

Package: FLRef (via r-universe)

October 9, 2026

Title Reference point computation for advice rules

Version 1.14

Maintainer Henning Winker <Henning.Winker@fao.org>

Description Blah

Depends R (>= 4.0), FLCore, ggplotFL, FLBRP, FLasher, mse, FLSRTMB

Imports methods, ggplot2, mvtnorm

License EUPL

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Repository <https://ices-tools-prod.r-universe.dev>

Date/Publication 2026-10-09 08:07:07 UTC

RemoteUrl <https://github.com/Henning-Winker/FLRef>

RemoteRef HEAD

RemoteSha 247f61c3cff90f7c8c5714157fbf950f8e42be92

Contents

ABItgt	4
ALK	5
alk.sample	5
ALKs	6
applyALK	7
applyAR	7
asem2spm	8
bioidx.sim	11
bisect	11
blag	14
build_gear	15
ca.sim	17
catch_by_gear	18

catch_gear	19
catch_n_gear	20
combine_sel_age	21
computeFbrp	22
computeFbrps	24
condition_alk	25
f_age_gear	25
f_share_gear	26
Fbrp	27
Fe40	27
flr2stars	28
Fmmy	28
Fp05	29
Fsim	30
Fsim_grid	31
fudc	33
fwd2ices	34
fwd2stars	35
fwd2type1	35
fwdF4B	36
gear_ialk	37
gear_par	38
getF	38
huecol	39
iALK	39
idx.sim	40
jabba2FLStockR	41
jabba2stars	42
joint_sel	42
len.sim	43
len_grid	45
lfd.sim	45
lfd.sim.season	47
lfd_sop	48
merge_catch	49
Mlorenzen	50
newselex	50
opt.bisect	51
pe_sampler	53
pgquant	54
plot_condALK	55
plot_condALK_bias	55
plot_lfd_ridgeline	56
plot_lfd_season	57
plot_pe	58
plot_pf	58
plot_sel_a_seasonal	59
plot_sel_age	60

plot_sel_la	62
plotAdvice	63
plotAR	64
plotbioage	66
plotbioyr	67
plotCECAF	67
plotdyn	69
ploteq	69
plotFsim	70
plotGFCM	71
plotMajuro	73
plotspr	74
plotWKREF	75
pop_n_season	76
rc4	77
ref.bisect	77
rffwd	78
rGclass	79
rmultinom_crm	80
rUnif_len	81
rUnif_lfd	82
sam2FLQuant	82
sam2FLStockR	83
sam2stars	83
sample_proc_dev	84
sampling.lfd.oem	85
schaefer.sim	90
seasonalize_catch_n_gear	91
sel_a_seasonal_avg	92
sel_dnormal_len	93
sel_la	94
sel_logistic_len	95
sel_normal_len	96
sops	96
spict2FLQuant	97
spict2FLStockR	98
spict2stars	99
spr2depletion	99
ss2FLStockR	100
ss2ices	101
ss2stars	101
ss3col	102
ss3devs	102
ss3vcv	103
ssmvln	103
stock2ratios	104
stock2spm	105
stockMedians	105

updCatch.n	106
updsr	106
updstars	107
vbbrp	107
wt_a_seasonal	108
Index	110

ABItgt	<i>ABItgt() Computes ABI for target F, e.g. ABImSY (Griffith et al. 2023)</i>
--------	---

Description

ABItgt() Computes ABI for target F, e.g. ABImSY (Griffith et al. 2023)

Usage

ABItgt(stock, ftgt = NULL, thresh = 0.9, ...)

Arguments

- stock object of class FLStock or FLStockR
- ftgt target F at equilibrium, e.g. Fmsy
- thresh quantile ageref treshold, default 0.9

Value

FLQuant

Examples

```
data(ple4)
ABImSY = ABItgt(ple4,ftgt=0.22,thresh=0.9)
plot(ABImSY)+ylim(0,2)+
  geom_hline(yintercept = c(0.8,1),col=c(2,1),linetype=c(2,1))+ylab(expression(ABI[MSY]))
```

 ALK

Convert an inverse age-length key to an age-length key

Description

Combines numbers-at-age with an inverse age-length key and conditions the result on length to obtain age proportions within each length class.

Usage

```
ALK(N_a, iALK)
```

Arguments

N_a	Numeric vector of numbers-at-age for one sampling event.
iALK	Inverse age-length key returned by [iALK()].

Value

An ‘FLPar’ containing age proportions by length class.

Examples

```
## Not run:
alk <- ALK(N_a = c(100, 80, 50, 20), iALK = ialk)

## End(Not run)
```

 alk.sample

Sample annual age-length keys by length class

Description

Generates multinomial age samples within length classes from annual age-length keys. The number sampled in each length class is limited by ‘nbin’ and by the corresponding length-frequency sample.

Usage

```
alk.sample(lfds, alks, nbin = 20, n.sample = 1)
```

Arguments

lfds	An 'FLQuant' containing length-frequency data by year and iteration.
alks	An 'FLPars' object containing annual age-length keys, typically returned by [ALKs()].
nbin	Maximum number of age observations sampled per length class. Defaults to '20'.
n.sample	Optional total length-sample size used to rescale 'lfds'. A value of '1' uses the supplied frequencies without rescaling.

Value

An 'FLPars' object containing sampled annual age-length keys.

Examples

```
## Not run:
sampled_alks <- alk.sample(lfds, alks, nbin = 20)

## End(Not run)
```

ALKs	<i>Construct annual age-length keys</i>
------	---

Description

Combines annual numbers-at-age with an inverse age-length key and returns age proportions within each length class for every year and iteration.

Usage

```
ALKs(object, iALK)
```

Arguments

object	An 'FLQuant' containing numbers-at-age by year and iteration.
iALK	Inverse age-length key returned by [iALK()].

Value

An 'FLPars' object with one age-length key per year.

Examples

```
## Not run:
alks <- ALKs(stock.n(stk), ialk)

## End(Not run)
```

applyALK	<i>Convert length frequencies to numbers-at-age</i>
----------	---

Description

Applies annual age-length keys to length-frequency data to estimate numbers-at-age by year and iteration.

Usage

```
applyALK(lfds, alks)
```

Arguments

lfds	An 'FLQuant' containing numbers-at-length.
alks	An annual 'FLPar' collection returned by [ALKs()], or a single 'FLPar' applied to every year.

Value

An 'FLQuant' containing estimated numbers-at-age.

Examples

```
## Not run:
numbers_at_age <- applyALK(lfds, alks)

## End(Not run)
```

applyAR	<i>applyAR() Function to apply ICES Advice rule for $F < MSY$ Btrigger</i>
---------	--

Description

applyAR() Function to apply ICES Advice rule for $F < MSY$ Btrigger

Usage

```
applyAR(b, btrigger, fmsy, bmin = 0, fmin = 0.001)
```

Arguments

b	biomass
btrigger	if b below btrigger F is reduced
fmsy	Fmsy target
bmin	Option for de facto fishing closure (e.g. bmin = blim)
fmin	F below bmin

Value

F advice

asem2spm	<i>asem2spm()</i>
----------	-------------------

Description

Converts equilibrium reference-point information from an FLBRP object into approximate Pella-Tomlinson surplus production model parameters. The function derives the shape parameter m from the equilibrium ratio B_{msy} / B_0 , and derives the intrinsic productivity parameter r from the harvest rate at MSY .

Usage

```
asem2spm(  
  object,  
  quant = c("vb", "ssb"),  
  fmsy = NULL,  
  rel = FALSE,  
  spcurve = FALSE  
)  
  
asem2spm(  
  object,  
  quant = c("vb", "ssb"),  
  fmsy = NULL,  
  rel = FALSE,  
  spcurve = FALSE  
)
```

Arguments

object	An FLBRP object containing equilibrium reference points and a grid of fishing mortality values.
quant	Character. Biomass metric used to derive the SPM approximation. Options are "vb" for vulnerable biomass or "ssb" for spawning-stock biomass. Default is "vb".

fmsy	Optional fishing mortality at MSY. If NULL, the value is taken from <code>refpts(object)["msy", "harvest"]</code> .
rel	Logical. If TRUE and <code>spcurve = TRUE</code> , return the surplus production curve in relative units, with biomass divided by B_0 and surplus production divided by MSY. Default is FALSE.
spcurve	Logical. If TRUE, return a list containing both the estimated SPM parameters and a data frame of the equilibrium surplus production curve. If FALSE, only the parameter vector is returned. Default is FALSE.

Details

The biomass quantity used for the approximation can be either vulnerable biomass, `quant = "vb"`, or spawning-stock biomass, `quant = "ssb"`. Optionally, the function can also return the equilibrium surplus production curve implied by the supplied FLBRP object.

The function approximates the age-structured equilibrium yield curve with a Pella-Tomlinson production function:

$$P(B) = B \frac{r}{m-1} \left[1 - \left(\frac{B}{B_0} \right)^{m-1} \right]$$

The shape parameter m is solved numerically from the relationship:

$$\frac{B_{MSY}}{B_0} = m^{-1/(m-1)}$$

and r is calculated from:

$$r = H_{MSY} \frac{m-1}{1-1/m}$$

where $H_{msy} = MSY / B_{msy}$. For `quant = "vb"`, B_{msy} and B_0 are based on vulnerable biomass calculated by `vbbpr()`. For `quant = "ssb"`, they are based on `ssb()`.

This approximation is useful for deriving SPM-compatible prior means for r and m , or for computing SPM-style process deviations from age-structured stochastic projections.

Value

prior means for r and m *FLPar*

If `spcurve = FALSE`, an FLPar object with approximate surplus production parameters:

r Intrinsic productivity parameter of the Pella-Tomlinson surplus production model.

m Pella-Tomlinson shape parameter.

B_{msyK} Equilibrium biomass ratio B_{msy} / B_0 .

H_{msy} Harvest rate at MSY, calculated as MSY / B_{msy} .

MSY Maximum sustainable yield.

B_{msy} Biomass at MSY for the selected biomass metric.

B0 Unfished biomass for the selected biomass metric.

Fmsy Fishing mortality at MSY.

If spcurve = TRUE, a list is returned with:

pars The FLPar object described above.

curve A data frame with columns Biomass, SP, and quant. If rel = TRUE, Biomass is scaled by B0 and SP by MSY.

Examples

```
data(ple4)
ple4 <- merge_catch(ple4)
sr <- srrTMB(as.FLSR(ple4,model=bevholt),spr0=mean(spr0y(ple4)))
brp = brp(FLBRP(ple4,sr))
asem2spm(brp)
plot_pf(brp)
plot_pf(brp,rel=TRUE)
## Not run:
data(ple4)

ple4 <- merge_catch(ple4)

sr <- srrTMB(
  as.FLSR(ple4, model = bevholt),
  spr0 = mean(spr0y(ple4))
)

brp <- brp(FLBRP(ple4, sr))

# Approximate SPM parameters using vulnerable biomass
pars_vb <- asem2spm(brp, quant = "vb")
pars_vb

# Approximate SPM parameters using spawning biomass
pars_ssb <- asem2spm(brp, quant = "ssb")
pars_ssb

# Return the surplus production curve in absolute units
sp <- asem2spm(brp, quant = "vb", spcurve = TRUE)
head(sp$curve)

# Return the surplus production curve in relative units
sp_rel <- asem2spm(brp, quant = "vb", spcurve = TRUE, rel = TRUE)
head(sp_rel$curve)

plot_pf(brp)
plot_pf(brp, rel = TRUE)

## End(Not run)
```

bioidx.sim	<i>generates FLIndexBiomass with random observation error from an FLStock</i>
------------	---

Description

generates FLIndexBiomass with random observation error from an FLStock

Usage

```
bioidx.sim(object, sel = catch.sel(object), sigma = 0.2, q = 0.001, rho = 0)
```

Arguments

object	FLStock
sel	FLQuant with selectivity.pattern
sigma	observation error for log(index)
q	catchability coefficient for scaling

Value

FLIndexBiomass

Examples

```
data(ple4)
sel = newselex(catch.sel(ple4),FLPar(S50=1.5,S95=2.1,Smax=4.5,Dcv=1,Dmin=0.1))
ggplot(sel)+geom_line(aes(age,data))+ylab("Selectivity")+xlab("Age")
object = propagate(ple4,10)
sel = newselex(catch.sel(object),FLPar(S50=2.5,S95=3.2,Smax=3.5,Dcv=0.6,Dmin=0.2))
idx = bioidx.sim(object,sel=sel,q=0.0001)
# Checks
ggplot(idx@sel.pattern)+geom_line(aes(age,data))+ylab("Selectivity")+xlab("Age")
ggplot(idx@index)+geom_line(aes(year,data,col=ac(iter)))+theme(legend.position = "none")+ylab("Index")
```

bisect	<i>Bisection search for a forecast target matching a performance statistic</i>
--------	--

Description

Uses a bisection algorithm to find the value of a forecast control quantity that gives a specified target value for a performance statistic. Typical use cases include finding the constant $\bar{f}$ that gives a chosen risk level, such as $P(SSB < B_{lim}) = 0.05$, over a projection period.

