculate mortality at
`m_init`	Initial rate of mortality at time 0 (or time t for `constant_mort`)
`m_inf`	Final rate of mortality as time approaches infinity
`alpha`	The rate at which mortality decreases across time
`t_scale`	A horizontal scaling parameter
`m_max`	The maximum mortality that is achieved at `time` = `t_scale`
`m_min`	The lowest mortality that the curve should reach
`beta`	Slope on the quadratic term for `parabolic_mort`

Details

These functions produced different shapes of mortality curves that are commonly found in fisheries. Some of the more common are `constant_mort` (which returns constant mortality across time), `exp_mort` (S-shaped decreasing curve), and `decreasing_mort` (non-linear decreasing curve). Others are less common and represent specific scenarios such as `gaussian_mort` (implemented to represent a bottleneck).

Value

A vector of numeric values for mortality rate at `time`

Examples

```
# an example in years
curve(exp_mort(x, 0.2, 0.1, 0.25), 0, 20)
# an example in days
curve(exp_mort(x, (1 / 365), (0.2 / 365), 0.005), 0, 1000)
```

---

n_to_stock — *Calculate the number of fish to stock based on desired recruit number and given mortality curve*

Description

This function is essentially the inverse of `recruits_at_time`. Given the number of fish desired at a certain time and the mortality function and parameters this function will calculate how many fish should be stocked into a system.

Usage

```
n_to_stock(
  time_at_stocking,
  time_at_rec,
  n_recruits_desired,
  mort_fun = exp_mort,
  mort_fun_args
)
```

Arguments

`time_at_stocking`	The time that fish are stocked (i.e. synonymous with the amount of time that fish are raised in a hatchery)
`time_at_rec`	The time at which a fish enters the fishery (i.e. the amount of time it takes a fish to grow to a desired length). Use `inv_vb` to calculate this.
`n_recruits_desired`	The number of recruits desired at `time_at_rec`
`mort_fun`	The mortality function, see ?mort_funs
`mort_fun_args`	List. Named arguments to be passed to `mort_fun`

Value

The number of fish to be stocked at `time_at_stocking` to get the desired number of fish at `time_at_rec` based on the mortality function and associated parameters

Examples

```
# how many fish to stock on day 100 if you want 10000 fish on day 1000
n_to_stock(10000, 100, 1000,
           mort_fun = exp_mort,
           mort_fun_args = list(m_init = (1 / 365),
                                m_inf = (0.2/365),
                                alpha = 0.005))
```

optistock_app — *Run Shiny app to create sandbox optistock CPF curves*

Description

This function will open a Shiny app where you can play around with parameters to see how the resulting CPF curve will change.

Usage

```
optistock_app()
```

Value

NULL. Opens and runs the Shiny application that comes with the `optistock` package

recruits_at_time — *Calculate the number of recruits left after given time based on mortality*

Description

This function will use the provided mortality function and parameters along with the length of time from stocking until the time in question to determine how many fish will be left at that time (i.e. how many fish die between `time_at_stocking` and `time_at_rec`).

Usage

```
recruits_at_time(
  time_at_stocking,
  time_at_rec,
  fish_init,
  mort_fun = exp_mort,
  mort_fun_args
)
```

Arguments

`time_at_stocking`	The day that fish are stocked (i.e. synonymous with the amount of time that fish are raised in a hatchery)
`time_at_rec`	The time at which a fish enters the fishery (i.e. the amount of time it takes a fish to grow to a desired length). Use `inv_vb` to calculate this.
`fish_init`	The initial number of fish stocked
`mort_fun`	The mortality function, see ?mort_funs
`mort_fun_args`	List. Named arguments to be passed to `mort_fun`

Details

This function calculates how many fish are left at a certain time based on the initial number of fish stocked and the integral of the mortality function. The number of fish left are computed using the following equation:

$$N_t = N_0 \exp \int_{T=0}^{t} f(t)$$

where

$$N_0$$

is the initial number of fish stocked and f(t) is the mortality function. The amount of time is provided to the function as the time at which fish are recruited into the fishery minus the time at which they are stocked. The time at which fish are recruited into the fishery can be calculated using the inverse von Bertalanffy growth function (see `inv_vb`).