Usage

```

bisect(
  stock,
  sr,
  deviances = rec(stock) %>% 1,
  metrics,
  refpts,
  statistic,
  years,
  pyears = years,
  tune,
  prob,
  tol = 0.01,
  maxit = 15,
  verbose = TRUE
)

```

Arguments

<code>stock</code>	An FLStock object used as the starting point for the forecast. The object can be propagated over iterations before calling <code>bisect()</code> .
<code>sr</code>	A stock-recruitment object or model passed to <code>fwd()</code> or <code>ffwd()</code> through the <code>sr</code> argument.
<code>deviances</code>	Recruitment deviances used in the stochastic forecast. Defaults to <code>rec(stock) %>% 1</code> , i.e. deterministic recruitment multipliers of one. For stochastic projections this is usually an FLQuant of lognormal or autocorrelated recruitment deviances.
<code>metrics</code>	A named list of metric functions passed to <code>performance()</code> , for example <code>list(SB = ssb)</code> .
<code>refpts</code>	Reference points passed to <code>performance()</code> , typically an FLPar object such as <code>FLPar(SBlim = 150000)</code> .
<code>statistic</code>	A named list defining the performance statistic to evaluate, passed to <code>performance()</code> . For example, a risk statistic based on $SB / SBlim < 1$.
<code>years</code>	Projection years over which the forecast control is applied.
<code>pyears</code>	Years over which the performance statistic is evaluated. Defaults to <code>years</code> . This is often a subset of the projection period, for example the final 50 years of a long stochastic forecast.
<code>tune</code>	A named numeric vector or list giving the forecast quantity to tune and its lower and upper bounds. For example, <code>list(fbar = c(0.1, 1.0))</code> . If the tuned quantity is named "f" or "fbar", <code>ffwd()</code> is used; otherwise <code>fwd()</code> is used.
<code>prob</code>	Numeric. Target value of the performance statistic. For example, 0.05 for a 5 percent risk threshold.
<code>tol</code>	Numeric. Absolute tolerance used to decide whether the target statistic has been reached. Default is 0.01.
<code>maxit</code>	Integer. Maximum number of bisection iterations. Default is 15.

verbose Logical. If TRUE, print the value of the tuned control, the resulting statistic, and the difference from the target at each bisection step.

Details

The function evaluates the forecast at the lower and upper bounds supplied in `tune`. If the requested probability lies between the two outcomes, the interval is repeatedly bisected until the performance statistic is within `tol` of `prob`, or until `maxit` iterations are reached.

The bisection search requires that the target objective is bracketed by the two values supplied in `tune`. In practice, this means that:

$$[g(x_{min}) - p][g(x_{max}) - p] < 0$$

where $g(x)$ is the performance statistic produced by forecasting with control value x , and p is the target value supplied by `prob`. If both bounds give performance statistics on the same side of the target, the function cannot identify a unique bracketed solution.

The function is most useful for risk-based advice calculations, such as finding the fishing mortality that gives a pre-specified probability of falling below `Blim` over a future evaluation period.

Value

If a solution is found, an `FLStock` object returned by `fwd()` or `ffwd()` for the tuned control value. If the requested target is already achieved at the lower or upper bound, the corresponding forecast object is returned. If the supplied range does not bracket the target statistic, a list with elements `min` and `max` is returned, containing the forecasts at the lower and upper bounds.

References

Burden, R. L. and Faires, J. D. (1985). Numerical Analysis. 3rd edition. PWS Publishers. Section 2.1: The Bisection Algorithm.

Examples

```
## Not run:
data(ple4)

# Extend and propagate stock
stock <- propagate(stf(ple4, end = 2118), 100)

# Define a Beverton-Holt stock-recruitment relationship
srr <- predictModel(
  model = rec ~ a * ssb * exp(-b * ssb),
  params = FLPar(a = 5.20, b = 1.65e-6)
)

# Generate autocorrelated recruitment deviances
devs <- ar1rlnorm(
  rho = 0.4,
  years = 2018:2118,
  iters = 100,
```

```

    meanlog = 0,
    sdlog = 0.5
  )

  # Define an ICES-style risk statistic
  statistic <- list(
    FP05 = list(
      ~ yearMeans((SB / SBlim) < 1),
      name = "P.05",
      desc = "ICES P.05"
    )
  )

  # Find fbar that gives P(SSB < Blim) = 0.05
  fp05fwd <- bisect(
    stock = stock,
    sr = srr,
    deviances = devs,
    metrics = list(SB = ssb),
    refpts = FLPar(SBlim = 150000),
    statistic = statistic,
    years = 2018:2118,
    pyears = 2069:2118,
    tune = list(fbar = c(0.1, 1.0)),
    prob = 0.05
  )

  ## End(Not run)

```

blag

blag()

Description

function to assign $B[y+1]$ to $B[y]$. Warning correlation structure of $B[y+1]$ and $F[y]$ is meaningless

Usage

```
blag(mvn, verbose = TRUE)
```

Arguments

mvn

Value

output list of quant posteriors and mle's

Author(s)

Henning Winker (GFCM)

build_gear

*Build gear selectivity, F-partition and sampling configurations***Description**

Constructs, for each gear of a multigear operating model, the selectivity-at-length, the ALK-integrated selectivity-at-age, the capture-conditioned inverse age-length key, and (optionally) the Fbar-standardised selectivity and the gear's share of total F-at-age.

Usage

```
build_gear(
  gear_cfg,
  lhpar,
  age,
  amin = min(age),
  amax = max(age),
  lmin = 5,
  lmax_mult = 1.2,
  bin = 1,
  cv = 0.1,
  timing = 0,
  fbar_range = NULL,
  scale = TRUE
)
```

Arguments

gear_cfg	Named list of gear configurations. Each gear must contain 'type' ("logistic", "normal" or "dnormal"), 'f_mult', 'ess_age' and 'ess_len'; all remaining entries are passed to [sel_la()] as that type's selectivity parameters ('L50'/'L95'; 'Lpeak'/'sd'; 'Lpeak'/'sd_left'/'sd_right').
lhpar	'FLPar' with at least 'linf', 'k' and 't0'.
age	Numeric vector of ages (the OM's age dimension).
amin, amax	Minimum and maximum ages passed to [sel_la()].
lmin	Lower limit of the length grid. Default '5'.
lmax_mult	Multiplier on 'linf' for the upper (plus-group) length. Default '1.2'.
bin	Length-bin width. Default '1'.
cv	Coefficient of variation of length-at-age. Default '0.1'.
timing	Within-year timing (fraction of a year) at which length-at-age is evaluated; growth uses 't0 - timing'. Use the mid-point of the fishing / sampling season (e.g. '0.5'). Default '0'.

<code>fbar_range</code>	Optional ages defining Fbar, either the full vector ('3:19') or 'c(minfbar, maxfbar)'. When supplied, each gear additionally carries 'sel_a_fbar' and 'f_share' (see Value). Required by [f_age_gear()], [joint_sel()] and the static form of [catch_n_gear()].
<code>scale</code>	Logical; scale 'sel_len' and 'sel_a' to a maximum of one.

Details

With $P(l \mid a, t)$ the biological inverse ALK at 'timing' t :

$$s_{g,a} = \sum_l P(l \mid a, t) s_g(l), \quad P(l \mid a, g) = \frac{P(l \mid a, t) s_g(l)}{s_{g,a}}.$$

F partition. With $\mathcal{A}$ the Fbar ages and $w_g = \text{'f_mult'}$, gear Fbar trajectories in the OM are $\bar{F}_{g,y} = w_g \bar{F}_y$. Gear F-at-age and the share of total F are then

$$F_{g,a,y} = w_g \tilde{s}_{g,a} \bar{F}_y, \quad \phi_{g,a} = \frac{w_g \tilde{s}_{g,a}}{\sum_h w_h \tilde{s}_{h,a}},$$

which does not depend on year or iteration: as long as all gears follow proportional Fbar trajectories, the F-share (and hence the gear split of catch-at-age) is a property of the gear configuration and belongs here. If gear trajectories diverge, compute time-varying shares with [f_share_gear()] instead.

All quantities live on one length grid, [len_grid()] (= [iALK()]’s grid, last bin a plus group); selectivity is evaluated at its bin mid-points with [sel_logistic_len()], [sel_normal_len()] or [sel_dnormal_len()]. 'amin'/'amax' are kept for backward compatibility and are unused.

Value

A named list, one element per gear, each a list with

name Gear name.

type Selectivity type.

sel_len 'FLQuant' selectivity-at-length $s_g(l)$ on the reconciled length grid (lower bin limits as dimnames).

sel_a 'FLQuant' ALK-integrated selectivity-at-age $s_{g,a}$, max-scaled if 'scale = TRUE'.

sel_a_fbar 'FLQuant' $\tilde{s}_{g,a} = s_{g,a} / \bar{s}_{g,\mathcal{A}}$, i.e. selectivity scaled to mean one over the Fbar ages, so that $F_{g,a,y} = \tilde{s}_{g,a} \bar{F}_{g,y}$. 'NULL' without 'fbar_range'.

f_share 'FLQuant' static share of total F-at-age taken by this gear, $\phi_{g,a}$ (see Details). Sums to one over gears at every age with non-zero F. 'NULL' without 'fbar_range'.

len_bins Data frame of lower, upper and mid lengths.

condALK Matrix $P(l \mid a, g)$, ages in rows.

f_mult, ess_age, ess_len As supplied in 'gear_cfg'.

lhpar 'lhpar', used as the default 'params' of [lfd.sim()].

lw Length-weight parameters 'c(a, b)' taken from 'lhpar' ('NULL' if 'lhpar' has no 'a'/'b').

wt_len Named vector of weight at length, $W(l) = a\bar{l}^b$ at bin mid-points (same units as ‘a’/‘b’), used by [lfd_sop()] to raise length samples to catch weight.

grid List of ‘lmin’, ‘lmax_mult’, ‘bin’, ‘cv’, ‘timing’, ‘fbar_range’ used to build the gear. Downstream helpers ([plot_condALK_bias()], [lfd.sim()]) when rebuilding the ALK) should default to these instead of repeating the arguments.

See Also

[joint_sel()], [f_age_gear()], [catch_n_gear()], [lfd.sim()]

Examples

```
## Not run:
gear_cfg <- list(
  Trawl = list(type = "dnormal", Lpeak = 23.4, sd_left = 3.2,
               sd_right = 11.2, f_mult = 0.9, ess_age = 50, ess_len = 150),
  Gillnet = list(type = "logistic", L50 = 29.7, L95 = 34.1,
                 f_mult = 0.1, ess_age = 30, ess_len = 100)
)
om_gears <- build_gear(gear_cfg, lhpar = lhpar, age = 0:20,
                      timing = 0.5, fbar_range = c(3, 19))
gear_par(om_gears, "f_mult")
joint_sel(om_gears)

## End(Not run)
```

ca.sim

Generate catch.n with lognormal annual and multinomial age composition observation error

Description

Generate catch.n with lognormal annual and multinomial age composition observation error

Usage

```
ca.sim(
  object,
  ess = 200,
  u = NULL,
  what = c("catch", "landings", "discards")[1],
  rescale = FALSE
)
```

Arguments

object	'FLQuant' of true numbers/catch-at-age.
ess	Effective sample size for the age-composition sample.
u	Optional. A '[ess, year, iter]' array of pre-generated 'Uniform(0,1)' deviates (e.g. from [rUnif_lfd()]) for CRN-safe sampling via [rmultinom_crn()]. If 'NULL' (default), sampling falls back to 'stats::rmultinom()' against the live RNG, i.e. the original 'ca.sim()' behaviour is unchanged.
what	One of "catch", "landings", "discards" (currently unused inside the function body, kept for interface compatibility with the original 'ca.sim()').
rescale	Logical. If 'TRUE', rescale the sampled age composition back to the abundance scale of 'object' (sampled proportions x true total), rather than returning raw counts on the 'ess' scale.

Value

'FLQuant', same dimensions as 'object', with sampled catch-at-age (or rescaled proportions x true total, if 'rescale=TRUE').

Examples

```
## Not run:
data(ple4)
object <- propagate(catch.n(ple4), 10)

## original behaviour, unchanged
ca1 <- ca.sim(object, ess = 200)

## CRN-safe: same u -> same sample, even if called twice
u <- array(runif(200 * dim(object)[2] * 10),
           dim = c(200, dim(object)[2], 10))
ca2 <- ca.sim(object, ess = 200, u = u)
ca3 <- ca.sim(object, ess = 200, u = u)
stopifnot(identical(ca2, ca3))

## End(Not run)
```

catch_by_gear

Relative catch by gear (fiticc() 'catch_by_gear' input)

Description

Sums gear catches over a window of years and normalises across gears:

$$\pi_g = \frac{\sum_{y \in \mathcal{Y}} C_{g,y}}{\sum_h \sum_{y \in \mathcal{Y}} C_{h,y}}.$$

Supplying the realised catch split over the LFD window (rather than the nominal 'f_mult') mirrors what an assessment would actually be given, and reflects that a gear's catch share differs from its Fbar share whenever gears select different ages.

Usage

```
catch_by_gear(  
  catch_n,  
  wt = NULL,  
  years = NULL,  
  iter = NULL,  
  type = c("weight", "numbers")  
)
```

Arguments

catch_n	‘FLQuants’ of catch numbers-at-age by gear (from [catch_n_gear()]).
wt	‘FLStock’ (its ‘catch.wt’ is used) or an ‘FLQuant’ of weight-at-age. Ignored for ‘type = "numbers"’.
years	Years to sum over (e.g. ‘lfd_years’). Default all.
iter	Optional iteration(s). With a single ‘iter’ a named vector is returned; otherwise an ‘iter x gear’ matrix (one row per fit).
type	“weight” (default) or “numbers”.

Value

Named numeric vector or matrix of catch shares (rows sum to one).

Examples

```
## Not run:  
catch_by_gear(catch_n, om_hist, years = lfd_years, iter = 1)  
cbg <- catch_by_gear(catch_n, om_hist, years = lfd_years) # all iters  
fiticc(lfd_i, stklen_i, sel_fun = sel_fun, catch_by_gear = cbg[i, ])  
  
## End(Not run)
```

catch_gear	<i>Catch by gear and year</i>
------------	-------------------------------

Description

Catch by gear and year

Usage

```
catch_gear(catch_n, wt = NULL, type = c("weight", "numbers"))
```

Arguments

catch_n	‘FLQuants’ of catch numbers-at-age by gear (from [catch_n_gear()]).
wt	‘FLStock’ (its ‘catch.wt’ is used) or an ‘FLQuant’ of weight-at-age. Ignored for ‘type = "numbers"’.
type	“weight” (default) or “numbers”.

Value

‘FLQuants’ of annual catch by gear (age summed).

catch_n_gear	<i>Partial (Baranov) catch-at-age by gear</i>
--------------	---

Description

Splits the catch implied by the stock’s realised total F among gears:

$$C_{g,a,y} = N_{a,y} \frac{\phi_{g,a,y} F_{a,y}}{Z_{a,y}} (1 - e^{-Z_{a,y}}), \quad Z_{a,y} = F_{a,y} + M_{a,y}.$$

Because shares sum to one, $\sum_g C_{g,a,y}$ equals the Baranov total catch exactly. This is the canonical definition used by the OM and by the OEM (‘sampling.lfd.oem()’), which should call this function rather than reproduce it.

Usage

```
catch_n_gear(stock, shares, years = dimnames(stock)$year, check = TRUE)
```

Arguments

stock	‘FLStock’ with realised ‘stock.n’, ‘harvest’ and ‘m’.
shares	Either a list returned by [build_gear()] (static ‘f_share’, recycled over years and iterations) or ‘FLQuants’ of shares from [f_share_gear()].
years	Years to compute. Default all years of ‘stock’.
check	Logical; verify that gear catches sum to the Baranov total (relative tolerance ‘1e-8’). Default ‘TRUE’.

Value

‘FLQuants’ of catch numbers-at-age by gear. The Baranov total is attached as ‘attr(, "total")’.

Examples

```
## Not run:
cn <- catch_n_gear(om_hist, om_gears, years = 1972:2024) # static shares
cn <- catch_n_gear(om_hist, f_share_gear(f_age))          # time-varying

## End(Not run)
```

combine_sel_age	<i>Combine gear-specific selectivity-at-age using relative apical F</i>
-----------------	---

Description

Combines gear-specific selectivity-at-age curves into a single joint selectivity-at-age curve using relative apical partial fishing mortality weights by gear.

Usage

```
combine_sel_age(object, fapic_rel, normalise_fapic = FALSE, scale = TRUE)
```

Arguments

object	A named list of gear-specific selectivity objects, such as 'sels\$siganus_sutor'. Each gear object should contain 'sel_a' or 'sel_age'.
fapic_rel	Numeric vector of relative apical partial F contributions by gear. Preferably named, with names matching gears in 'object', for example 'c(traps = 1, gillnet = 0.3, linefish = 0.1)'. If unnamed, its length must match 'length(object)', and names are assigned from 'names(object)' in their existing order.
normalise_fapic	Logical. Should 'fapic_rel' be normalised to sum to one before combining curves? Default is 'FALSE'. For the final scaled selectivity shape this has no effect when 'scale = TRUE', but it may be useful for interpretation.
scale	Logical. Should the combined curve be scaled to a maximum of one? Default is 'TRUE'.

Details

Gear-specific selectivity curves are assumed to be scaled to a maximum of one. The combined raw fishing-mortality shape is calculated as:

$$\text{'sel[a] = sum_g fapic_rel[g] * sel_g[a]'}$$

If 'scale = TRUE', the combined curve is scaled to a maximum of one:

$$\text{'sel[a] = sel[a] / max(sel)'}$$

The weights are interpreted as relative apical partial F contributions by gear, not relative Fbar contributions. This avoids dependence on an arbitrary Fbar age range and is consistent with constructing fleet-specific fishing mortality as:

$$\text{'F_g[a, y] = Fapic_g[y] * sel_g[a]'}$$

before summing across gears.

Value

An 'FLQuant' containing the combined selectivity-at-age.

Examples

```
## Not run:
## Preferred: named vector
fapic_rel <- c(traps = 1, gillnet = 0.3, linefish = 0.1)

sel_joint <- combine_sel_age(
  object = sels$siganus_sutor,
  fapic_rel = fapic_rel
)

plot_sel_age(sel_joint)

## Convenience: unnamed vector matched to names(object)
sel_joint <- combine_sel_age(
  object = sels$siganus_sutor,
  fapic_rel = c(1, 0.3, 0.1)
)

## End(Not run)
```

computeFbrp	<i>computeFbrp() Computes biological reference points corresponding to the proxy Fbrp</i>
-------------	---

Description

computeFbrp() Computes biological reference points corresponding to the proxy Fbrp

Usage

```
computeFbrp(
  stock,
  sr = "missing",
  proxy = NULL,
  x = NULL,
  blim = 0.1,
  type = c("b0", "btgt", "value"),
  btri = "missing",
  bpa = "missing",
  bthresh = "missing",
  verbose = T,
  fmax = 10,
  ...
)
```

Arguments

stock	object of class FLStock
sr	stock recruitment model of class FLSR
proxy	choice of Fmsy proxies (combinations permitted) • "sprx" spawning potential ratio spr/spr_0 with basis x • "bx" SSB as fraction $xSSB_0$ • "f0.1" 10 • "fe40" Patterns estimator for Fmsy • "msy" maximum surplus production (not defined for segreg) • numeric user value
x	basis in percent for sprx and bx, e.g. 40 for spr40
blim	values < 1 are taken as fraction to B0 and blim > 1 as absolute values unless specified otherwise
type	type of blim input, values < 1 are • "b0" fraction to B0 • "btgt" fraction to Btarget (first occurring in proxy) • "value" absolute value
btri	Btrigger can specified as absolute value
bpa	Bpa can specified as absolute value
bthresh	Bthresh (GFCM) interchangeable use with Bpa
verbose	
fmax	maximum Flim = $\max(\text{Flim}, f_{\max} * \text{Fbrp})$

Value

brp object of class FLBRP with computed Fbrp reference points

Examples

```
data(ple4)
srr = srrTMB(as.FLSR(ple4,model=rickerSV),spr0=spr0y(ple4))
brp = computeFbrp(stock=ple4,sr=srr,proxy=c("sprx","f0.1"),blim=0.1,type="b0")
ploteq(brp,obs=TRUE,refpts="msy")
```

computeFbrps	<i>computeFbrps()</i> Computes biological reference points corresponding to the proxy Fbrp
--------------	--

Description

computeFbrps() Computes biological reference points corresponding to the proxy Fbrp

Usage

```
computeFbrps(
  stock,
  sr = "missing",
  proxy = c("sprx", "bx", "all"),
  fmsy = FALSE,
  f0.1 = TRUE,
  fmax = 5,
  verbose = T,
  ...
)
```

Arguments

stock	object of class FLStock
sr	stock recruitment model of class FLSR
fmsy	if TRUE, Fmsy is computed (not suggest for segreg or geomean sr)
f0.1	if TRUE, F0.1 is computed
fmax	maximum Flim = minfmax*Fbrp)
verbose	
proxies	choice of Fmsy proxies • "all" both sprx and bx • "sprx" spawning potential ratio spr/spr0 with basis x • "bx" SSB as fraction xSSB0

Value

brp object of class FLBRP with computed Fbrp reference points

condition_alk	<i>Fold length selectivity into an inverse ALK</i>
---------------	--

Description

Reweights an inverse age-length key by selectivity-at-length and normalises each age row to obtain the conditional distribution of length among fish caught by a gear.

Usage

```
condition_alk(ialk_mat, sel_len_vec)
```

Arguments

ialk_mat	Numeric matrix with ages in rows and length classes in columns. Each age row should sum to one.
sel_len_vec	Named numeric vector of selectivity-at-length. Values are reordered to match 'colnames(ialk_mat)'.

Value

A numeric matrix with the same dimensions as 'ialk_mat'. Rows sum to one, except ages with zero selected probability, which are returned as zero rows.

f_age_gear	<i>Gear-specific fishing mortality-at-age from gear Fbar trajectories</i>
------------	---

Description

$$F_{g,a,y} = \tilde{s}_{g,a} \bar{F}_{g,y}$$

with $\tilde{s}_{g,a}$ the Fbar-standardised selectivity stored by [build_gear()]. By construction 'fbar()' of each gear's F-at-age equals the input trajectory.

Usage

```
f_age_gear(stock, om_gears, fbar_gear)
```

Arguments

stock	'FLStock' providing the age / year / iter template (typically the propagated OM before 'fwd()').
om_gears	Named list returned by [build_gear()] with 'fbar_range'.
fbar_gear	'FLQuants' of gear Fbar trajectories (year x iter), named as 'om_gears'.

Value

‘FLQuants’ of F-at-age by gear over the years of ‘fbar_gear’.

Examples

```
## Not run:
fbar_gear <- FLQuants(lapply(gear_par(om_gears, "f_mult"),
                             function(w) f_hist_base * w))
f_age <- f_age_gear(stki, om_gears, fbar_gear)
harvest(stki)[, dimnames(fbar_gear[[1]])$year] <- Reduce(`+`, f_age)

## End(Not run)
```

f_share_gear

Time-varying share of total F-at-age by gear

Description

$$\phi_{g,a,y} = F_{g,a,y} / \sum_h F_{h,a,y}$$

General form of ‘om_gears[[g]]\$f_share’, needed when gear Fbar trajectories are not proportional (e.g. an MP that manages one gear only).