Value

The number of fish that will be left given the mortality function, its parameters, and the time (`time_at_rec - time_at_stocking`)

Examples

```
mort_args <- list(
  m_init = (1 / 365),
  m_inf = (0.2/365),
  alpha = 0.005
)
recruits_at_time(100, 1000, 1000,
                 mort_fun = exp_mort,
                 mort_fun_args = mort_args)
```

`spp_examples` *Species examples from optistocking manuscript*

Description

This family of functions opens R scripts to run the scenarios that are used in the optistocking paper. Simply call the function to open the file that contains R code for the species' of interest.

Usage

```
walleye_example()

musky_example()

rainbow_trout_example()

chinook_example()
```

Value

NULL. Opens an R script with an example

Examples

```
## Not run:
walleye_example()

## End(Not run)
```

`total_cost` *Compute direct total cost to raise hatchery fish*

Description

This function computes the total cost to raise fish in a hatchery until `time`. This function differs from `total_daily_cost` by directly computing the total cost rather than integrating a daily cost estimate.

Usage

```
total_cost(
  time,
  time_slope = 1,
  time_exp = 1,
  init_cost = 0,
  recruits = 1,
  rec_exp = 1
)
```

Arguments

`time`	The amount of time that fish are raised in hatchery
`time_slope`	Controls how quickly the slope increases over time
`time_exp`	Controls the non-linearity of the curve over time
`init_cost`	The initial cost (i.e. intercept of the curve)
`recruits`	The number of recruits
`rec_exp`	Controls the non-linearity of the curve across recruit number

Details

The `total_cost` function computes a cost curve according to the following equation:

$$C = \alpha * T^{\gamma} + \beta + R^{\tau}$$

where α corresponds to the `time_slope` argument, γ is the `time_exp` parameter, β is the intercept (or `init_cost`), R is the number of recruits, and τ is the recruitment exponent corresponding to `rec_exp`

Value

A vector of values representing cost for the given time, recruit number, and associated variables

Examples

```
curve(total_cost(x, time_slope = 0.05, time_exp = 1.2), 0, 100)
curve(total_cost(x, time_slope = 0.05, time_exp = 0.5), 0, 100)
```

`total_daily_cost` *Compute the total daily cost of raising hatchery fish*

Description

This function takes the definite integral from time t = 0 until the given `time` of the `daily_cost_fun`. This integral is then the total cost of raising x number of fish until `time` given the other cost function parameters.

Usage

```
total_daily_cost(
  time,
  recruits,
  daily_cost,
  init_cost = 0,
  time_slope = 0,
  time_exp = 1,
  rec_slope = 1,
  rec_exp = 1,
  type = "multiplicative"
)
```

Arguments

`time`	The time at which fish are raised in hatchery
`recruits`	The number of recruits raised
`daily_cost`	Baseline daily cost to raise a single fish
`init_cost`	An intercept on the total cost function
`time_slope`	The slope term on the amount of time (see details)
`time_exp`	The exponent on the amount of time
`rec_slope`	The slope term on the number of recruits
`rec_exp`	The exponent on the number of recruits
`type`	Either multiply the number of recruits times the cost-at-time or add to it (see Details).

Value

The total cost across time to raise the number of `recruits`. This is simply the integral from time t = 0 until `time` of the `daily_cost_fun` function.

Examples

```
# total cost of raising 1000 fish for 100 days at given parameters
total_daily_cost(time = 100, recruits = 100,
          daily_cost = 0.05,
          time_slope = 0.01, time_exp = 1.2,
          rec_slope = 0.05, rec_exp = 1)
```

`vbgf` *Basic von Bertalanffy growth function (VBGF)*

Description

Basic von Bertalanffy growth function (VBGF)

Usage

```
vbgf(time, linf, k, t0)
```

Arguments

`time`	Time at which to calculate size
`linf`	The L_∞ parameter of the VBGF
`k`	The k parameter of the VBGF
`t0`	The t_0

Value

A numeric vector of lengths given the age (or amount of time) and parameters

Examples

```
curve(vbgf(x, 30, 0.25, -0.2), 0, 10)
curve(vbgf(x, 30, (0.25 / 365), -0.2), 0, 10 * 365)
```

Index