Usage

```
f_share_gear(f_age)
```

Arguments

f_age ‘FLQuants’ of gear F-at-age, e.g. from [f_age_gear()].

Value

‘FLQuants’ of shares (0 where total F is 0).

Fbrp	<i>Fbrp()</i> Extract Fbrp based reference points from output of computeFbrp
------	--

Description

Fbrp() Extract Fbrp based reference points from output of computeFbrp

Usage

Fbrp(brp)

Arguments

brp	input of class FLBRP from ComputeFbrp
-----	---------------------------------------

Value

FLPar object with computed Fbrp reference points

Fe40	<i>Fe40()</i> Patterson estimator for Fmsy
------	--

Description

Fe40() Patterson estimator for Fmsy

Usage

Fe40(stock, nyears = 3)

Arguments

stock	input of class FLStock
nyears	number of years to average

Value

value

flr2stars	<i>flr2stars()</i>
-----------	--------------------

Description

flr2stars()

Usage

flr2stars(object, uncertainty = NULL, quantiles = c(0.05, 0.95))

Arguments

object	of class FLStockR (MLE)
uncertainty	of class FLStock with iters
quantities	default is 90CIs as c(0.05,0.95)

Value

STARS list with \$timeseris and \$refpts

Fmmy	<i>Fmmy()</i> Uses opt.bisect to derive the F at Maximum Median Yield from stochastic simulations
------	---

Description

Fmmy() Uses opt.bisect to derive the F at Maximum Median Yield from stochastic simulations

Usage

```
Fmmy(  
  brp,  
  sigmaR = 0.5,  
  rho = 0,  
  nyears = 100,  
  iters = 250,  
  yrs.eval = NULL,  
  range = "missing",  
  tol = 0.001,  
  maxit = 15,  
  verbose = TRUE  
)
```

Arguments

brp	output object from computeFbrp() of class FLBRP
sigmaR	lognormal recruitment standard deviation
rho	AR1 recruitment autocorrelation coefficient
nyears	number of simulation years
iters	number simulation iterations
yrs.eval	last years to be used evaluation period, default nyears/2
range	range of Fbar value to be evaluated
tol	tolerance
maxit	number of steps
verbose	cat comments

Value

list of FLPar, FLStock and FLBRP objects

Examples

```
data(ple4)
bh = srrTMB(as.FLSR(ple4,model=bevholtsV),spr0=spr0y(ple4))
brp = computeFbrp(ple4,bh,proxy=c("bx","msy"),x=35,blim=0.1)
fmyy = Fmyy(brp,sigmaR=0.7,rho=0.3)
getF(fmyy) # FMY value
plotFsim(fmyy)
brpfmyy = computeFbrp(ple4,bh,proxy=getF(fmyy),blim=0.1)
fsim = Fsim(brpfmyy,sigmaR=0.7,rho=0.3)
plotFsim(fsim)
```

Fp05	<i>Calculates the Fbar value giving a maximum probability of ssb being below Blim of 5 percent</i>
------	--

Description

Calculates the Fbar value giving a maximum probability of ssb being below Blim of 5 percent

Usage

```
Fp05(
  object,
  iters = "missing",
  range = "missing",
  tol = 0.001,
  maxit = 20,
  verbose = TRUE
)
```

Arguments

object	output from Fsim()
iters	Number of iterations, cannot exceed input object
range	range of Fbar value to be evaluated
verbose	Should progress be shown, TRUE.

Value

list

Examples

```
data(ple4)
bh = srrTMB(as.FLSR(ple4,model=bevholtSV),spr0=spr0y(ple4))
brp = computeFbrp(ple4,bh,proxy="bx",x=35,blim=0.2) # set Blim higher
fsim = Fsim(brp,sigmaR=0.7,rho=0.3,iters=500)
plotFsim(fsim)
fp.05 = Fp05(fsim)
plotFsim(fp.05,panels=c(2,4)) # black line is Fp0.05
getF(fp.05)
```

Fsim	<i>Fsim()</i> Simulates stochastic stock dynamics under under constant Fbrp
------	---

Description

Fsim() Simulates stochastic stock dynamics under under constant Fbrp

Usage

```
Fsim(
  brp,
  Ftgt = NULL,
  sigmaR = 0.5,
  rho = 0,
  nyears = 100,
  iters = 250,
  yrs.eval = NULL,
  verbose = TRUE
)
```

Arguments

brp	output object from computeFbrp() of class FLBRP
sigmaR	lognormal recruitment standard deviation
rho	AR1 recruitment autocorrelation coefficient
nyears	number of simulation years
iters	number simulation iterations
yrs.eval	last years to be used evaluation period, default nyears/2
verbose	cat comments

Value

list of FLPar, FLStock and FLBRP objects

Examples

```
data(ple4)
hs = srrTMB(as.FLSR(ple4,model=segreg),spr0=spr0y(ple4),lplim=0.05,uplim=0.25)
blim = params(hs)[[2]]
brp = computeFbrp(ple4,hs,proxy=c("sprx","f0.1","msy"),x=40,blim=blim)
ploteq(brp)
fsim = Fsim(brp,sigmaR=0.7,rho=0.3)
plotFsim(fsim)
plotFsim(fsim,panels=2)
```

Fsim_grid

*Stochastic forward projections over an F grid***Description**

Runs stochastic FLR forward projections with ffwd() over a grid of fishing mortality values. By default, the grid is taken from fbar(brp), with the first value replaced by Fmsy. For each F value, a corresponding FLBRP, FLStock, and FLSR object is constructed, propagated over iterations, and projected forward using recruitment deviances generated by rlnormar1().

Usage

```
Fsim_grid(
  brp,
  Fgrid = NULL,
  sigmaR = 0.5,
  rho = 0,
  nyears = 100,
  iters = 10,
  yrs.eval = NULL,
  parallel = FALSE,
  workers = NULL,
```

```

    progress = TRUE,
    verbose = TRUE
  )

```

Arguments

<code>brp</code>	An FLBRP object containing biological reference points, stock-recruitment information, and an <code>fbar</code> grid.
<code>Fgrid</code>	Optional numeric vector of fishing mortality values. If NULL, values are extracted from <code>fbar(brp)</code> .
<code>sigmaR</code>	Numeric. Lognormal recruitment standard deviation.
<code>rho</code>	Numeric. AR(1) autocorrelation coefficient for recruitment deviances.
<code>nyears</code>	Integer. Number of projection years.
<code>iters</code>	Integer. Number of stochastic iterations.
<code>yrs.eval</code>	Optional integer. Number of final years intended for evaluation. If NULL, defaults to the second half of the projection.
<code>parallel</code>	Logical. If TRUE, run projections in parallel using doFuture .
<code>workers</code>	Optional integer. Number of parallel workers.
<code>verbose</code>	Logical. If TRUE, print progress messages.
<code>keep.stock</code>	Logical. Currently unused; retained for compatibility.

Details

The function is intended for exploring the emergent biomass process error implied by an age-structured operating model under different constant fishing mortality levels. The output can be passed to `pe_sampler()` to obtain a data frame of depletion and process-error estimates.

The function constructs one FLBRP object per `F` value by replacing `fbar(brp)` with a constant `F`. These objects are then converted to FLStock and FLSR objects before stochastic forward projection.

Recruitment deviances are generated once using `rlnormar1()` and then passed to each `ffwd()` call. This allows the `F` scenarios to be compared under the same recruitment-deviation structure.

Value

An FLStocks object containing one projected FLStock per `F` value in the grid. The stocks are named using the corresponding `F` values.

Examples

```

## Not run:
data(ple4)
ple4 <- merge_catch(ple4)
bh <- srrTMB(
  as.FLSR(ple4, model = bevholt),
  spr0 = spr0y(ple4),
  r0 = NULL
)

```

```
brp <- computeFbrp(ple4, bh, proxy = c("msy"))

sim <- Fsim_grid(
  brp,
  sigmaR = 0.5,
  rho = 0.3,
  nyears = 100,
  iters = 20,
  parallel = TRUE,
  workers= 12
)

class(sim)
names(sim)

## End(Not run)
```

fudc

generates an up-down-constant F-pattern

Description

generates an up-down-constant F-pattern

Usage

```
fudc(
  object,
  fref = 0.2,
  fhi = 2.5,
  flo = 0.8,
  sigmaF = 0.2,
  breaks = c(0.5, 0.75)
)
```

Arguments

object	An <i>*FLStock*</i>
fref	reference denominator for fbar
fhi	factor for high F as $fhi = fbar/fref$
flo	factor for low F as $flo = fbar/fref$
sigmaF	variation on fbar
breaks	relative location of directional change

Value

FLQuant

Examples

```

data(ple4)
sr <- srrTMB(as.FLSR(ple4,model=bevholtsV),spr0=mean(spr0y(ple4)))
brp = computeFbrp(ple4,sr,proxy="msy")
fmsy = Fbrp(brp)["Fmsy"]
stki = propagate(ple4,100)
fy = fudc(ple4,fhi=2,flo=0.9,fref=fmsy,sigmaF=0)
fyi = fudc(stki,fhi=2,flo=0.9,fref=fmsy,sigmaF=0.2)
plot(fy,fyi)+ylab("F")
#Forecasting
om <- FLStockR(ffwd(stki,sr,fbar=fyi))
om@refpts = Fbrp(brp)
plotAdvice(window(om,start=1960))

```

fwd2ices

*Function to summarise forecast results***Description**

Function to summarise forecast results

Function to summarise ICES fishing opportunities

Usage

```
fwd2ices(stock, uncertainty, eval.yrs = NULL, dB = NULL, refyr = NULL)
```

```
fwd2ices(stock, uncertainty, eval.yrs = NULL, dB = NULL, refyr = NULL)
```

Arguments

uncertainty	*FLStocks* with list of *FLStockR* objects and iters
eval.yrs	evaluation years of forecast
dB	computes change in percentage biomass to refyr
refyr	change in percentage biomass to refyr
object	*FLStocks* with list of *FLStockR* objects
rel	if TRUE ratios B/Btgt and F/Ftgt are shown
stk	*FLStocks* with list of *FLStockR* objects

Value

data.frame

data.frame

fwd2stars

Function to summarise forecast results

Description

Function to summarise forecast results

Usage

```
fwd2stars(object, eval.yrs = NULL, rel = NULL, dB = NULL, refyr = NULL)
```

Arguments

object	*FLStocks* with list of *FLStockR* objects
eval.yrs	evaluation years of forecast
rel	if TRUE ratios B/Btgt and F/Ftgt are shown
dB	computes change in percentage biomass to refyr

Value

data.frame

fwd2type1

Function to summarise Type 1 GFCM fishing opportunities

Description

Function to summarise Type 1 GFCM fishing opportunities

Usage

```
fwd2type1(
  stock,
  uncertainty = NULL,
  eval.yrs = NULL,
  dB = NULL,
  refyr = NULL,
  rel = NULL,
  osa = FALSE,
  maxdec = 2
)
```

Arguments

uncertainty	*FLStocks* with list of *FLStockR* objects and iters
eval.yrs	evaluation years of forecast
refyr	change in percentage biomass to refyr
rel	values of biomass and F are expressed relative to the target
osa	one-step-ahead for the biomass evaluation year
maxdec	maximum decimal
stk	*FLStocks* with list of *FLStockR* objects

Value

data.frame

fwdF4B

fwdF4B () Bisection Function to search F for a given biomass

Description

fwdF4B () Bisection Function to search F for a given biomass

Usage

```
fwdF4B(
  stock,
  sr,
  btgt,
  nfy = 3,
  niy = 1,
  ival = niy,
  imet = "TAC",
  ftune = c(0, 2),
  tol = 0.001,
  verbose = TRUE,
  ssbQ = 1,
  recQ = 1
)
```

Arguments

stock	FLStock object
sr	stock recruitment relationship
btgt	target biomass of the search
nfy	number of forecast (default 3)
niy	number of intermediate years (default 1)

ival	intermediate year value
imet	intermediate year metric ("TAC","F")
ftune	tuning limits for F search c(0.1,2)
tol	precision tolerance
verbose	

Value

FLStock

gear_ialk	<i>Biological inverse ALK on a gear's own length grid</i>
-----------	---

Description

Builds $P(l \mid a, t)$ with exactly the grid, CV and timing stored in ‘gear\$grid’, so its columns always match ‘gear\$sel_len’ and ‘gear\$condALK’. The upper limit is passed to [iALK()] explicitly via ‘max’, so ‘iALK()’s own ‘lmax’ default can no longer diverge.

Usage

```
gear_ialk(  
  gear,  
  age = as.numeric(rownames(gear$condALK)),  
  timing = gear$grid$timing %||% 0,  
  params = gear$lhpar,  
  model = vonbert,  
  reflen = NULL  
)
```

Arguments

gear	A gear object from [build_gear()] (needs ‘lhpar’, ‘grid’).
age	Ages. Default: rows of ‘gear\$condALK’.
timing	Within-year time; default ‘gear\$grid\$timing’.
params	‘FLPar’ with ‘linf’, ‘k’, ‘t0’; default ‘gear\$lhpar’.
model, reflen	Passed to [iALK()].

Value

Age x length matrix (rows sum to one).

gear_par	<i>Extract a scalar gear entry across all gears</i>
----------	---

Description

Extract a scalar gear entry across all gears

Usage

```
gear_par(om_gears, what)
```

Arguments

om_gears	Named list returned by [build_gear()].
what	Name of a scalar entry, e.g. "f_mult", "ess_age", "ess_len", "type".

Value

Named vector (numeric, or character for "type").

Examples

```
## Not run: gear_par(om_gears, "f_mult")
```

getF	<i>getF()</i>
------	---------------

Description

Helper functio to extract F from various FLRef output

Usage

```
getF(x)
```

Arguments

x	output object from computeFbrp() of class FLBRP
---	---

huecol	<i>huecol</i>
--------	---------------

Description

huecol

Usage

huecol(n, alpha = 1)

Arguments

n	number of colors
alpha	translucency

iALK	<i>Construct an inverse age-length key</i>
------	--

Description

Constructs the probability distribution of length conditional on age from a growth model and a coefficient of variation in length-at-age.

Usage

```
iALK(  
  params,  
  model = vonbert,  
  age,  
  cv = 0.1,  
  lmin = 5,  
  lmax = 1.2,  
  bin = 1,  
  max = ceiling(linf * lmax),  
  reflen = NULL  
)
```

Arguments

params	Growth parameters coercible to an 'FLPar'. Parameters required by 'model' must be named; the default von Bertalanffy model uses 'linf', 'k', and 't0'.
model	Growth function or model object used to predict mean length-at-age. Defaults to 'vonbert'.
age	Numeric vector of ages.

cv	Numeric coefficient of variation in length-at-age. Defaults to '0.1'.
lmin	Numeric lower limit of the length grid. Defaults to '5'.
lmax	Numeric multiplier applied to 'linf' when calculating the upper length limit. Defaults to '1.2'.
bin	Numeric length-bin width. Defaults to '1'.
max	Numeric upper limit of the length grid. Defaults to 'ceiling(linf * lmax)'.
reflen	Optional reference length used to define a constant standard deviation as 'cv * reflen'. When 'NULL', the standard deviation is 'cv' times predicted length-at-age.

Value

An 'FLPar' containing an age-by-length inverse age-length key. Rows represent ages, columns represent length classes, and each age row sums to one.

Examples

```
## Not run:
ialk <- iALK(
  params = FLPar(linf = 45, k = 0.4, t0 = -0.3),
  age = 0:5,
  lmin = 5,
  lmax = 1.2,
  bin = 1
)

## End(Not run)
```

idx.sim	<i>generates FLIndex with lognormal annual and multinomial age composition observation error</i>
---------	--

Description

generates FLIndex with lognormal annual and multinomial age composition observation error

Usage

```
idx.sim(
  object,
  sel = catch.sel(object),
  ages = NULL,
  years = NULL,
  ess = 200,
  sigma = 0.2,
  q = 0.01
)
```

Arguments

object	FLStock
sel	FLQuant with selectivity.pattern
ages	define age range
years	define year range
ess	effective sample size for age composition sample
sigma	annual observation error for log(q)
q	catchability coefficient for scaling

Value

FLIndex

Examples

```
data(ple4)
sel = newselex(catch.sel(ple4), FLPar(S50=1.5, S95=2.1, Smax=4.5, Dcv=1, Dmin=0.1))
ggplot(sel)+geom_line(aes(age, data))+ylab("Selectivity")+xlab("Age")
object = propagate(ple4, 10)
idx = idx.sim(object, sel=sel, ess=200, sigma=0.2, q=0.01, years=1994:2017)
# Checks
ggplot(idx@sel.pattern)+geom_line(aes(age, data))+ylab("Selectivity")+xlab("Age")
ggplot(idx@index)+geom_line(aes(year, data, col=ac(iter)))+facet_wrap(~age, scales="free_y")+
theme(legend.position = "none")+ylab("Index")
```

jabba2FLStockR

jabba2FLStockR()

Description

jabba2FLStockR()
jabba2FLStockR()

Usage

```
jabba2FLStockR(jabba, blim = 0.3, bthr = 0.5, thin = 10, rel = FALSE)

jabba2FLStockR(jabba, blim = 0.3, bthr = 0.5, thin = 10, rel = FALSE)
```

Arguments

jabba	fit from JABBA fit_jabba() or jabba\$kbtrj
blim	biomass limit reference point as fraction of Bmsy
thin	thinnig rate of retained iters
rel	if TRUE ratios BBmsy and FFmsy are stored
bpa	biomass precautionary reference point as fraction of Bmsy

Value

FLStockR with refpts
FLStockR with refpts

jabba2stars	<i>jabba2stars()</i>
-------------	----------------------

Description

jabba2stars()

Usage

jabba2stars(jabba, quantiles = c(0.05, 0.95), blim = 0.3, bthr = 0.5)

Arguments

jabba	fit from JABBA fit_jabba() or jabba\$kbtrj
quantiles	default is 90CIs as c(0.05,0.95)
blim	biomass limit point as fraction of Bmsy, default 0.3Bmsy (ICES)
bthr	biomass precautionary point as fraction of Bmsy, default 0.5Bmsy (ICES)

Value

STARS list with \$timeseris and \$refpts

joint_sel	<i>Joint (fleet-aggregated) selectivity-at-age</i>
-----------	--

Description

Combines gear selectivities on the Fbar scale, weighted by each gear’s share of Fbar (‘f_mult’):

$$S_a = \frac{\sum_g w_g \tilde{s}_{g,a}}{\max_a \sum_g w_g \tilde{s}_{g,a}}.$$

This is the selectivity of total F implied by [f_age_gear()], and is what the aggregate ‘FLBRP’/‘FLStock’ must carry for the equilibrium reference points to be consistent with the multigear dynamics.

Usage

joint_sel(om_gears, scale = TRUE)

Arguments

om_gears Named list returned by [build_gear()] with 'fbar_range'.

scale Logical; scale to a maximum of one. Default 'TRUE'.

Details

Note that [combine_sel_age()] weights *max-scaled* curves by relative *apical* F; the two agree only if the weights are converted between bases. Use 'joint_sel()' whenever 'f_mult' is defined as a share of Fbar.

Value

'FLQuant' of joint selectivity-at-age.

len.sim	<i>Simulate length-frequency samples from numbers-at-age</i>
---------	--

Description

Projects numbers-at-age through an inverse age-length key at one or more within-year sampling times and draws a multinomial length sample. When a gear is supplied, the inverse key is conditioned on its selectivity-at-length.

Usage

```
len.sim(
  N_a,
  params,
  model = vonbert,
  ess = 250,
  timing = seq(0, 11/12, 1/12),
  unit = "cm",
  scale = TRUE,
  reflen = NULL,
  bin = gear$grid$bin %||% 1,
  cv = gear$grid$cv %||% 0.1,
  lmin = gear$grid$lmin %||% 5,
  lmax = gear$grid$lmax_mult %||% 1.2,
  gear = NULL,
  u = NULL
)
```

Arguments

<code>N_a</code>	An 'FLQuant' containing numbers-at-age or catch-at-age.
<code>params</code>	Growth parameters coercible to an 'FLPar'; the default growth model requires 'linf', 'k', and 't0'.
<code>model</code>	Growth function or model object. Defaults to 'vonbert'.
<code>ess</code>	Integer effective sample size. It is divided equally, after rounding, among the values in 'timing'. Defaults to '250'.
<code>timing</code>	Numeric vector of sampling times expressed as fractions of a year. Growth at time 't' is evaluated using 't0 - t'. Defaults to monthly sampling times from '0' to '11/12'.
<code>unit</code>	Character label for the length unit. Defaults to "cm".
<code>scale</code>	Logical; if 'TRUE', rescale each year and iteration to the corresponding total in 'N_a'.
<code>reflen, bin, cv, lmin, lmax</code>	Arguments passed to [iALK()] ('lmax' is the multiplier on 'linf'). When 'gear' is supplied, 'bin', 'cv', 'lmin' and 'lmax' default to 'gear\$grid'.
<code>gear</code>	Optional gear object as returned by [build_gear()]. If supplied, 'gear\$sel_len' is incorporated through [condition_alk()].
<code>u</code>	Optional list of common-random-number arrays, one per timing event, as returned by [rUnif_len()]. If 'NULL', multinomial samples are drawn from the current random-number stream.

Details

This is a single-stage length sampler. In contrast, [lfd.sim()] first samples catch-at-age at low effective sample size and then conditions the length draw on that sampled age composition.

Value

An 'FLQuant' containing sampled length frequencies by year and iteration.

Examples

```
## Not run:
lfd_survey <- len.sim(stock.n(om)[, "2024"], params = lhpar, ess = 300,
  timing = 0.5)
u_len <- rUnif_len(trawl, timing = seq(0, 11/12, 1/12),
  years = an(dimnames(N_a)$year), nsim = dims(N_a)$iter,
  seed = 456)
lfd_survey_g <- len.sim(N_a, params = lhpar, ess = trawl$ess_len,
  timing = seq(0, 11/12, 1/12),
  gear = trawl, u = u_len)

## End(Not run)
```

len_grid	<i>Length grid shared by iALK(), build_gear() and all downstream helpers</i>
----------	--

Description

Lower bin limits `'seq(lmin, ceiling(linf * lmax_mult), bin)'`, exactly as `[iALK()]` builds them; the last bin is the plus group $[l_{max}, \infty)$.

Usage

```
len_grid(linf, lmin = 5, lmax_mult = 1.2, bin = 1)
```

Arguments

linf	Asymptotic length.
lmin, lmax_mult, bin	Grid settings.

Details

Background: before this helper, `'sel_la()'` (exclusive upper limit) and `'iALK()'` (plus-group upper limit) built their grids differently, and every helper that rebuilt an `'iALK()'` (`'plot_condALK()'`, `'plot_condALK_bias()'`, `'lfd.sim(timing=)'`, `'sel_a_seasonal_avg()'`, `'wt_a_seasonal()'`, `'plot_sel_a_seasonal()'`) used its own default `'lmax_mult = 1.2'`. With `'build_gear(lmax_mult = 1.3)'` those helpers built a shorter grid than the gear's (e.g. 63 vs 69 bins for `Linf = 55.7`), and `'ialk'` take the grid from `'gear$grid'` (via `[gear_ialk()]` or an explicit `'max'`).

Value

Numeric vector of lower bin limits.

lfd.sim	<i>Simulate a two-stage gear-specific length-frequency sample</i>
---------	---

Description

Generates fishery-dependent length-frequency data from gear-specific catch-at-age using a two-stage sampling design.

Usage

```

lfd.sim(
  object,
  gear,
  ess_age = gear$ess_age,
  ess_len = gear$ess_len,
  u_age = NULL,
  u_len = NULL,
  scale = TRUE,
  timing = NULL,
  params = gear$lhpar,
  model = vonbert,
  reflen = NULL,
  cv = gear$grid$cv %||% 0.1,
  lmin = gear$grid$lmin %||% 5,
  lmax_mult = gear$grid$lmax_mult %||% 1.2,
  bin = gear$grid$bin %||% 1
)

```

Arguments

<code>object</code>	An 'FLQuant' containing true catch-at-age or numbers-at-age for one gear.
<code>gear</code>	Gear object returned by [build_gear()].
<code>ess_age</code>	Integer effective sample size for the catch-at-age draw.
<code>ess_len</code>	Integer effective sample size for the length draw.
<code>u_age</code>	Optional common-random-number array with dimensions '[ess_age, year, iter]' for the age draw.
<code>u_len</code>	Optional common-random-number array for one sampling time, or a list of arrays when 'timing' contains multiple values.
<code>scale</code>	Logical; if 'TRUE', rescale each sampled length distribution to the corresponding total in 'object'.
<code>timing</code>	Optional numeric vector of within-year sampling times. If 'NULL', use the conditional ALK stored in 'gear'. Otherwise, rebuild the conditional ALK at each time using 't0 - timing'.
<code>params</code>	Growth parameters coercible to an 'FLPar'. Required when 'timing' is supplied.
<code>model</code>	Growth function or model object. Defaults to 'vonbert'.
<code>reflen, cv, lmin, lmax_mult, bin</code>	Arguments for rebuilding the conditional ALK when 'timing' is supplied; 'cv', 'lmin', 'lmax_mult' and 'bin' default to 'gear\$grid', so the rebuilt key always matches the gear.

Details

Sampling proceeds as follows:

1. A catch-at-age composition is sampled with effective sample size 'ess_age' using [ca.sim()].

2. The sampled age composition is projected through $P(l \mid a, g)$ and sampled at effective sample size 'ess_len'.

Value

An 'FLQuant' containing sampled length frequencies by year and iteration.

Examples

```
## Not run:
lfd_trawl <- lfd.sim(catch_n_gear$Trawl, trawl)
devs <- rUnif_lfd(om_gears, years = an(dimnames(catch_n_gear$Trawl)$year),
                 nsim = dim(catch_n_gear$Trawl)[6], seed = 456)
lfd_trawl_crn <- lfd.sim(
  catch_n_gear$Trawl,
  trawl,
  u_age = devs$age$Trawl,
  u_len = devs$len$Trawl
)

## End(Not run)
```

lfd.sim.season	<i>Simulate seasonal length-frequency samples</i>
----------------	---

Description

Applies [lfd.sim()] separately to each season of a season-structured catch-at-age object. Seasonal midpoint timing is '(season - 0.5) / n_seasons'.

Usage

```
lfd.sim.season(
  object,
  gear,
  n_seasons = dim(object)[4],
  ess_age = gear$ess_age,
  ess_len = gear$ess_len,
  params = NULL,
  ...
)
```

Arguments

object	An 'FLQuant' containing catch-at-age with a populated season dimension.
gear	Gear object returned by [build_gear()].

n_seasons	Integer number of seasons. Defaults to the size of the season dimension in ‘object’.
ess_age, ess_len	Effective sample sizes. A scalar is divided equally, after rounding, among seasons; a vector of length ‘n_seasons’ specifies season-specific values.
params	Growth parameters coercible to an ‘FLPar’; must contain ‘linf’, ‘k’, and ‘t0’.
...	Additional arguments passed to [lfd.sim()].

Details

A separate stage-1 age sample is drawn for each season. Scalar sample sizes are divided using ‘round(ess / n_seasons)’ in every season.

Value

An ‘FLQuant’ containing sampled length frequencies by season, year, and iteration.

Examples

```
## Not run:
lfd.sim.season(obj, trawl, n_seasons = 4, params = lhparams)
lfd.sim.season(obj, trawl, n_seasons = 4, ess_len = c(0, 200, 200, 100),
               params = lhparams)

## End(Not run)
```

lfd_sop	<i>Raise a length-frequency sample to catch weight (sum of products)</i>
---------	--

Description

Length analogue of ‘FLRef::sops()’ (which raises catch-at-age with ‘catch.wt’): scales each year/unit/iteration of a length sample so that its sum of products with weight-at-length equals the catch weight,

$$\tilde{N}_{l,y,u} = N_{l,y,u} \frac{C_{y,u}}{\sum_{l'} N_{l',y,u} W(l')}.$$

Years/units with zero sample or zero catch return zero.

Usage

```
lfd_sop(lfd, wt_len, catch)
```

Arguments

lfd	‘FLQuant’ length sample (len x year x unit x ... x iter).
wt_len	Named numeric weight-at-length (names = length bins of ‘lfd’), e.g. ‘gear\$wt_len’ from [build_gear()], or a gear object.
catch	‘FLQuant’ catch weight (year x unit x ... x iter) in the same units as ‘wt_len’.

Value

‘FLQuant’ of raised numbers-at-length, with ‘attr(, "raising")’ the raising factor per year/unit/iter.

See Also

[sampling.lfd.oem()] (‘lfd.raising = TRUE’)

merge_catch

Merge catch into landings and remove discards

Description

Converts an FLStock object with total catch numbers into a landings-only stock by assigning catch.n to landings.n and setting discards.n to zero. Landings and discards weights are then recomputed.

Usage

```
merge_catch(stock)
```

Arguments

stock An FLStock object.

Details

This is a convenience function for cases where catch should be treated as landed catch, for example before forward projection or when preparing simplified single-fleet inputs. The function modifies:

- landings.n(stock): set equal to catch.n(stock)
- discards.n(stock): set to zero
- landings(stock): recomputed with computeLandings()
- discards(stock): recomputed with computeDiscards()

Value

An FLStock object where all catch numbers are assigned to landings and discards are set to zero.

Examples

```
## Not run:
stock <- merge_catch(stock)

# Check that discards have been removed
range(discards.n(stock))
discards(stock)

## End(Not run)
```

Mlorenzen	<i>Mlorenzen</i>
-----------	------------------

Description

computes Lorenzen M with scaling option

Usage

Mlorenzen(object, Mref = "missing", Aref = 2)

Arguments

- object weight-at-age of class **FLQuant**
- Mref reference M for scaling
- Aref reference Age for scaling

Value

FLQuant m()

Examples

```
data(ple4)
Ml = Mlorenzen(stock.wt(ple4))
# Scale
Ms = Mlorenzen(stock.wt(ple4),Mref=0.2,Aref=2)
flqs = FLQuants(Lorenzen=Ml,Scaled=Ms)
```

newselex	<i>generates flexible 5-paramater selex curves</i>
----------	--

Description

generates flexible 5-paramater selex curves

Usage

newselex(object, selexpars)

Arguments

object	FLQuant from catch.sel() or sel.pattern()
selexpars	Selectivity Parameters selexpars S50, S95, Smax, Dcv, Dmin • S50: age at 50 • S95: age at 50 • Smax: age at peak of selectivity before descending limb • Dcv: CV demeterming the steepness of the descending half-normal slope • Dmin: determines the minimum retention of oldest fishes

Value

FLquant with selectivity pattern

Examples

```
data(ple4)
sel = newselex(catch.sel(ple4),FLPar(S50=2,S95=3,Smax=4.5,Dcv=0.6,Dmin=0.3))
ggplot(sel)+geom_line(aes(age,data))+ylab("Selectivity")+xlab("Age")
# Simulate
harvest(ple4)[,] = sel
sr <- srrTMB(as.FLSR(ple4,model=bevholtsV),spr0=mean(spr0y(ple4)))
brp = computeFbrp(ple4,sr,proxy="msy")
fbar(brp) = FLQuant(rep(0.01,70))
stk = as(brp,"FLStock")
units(stk) = standardUnits(stk)
its = 100
stk <- FLStockR(propagate(stk, its))
stk@refpts= Fbrp(brp)
b0=an(Fbrp(brp)["B0"])
control = FLPar(Feq=0.15,Frate=0.1,Fsigma=0.15,SB0=b0,minyear=2,maxyear=70,its=its)
run <- rffwd(stk, sr=sr,control=control,deviances=ar1rlnorm(0.3, 1:70, its, 0, 0.6))
plotAdvice(run)
```

Description

The plain bisection algorithm (Burden & Douglas, 1985) is employed here to find the value of a given forecast target quantity (e.g. 'fbar') for which a selected value of a performance statistic is obtained over a chosen period.

Usage

```
opt.bisect(
  stock,
  sr,
  deviances = rec(stock) %>% 1,
  metrics,
  statistic,
  years,
  pyears = years,
  tune,
  tol = 0.001,
  maxit = 15,
  log = TRUE,
  verbose = TRUE
)
```

Arguments

stock	object class FLStock
sr	object class FLSR
metrics	FLQuant of FLStock to be defined
statistic	
years	years to be evaluated
tune	range for input x
tol	tolerance level
maxit	number of optimisation steps
log	if TRUE, optimise on log-scale

Author(s)

Credits to Iago Mosqueira

References

Burden, Richard L.; Faires, J. Douglas (1985), "2.1 The Bisection Algorithm", Numerical Analysis (3rd ed.), PWS Publishers, ISBN 0-87150-857-5

Examples

```
data(ple4)
stock <- propagate(stf(ple4, end=2118), 200)
srr <- predictModel(model=rec ~ ifelse(ssb <= b, a * ssb, a * b), params=FLPar(a=1.29, b=1.35e+06))
# GENERATE SRR deviances
devs <- ar1rlnorm(rho=0.4, 2018:2118, iters=200, meanlog=0, sdlog=0.5)
# DEFINE MMY statistic
statistic <- list(MMY=list(~apply(L,1,median), name="MMY",
  desc="ICES Maximum Median Yield"))
```

```
# CALL bisect over 100 years, Fmmy calculated over last 50.
fmmmy <- opt.bisect(stock, sr=srr, deviances=devs, metrics=list(L=landings),
  statistic=statistic, years=2018:2118,
  pyears=2069:2118, tune=list(fbar=c(0.01, 0.2)))
# fmmmy
mean(fbar(fmmmy)[,ac(2069:2118)])
```

pe_sampler

Summarise biomass process error over an F grid

Description

Applies `sample_proc_dev()` to each projected stock in an `FLStocks` object and returns a compact data frame of *F*, depletion, and process-error estimates.

Usage

```
pe_sampler(stks, brp, biomass = c("vb", "ssb"), eps = 1e-12)
```

Arguments

<code>stks</code>	An <code>FLStocks</code> object, usually returned by <code>Fsim_grid()</code> .
<code>brp</code>	An <code>FLBRP</code> object used to derive equivalent SPM parameters.
<code>biomass</code>	Character. Biomass metric to use. Either "vb" or "ssb".
<code>eps</code>	Numeric. Small positive value used to avoid logarithms of zero or negative predicted biomass.

Details

This is a convenience wrapper around `sample_proc_dev()`. It is useful for producing the main diagnostic relationship:

$$\sigma_{proc} \sim B/B_{MSY}$$

The output can be plotted directly to visualise how the emergent biomass process error changes with stock depletion or fishing pressure.

Value

A data.frame with one row per *F* scenario and columns:

f Median fishing mortality in the projected stock.
sigma Sampled biomass process-error standard deviation.
bbmsy Median biomass relative to *B*_{msy}.
bb0 Median biomass relative to *B*₀.

Examples

```
## Not run:
data(ple4)

bh <- srrTMB(
  as.FLSR(ple4, model = bevholt),
  spr0 = spr0y(ple4),
  r0 = NULL
)

brp <- computeFbrp(ple4, bh, proxy = c("msy"))

sim <- Fsim_grid(
  brp,
  sigmaR = 0.5,
  rho = 0.3,
  nyears = 100,
  iters = 100,
  parallel = TRUE,
  workers = 4
)

ps <- pe_sampler(sim, brp, biomass = "vb")

head(ps)

plot_pe(ps)

## End(Not run)
```

pgquant

sets plus group on FLQuant

Description

sets plus group on FLQuant

Usage

```
pgquant(object, pg)
```

Arguments

object FLQuant
pg

Value

FLQuant

plot_condALK	<i>Heatmap of a gear's conditional inverse ALK</i>
--------------	--

Description

UPDATED: the optional mean-length overlay is built with [gear_ialk()], so it uses the gear's own grid (any 'lmax_mult') and timing. Previously a hard-coded 'lmax_mult = 1.2' gave a non-conformable ' was built with a different 'lmax_mult', and 't = 0' biology was compared to 't = timing' catch.

Usage

```
plot_condALK(gear, params = NULL, low = "#FFFFCC", high = "#7F0000", ...)
```

Arguments

gear	Gear object from [build_gear()].
params	Optional 'FLPar' ('linf', 'k', 't0'). If supplied (or 'TRUE', meaning 'gear\$lhpar'), overlays biological vs. capture-conditioned mean length-at-age.
low, high	Fill colours.
...	Ignored; accepts the former 'cv', 'lmin', 'lmax_mult', 'bin' arguments, now taken from 'gear\$grid'.

Value

A 'ggplot' object.

plot_condALK_bias	<i>Raw vs. gear-conditioned mean length-at-age</i>
-------------------	--

Description

UPDATED: the raw biological line is now evaluated at the same within-year 'timing' (and grid) the gears were built with. Previously it was evaluated at 'timing = 0' while 'condALK' used 't0 - timing', so for 'timing = 0.5' half a year of growth appeared as "selectivity bias".

Usage

```
plot_condALK_bias(
  om_gears,
  params = om_gears[[1]]$lhpar,
  timing = om_gears[[1]]$grid$timing %||% 0,
  cv = om_gears[[1]]$grid$cv %||% 0.1,
  lmin = om_gears[[1]]$grid$lmin %||% 5,
  lmax_mult = om_gears[[1]]$grid$lmax_mult %||% 1.2,
  bin = om_gears[[1]]$grid$bin %||% 1
)
```

Arguments

om_gears	Named list returned by [build_gear()].
params	'FLPar' with 'linf', 'k', 't0'. Default the gear's 'lhpar'.
timing, cv, lmin, lmax_mult, bin	Default to 'om_gears[[1]]\$grid'.

Value

A 'ggplot' object.

plot_lfd_ridgeline	<i>Plot length-frequency distributions as temporal ridgelines</i>
--------------------	---

Description

Displays annual or seasonal length-frequency distributions along a continuous time axis, with one panel per gear. Optional von Bertalanffy cohort trajectories can be overlaid for reference.

Usage

```
plot_lfd_ridgeline(
  panels,
  lhpar = NULL,
  scale = 0.9,
  cohort_pad = 15,
  annual = FALSE
)
```

Arguments

panels	Named list of length-frequency 'FLQuant' objects. List names determine panel order and labels.
lhpar	Optional growth parameters containing 'linf', 'k', and 't0' for cohort reference lines.
scale	Numeric ridge-height multiplier. Defaults to '0.9'.
cohort_pad	Integer number of additional cohorts generated beyond each end of the plotted time range. Defaults to '15'.
annual	Logical; if 'TRUE', sum the season dimension before plotting.

Details

For seasonal data, ridge positions are calculated as 'year + (season - 0.5) / n_seasons'. When 'annual = TRUE', seasonal data are aggregated with 'seasonSums()'.

Value

A ‘ggplot’ object.

Examples

```
## Not run:
plot_lfd_ridgeline(lfds, lhpar = lhpars)
plot_lfd_ridgeline(lfd_season, annual = TRUE, lhpar = lhpars)

## End(Not run)
```

plot_lfd_season	<i>Plot seasonal length-frequency distributions</i>
-----------------	---

Description

Plots seasonal length-frequency distributions for a selected year, with seasons distinguished by colour and gears shown in separate panels.

Usage

```
plot_lfd_season(lfd_season, year, iter = 1)
```

Arguments

lfd_season	Named list of seasonal length-frequency ‘FLQuant’ objects.
year	Character or numeric year to plot.
iter	Integer simulation iteration. Defaults to ‘1’.

Value

A ‘ggplot’ object.

Examples

```
## Not run:
plot_lfd_season(list(Trawl = lfd_season_trawl, Gillnet = lfd_season_gillnet),
  year = "2020")

## End(Not run)
```

plot_pe	<i>Plot biomass process error against B/Bmsy</i>
---------	--

Description

Plots sampled biomass process error against biomass relative to Bmsy. Optionally both axes can be normalised relative to a reference row, typically row 1, which corresponds to the Fmsy scenario in Fsim_grid().

Usage

```
plot_pe(ps, rel = TRUE, ref = 1, line = TRUE, points = TRUE, reverse_x = FALSE)
```

Arguments

ps	data.frame returned by pe_sampler(), with columns sigma and bbmsy.
rel	Logical. If TRUE, both B/Bmsy and sigma are divided by their reference values.
ref	Integer. Reference row used for normalisation. Defaults to 1.
line	Logical. Add line.
points	Logical. Add points.
reverse_x	Logical. Reverse x-axis.

Value

A ggplot object.

plot_pf	<i>plot_pf()</i>
---------	------------------

Description

plots production functions

Usage

```
plot_pf(object, quant = c("vb", "ssb"), fmsy = NULL, rel = FALSE)
```

Arguments

object	An *FLBRP*
quant	choose between vb and ssb or both
fmsy	default if Fmsy
rel	if TRUE ratios are produced for spcurve

Value

ggplot

Examples

```
data(ple4)
ple4 <- merge_catch(ple4)
sr <- srrTMB(as.FLSR(ple4,model=bevholtSV),spr0=mean(spr0y(ple4)))
brp = brp(FLBRP(ple4,sr))
asem2spm(brp)[1:4]
plot_pf(brp)
plot_pf(brp,rel=TRUE)
```

plot_sel_a_seasonal	<i>Compare single-timing and seasonal selectivity-at-age</i>
---------------------	--

Description

Plots the selectivity-at-age stored in a gear object together with the corresponding seasonally averaged curve. Individual seasonal curves can be displayed as reference lines.

Usage

```
plot_sel_a_seasonal(
  gear,
  lhpar,
  age,
  n_seasons,
  show_seasons = TRUE,
  model = vonbert,
  refln = NULL,
  cv = gear$grid$cv %||% 0.1,
  lmin = gear$grid$lmin %||% 5,
  lmax_mult = gear$grid$lmax_mult %||% 1.2,
  bin = gear$grid$bin %||% 1
)
```

Arguments

gear	Gear object returned by [build_gear()].
lhpar	Life-history parameters containing 'linf', 'k', and 't0'.
age	Numeric vector of ages.
n_seasons	Integer number of equal within-year seasons.
show_seasons	Logical; if 'TRUE', add the individual seasonal curves.
model	Growth function or model object. Defaults to 'vonbert'.
refln	Optional reference length passed to [iALK()].

cv, lmin, lmax_mult, bin
Length-at-age CV and length grid; default to 'gear\$grid'. The key is always built on the grid of 'gear\$sel_len'.

Value

A 'ggplot' object.

Examples

```
## Not run:
plot_sel_age_seasonal(
  om_gears$Trawl,
  lhpars,
  age = 0:5,
  n_seasons = 12
)

## End(Not run)
```

plot_sel_age	<i>Plot selectivity-at-age</i>
--------------	--------------------------------

Description

Plots one or more selectivity-at-age curves stored as 'FLQuant' objects.

Usage

```
plot_sel_age(
  object,
  age_by = 1,
  points = TRUE,
  title = NULL,
  line_width = 0.7,
  point_size = 1.5,
  colours = NULL,
  ylab = "Selectivity",
  xlab = "Age"
)
```

Arguments

- object An 'FLQuant' containing selectivity-at-age, or a list of 'FLQuant' objects. If a list is supplied, each element is plotted as a separate curve. List names are used as legend labels.
- age_by Numeric. Spacing of age-axis breaks. Default is '1'.

points	Logical. Should points be added to the curves? Default is 'TRUE'.
title	Optional plot title. Can be a character string or expression.
line_width	Numeric. Line width passed to 'ggplot2::geom_line()'. Default is '0.7'.
point_size	Numeric. Point size passed to 'ggplot2::geom_point()'. Default is '1.5'.
colours	Optional named vector of colours. Names must match curve names. If 'NULL', ggplot2 default colours are used.
ylab	Character. Y-axis label. Default is '"Selectivity"'.
xlab	Character. X-axis label. Default is '"Age"'.

Details

The function accepts either a single 'FLQuant' or a named list of 'FLQuant' objects. It is intended for comparing joint or gear-specific selectivity-at-age curves used in operating models.

The age dimension is extracted from the 'age' dimension of each 'FLQuant'. Plus-group labels such as '"10+"' are converted to numeric values for plotting.

If a single 'FLQuant' is provided, the legend is suppressed. If a list of 'FLQuant' objects is provided, the legend is shown using the list names.

Value

A 'ggplot' object.

Examples

```
## Not run:
## Single curve
plot_sel_age(sel_joint)

## Multiple species
sel_joint <- list(
  siganus = sel_joint_siganus,
  lethrinus = sel_joint_lethrinus,
  lutjanus = sel_joint_lutjanus,
  shark = sel_joint_shark
)

plot_sel_age(sel_joint)

## End(Not run)
```

plot_sel_la	<i>Plot selectivity at length and age</i>
-------------	---

Description

Plots selectivity-at-length and selectivity-at-age side by side for one selectivity object or for a named list of gear-specific selectivity objects.

Usage

```
plot_sel_la(
  object,
  ncol = 2,
  points_age = TRUE,
  title = NULL,
  line_width = 0.7,
  point_size = 1.5,
  len_by = 5,
  age_by = 1,
  colours = NULL
)
```

Arguments

object	A selectivity object returned by 'sel_la()', or a named list of such objects. For a single object, the function looks for 'sel_len' or 'sel_l', and for 'sel_age' or 'sel_a'. For a list, each element is assumed to represent one gear.
ncol	Numeric. Number of columns in the facet layout. Default is '2'.
points_age	Logical. Should points be added to the age-selectivity panel? Default is 'TRUE'.
title	Optional plot title. Can be a character string or expression.
line_width	Numeric. Line width passed to 'ggplot2::geom_line()'. Default is '0.7'.
point_size	Numeric. Point size for the age-selectivity panel. Default is '1.5'.
len_by	Numeric. Spacing of x-axis breaks for the length panel. Default is '5'.
age_by	Numeric. Spacing of x-axis breaks for the age panel. Default is '1'.
colours	Optional named vector of colours for gears. Names must match gear names. If 'NULL', ggplot2 default colours are used.

Details

The function is intended as a diagnostic plot for operating-model selectivity assumptions. It accepts objects returned by 'sel_la()', where selectivity-at-length and selectivity-at-age are stored as 'FLQuant' objects. If a named list is supplied, each element is treated as a gear and overlaid in both panels.

Length selectivity is plotted against length-bin midpoints when these are available through 'object\$len_bins' or the '"len_bins"' attribute of the length 'FLQuant'. Otherwise, midpoints are inferred from the lower length-bin labels and the median bin width.

Value

A 'ggplot' object.

Examples

```
## Not run:
plot_sel_la(sels$lutjanus_bohar, len_by = 10, age_by = 1)

plot_sel_la(
  sels$carcharhinus_melanopterus,
  len_by = 20,
  age_by = 2,
  title = expression(italic("Carcharhinus melanopterus"))
)

## End(Not run)
```

plotAdvice	<i>plotAdvice Plots stochastic stock dynamics against refpts for constant Fsim()</i>
------------	--

Description

plotAdvice Plots stochastic stock dynamics against refpts for constant Fsim()

Usage

```
plotAdvice(
  object,
  rpts = "missing",
  type = NULL,
  yield = c("catch", "landings")[1],
  plotrefs = TRUE,
  probs = c(0.05, 0.2, 0.5, 0.8, 0.95),
  colour = "dodgerblue",
  ncol = NULL,
  osa = FALSE,
  label.size = 2.5,
  ssbQ = 1,
  recQ = 1
)
```

Arguments

type	age-structured "asm" or surplus production "spm" plotting style
yield	option to select "catch" (default) or "landings"

plotrefs	if TRUE reference points are plotted
probs	determine credibility intervals, default 80th, 90th percentiles #' @param ncol number of plot panel columns
colour	color of CIs
osa	if TRUE it shows one-step-ahead for SSB and Rec
label.size	size of refpts labels
ssbQ	spawning quarter for seasonal models
recQ	recruitment quarter seasonal models
stock	FLStock or FLStockR
refpts	as FLPar or Fbrp() if FLStockR is not provided or should be overwritten

Value

ggplot

Examples

```
data(ple4)
srr = srrTMB(as.FLSR(ple4,model=rickerSV),spr0=spr0y(ple4))
brp = computeFbrp(stock=ple4,sr=srr,proxy=c("sprx","f0.1","fe40"),blim=0.1,type="b0")
plotAdvice (ple4,brp)
```

plotAR

plotAR Plots the new proposed ICES advice rule

Description

plotAR Plots the new proposed ICES advice rule

Usage

```
plotAR(
  pars,
  ftgt = 1,
  btrigger = "missing",
  bpa = "missing",
  bthresh = "missing",
  fpa = "missing",
  fthresh = "missing",
  bclose = 0,
  fmin = 0,
  obs = "missing",
  kobe = TRUE,
  alpha = 1,
  xmax = 1.2,
```

```

ymax = 1.5,
ylab = "missing",
xlab = "missing",
rel = FALSE,
expand = TRUE,
labels = TRUE,
label.cex = 3.5,
critical = TRUE
)

```

Arguments

<code>pars</code>	FLPar object or computeFbrp() output • 1: "Fbrp" # "F.." must first • 2: "Btgt" • 3: "Blim" • 4: "B0"
<code>ftgt</code>	factor to adjust Fmsy or its proxy e.g. 0.8Fmsy
<code>btrigger</code>	biomass trigger below which F is linearly reduced, if > 10 value, else factor*Btgt
<code>bpa</code>	precautionary biomass threshold, if > 10 value, else factor*Blim
<code>fpa</code>	option to input Fpa value
<code>bclose</code>	biomass that invokes fishing closure
<code>fmin</code>	minimum allowable (bycatch) fishing mortality under closure
<code>obs</code>	option to show observation with input class 'FLStock'
<code>kobe</code>	add kobe colour-coding
<code>alpha</code>	transparency of shading
<code>xmax</code>	multiplier for upper default xlim
<code>ymax</code>	multiplier for upper default ylim
<code>ylab</code>	option customize ylab
<code>xlab</code>	option customize xlab
<code>rel</code>	option to denote x,y labs as relative B/Btgt and F/Ftgt
<code>expand</code>	option to expand the plot area to border - default TRUE
<code>labels</code>	annotate reference point labels
<code>critical</code>	option to highlight critical zone below blim
<code>labelslabel.cex=3.5</code>	set size of labels

Value

ggplot

Examples

```

data(ple4)
srr = srrTMB(as.FLSR(ple4,model=segreg),spr0=spr0y(ple4))
blim = params(srr)[[2]]
brp = computeFbrp(stock=ple4,sr=srr,proxy="f0.1",blim=blim)
rpt = Fbrp(brp)
plotAR(rpt,btrigger=an(0.8*rpt["Btgt"]))
# Use Bpa as trigger (ICES style)
plotAR(rpt,obs=ple4,bpa=1.4)
# Change kobe to greyscale
plotAR(rpt,obs=ple4,bpa=1.4,kobe=FALSE)
# add fishing closure with minimum unavoidable F and Btrigger
plotAR(rpt,obs=ple4,bpa=1.4,btrigger=0.7,kobe=TRUE,bclose=1,fmin=0.01)
# show a relative
plotAR(rpt,obs=ple4,rel=TRUE,bpa=1.4,btrigger=0.7,kobe=TRUE,bclose=1,fmin=0.02)

```

plotbioage

plotbioage() Plots stock N_a , W_a , M_a and Mat_a by year

Description

plotbioage() Plots stock N_a , W_a , M_a and Mat_a by year

Usage

```

plotbioage(
  stk,
  metrics = c("Weight", "Maturity", "M", "Selectivity"),
  bysex = FALSE,
  ncol = 2
)

```

Arguments

stk	stock object class FLStock
metrics	choose Weight, Maturity, M, Selectivity
bysex	plot by sex default FALSE
ncol	number of columns in multiplot

Value

ggplot

Examples

```

data(ple4)
plotbioage(ple4)

```

plotbioyr	<i>plotbioyr()</i> Plots stock N_a , W_a , M_a and Mat_a across years
-----------	---

Description

plotbioyr() Plots stock N_a , W_a , M_a and Mat_a across years

Usage

```
plotbioyr(stk, ncol = 2)
```

Arguments

stk	stock object class FLStock
ncol	number of columns in multiplot

Value

ggplot

Examples

```
data(ple4)
plotbioyr(ple4)
```

plotCECAF	<i>plotGFCM()</i>
-----------	-------------------

Description

Produces candidate GFCM advice plot

Usage

```
plotCECAF(
  ftgt = 1,
  btgt = 1,
  blim = 0.3,
  btrigger = "missing",
  bthr = 0.8,
  fthr = 1.2,
  bclose = 0.3,
  fmin = 0.2,
  fadv = 0.8,
  obs = "missing",
  kobe = TRUE,
```

```

    alpha = 1,
    xmax = 1.3,
    ymax = 1.5,
    ylab = "missing",
    xlab = "missing",
    rel = TRUE,
    expand = TRUE,
    labels = TRUE,
    critical = kobe,
    text = TRUE
  )

```

Arguments

btgt	Btarget corresponding Ftgt
blim	biomass limit point used as fraction of Btgt, default 0.25-0.3Btgt
btrigger	biomass trigger (can differ from Bthr)
bthr	biomass precautionary point as fraction of Bmsy, default 0.5Btgt
fthr	upper Ftgt threshold, default 1.2 Ftgt
bclose	fishing closure
fmin	minimum F at closure
fadv	advice F from HCR
obs	"missing" option to show observed ssb,fba
kobe	if TRUE option to add KOBE colors
alpha	option for transparency
xmax	multiplier of btgt for ylim
ymax	multiplier of ftgt for xlim
ylab	
xlab	
rel	illustrate relative to btgt and ftgt if TRUE
expand	= TRUE graphic param
labels	if true add labels
critical	show area below Blim as dark red
text	show text
ftg	Ftarget

Value

GFCM advice plot

Examples

```
plotGFCM(fadv=0.95,btrigger=0.5)
```

plotdyn	<i>plotdyn()</i> Plots stock trajectories at age
---------	--

Description

plotdyn() Plots stock trajectories at age

Usage

```
plotdyn(stk, ncol = 2)
```

Arguments

stk	stock object class FLStock
ncol	number of columns in multiplot

Value

ggplot

Examples

```
data(ple4)
plotdyn(ple4)
```

ploteq	<i>ploteq()</i> Modification of method plot('FLBRP') to plot equilibrium output of computeFbrp()
--------	--

Description

ploteq() Modification of method plot('FLBRP') to plot equilibrium output of computeFbrp()

Usage

```
ploteq(
  brps,
  refpts = "missing",
  obs = FALSE,
  rel = FALSE,
  rpf = TRUE,
  dashed = rpf,
  colours = "missing",
  panels = NULL,
  ncol = 2
)
```

Arguments

brps	output object from computeFbrp of class FLBRP
refpts	Reference points, defaults are computed refpts from computeFbrp() • Fbrp • Blim • B0 • Btri
obs	Should observations be plotted? Defaults to 'FALSE'.
rel	option to denote x,y labs as relative B/Btgt and F/Ftgt
rpf	adds refpts in plots
dashed	plots vertical dashed lines to highlight refpts locations
colours	refpts colours, default is designed for computeFbrp() output
panels	plot panel option 1:4
ncol	number of plot panel columns

Value

ggplot

Examples

```
data(ple4)
srr = srrTMB(as.FLSR(ple4,model=rickerSV),spr0=spr0y(ple4))
brp = computeFbrp(stock=ple4,sr=srr,proxy=c("sprx","f0.1","msy"),blim=0.1,type="b0")
ploteq(brp,obs=TRUE)
ploteq(brp,obs=TRUE,refpts="msy",rel=TRUE)
brp.pa = computeFbrp(stock=ple4,sr=srr,proxy=c("msy","sprx","f0.1"),blim=0.1,bpa=Fbrp(brp)["Blim"]*2,type="b0")
ploteq(brp.pa,obs=TRUE,rel=TRUE)
```

plotFsim	<i>plotFsim Plots stochastic stock dynamics against refpts for constant Fsim()</i>
----------	--

Description

plotFsim Plots stochastic stock dynamics against refpts for constant Fsim()

Usage

```
plotFsim(
  object,
  worms = TRUE,
  thinning = 10,
  probs = c(0.05, 0.2, 0.5, 0.8, 0.95),
```

```

    plotrefs = TRUE,
    colour = "missing",
    ncol = "missing",
    label.size = 3,
    yrs.eval = NULL,
    panels = "missing"
  )

```

Arguments

object	output object from Fsim()
worms	option to show individual iterations
thinning	thinning rate of iterations shows, e.g. 10 shows every 10th
probs	determine credibility intervals, default 80th, 90th percentiles
plotrefs	if TRUE reference points are plotted
colour	color of CIs
ncol	number of plot panel columns
label.size	size of reference points
yrs.eval	last years to be used evaluation period, default half nyears

Value

ggplot

plotGFCM	<i>plotGFCM()</i>
----------	-------------------

Description

Produces candidate GFCM advice plot

Usage

```

plotGFCM(
  ftgt = 1,
  btgt = 1,
  blim = 0.25,
  btrigger = "missing",
  bthr = 0.5,
  fthr = 1.2,
  bclose = 0,
  fmin = 0,
  fadv = 0.8,
  obs = "missing",
  kobe = TRUE,

```

```

alpha = 1,
xmax = 1.3,
ymax = 1.5,
ylab = "missing",
xlab = "missing",
rel = TRUE,
expand = TRUE,
labels = TRUE,
critical = kobe,
text = TRUE
)

```

Arguments

btgt	Btarget corresponding Ftgt
blim	biomass limit point used as fraction of Btgt, default 0.25-0.3Btgt
btrigger	biomass trigger (can differ from Bthr)
bthr	biomass precautionary point as fraction of Bmsy, default 0.5Btgt
fthr	upper Ftgt threshold, default 1.2 Ftgt
bclose	fishing closure
fmin	minimum F at closure
fadv	advice F from HCR
obs	"missing" option to show observed ssb,fba
kobe	if TRUE option to add KOBE colors
alpha	option for transparency
xmax	multiplier of btgt for ylim
ymax	multiplier of ftgt for xlim
ylab	
xlab	
rel	illustrate relative to btgt and ftgt if TRUE
expand	= TRUE graphic param
labels	if true add labels
critical	show area below Blim as dark red
text	show text
ftg	Ftarget

Value

GFCM advice plot

Examples

```
plotGFCM(fadv=0.95,btrigger=0.5)
```

plotMajuro

*Plots the new proposed ICES advice rule***Description**

Plots the new proposed ICES advice rule

Usage

```
plotMajuro(
  ftgt = 1,
  fthresh = 1.1,
  btgt = 1,
  blim = 0.1,
  btrigger = 0.8 * btgt,
  bthresh = 0.5 * btgt,
  bclose = 0,
  fmin = 0,
  obs = "missing",
  kobe = TRUE,
  alpha = 1,
  xmax = 1.5,
  ymax = 1.5,
  ylab = "missing",
  xlab = "missing",
  rel = FALSE,
  expand = TRUE,
  labels = TRUE,
  critical = kobe
)
```

Arguments

ftgt	Target $F = \min(F_{brp}, F_{p0.5})$
btgt	Biomass target corresponding to F_{brp}
blim	biomass limit
btrigger	biomass trigger below which F is linearly reduced
bthresh	biomass threshold beyond which biomass is classified sustainable
bclose	biomass that invokes fishing closure
fmin	minimum allowable (bycatch) fishing mortality under closure
obs	option to show observation with input class 'FLStock'
kobe	add kobe colour-coding
alpha	transparency of shading
xmax	multiplier for upper default xlim

ymax	multiplier for upper default ylim
ylab	option customize ylab
xlab	option customize xlab
rel	option to denote x,y labs as relative B/Btgt and F/Ftgt
expand	option to expand the plot area to border - default TRUE
labels	annotate reference point labels
critical	option to highlight critical zone below blim

Value

ggplot

Examples

```
plotMajuro()
```

plotspr	<i>plotspr()</i> Plots current vs unfished spawning biomass per recruit at age
---------	--

Description

plotspr() Plots current vs unfished spawning biomass per recruit at age

Usage

```
plotspr(stk, nyears = 3)
```

Arguments

stk	stock object class FLStock
nyears	number of current last years, default is 3
ncol	number of columns in multiplot

Value

ggplot

Examples

```
data(ple4)
plotbioage(ple4)
```

plotWKREF

*plotWKREF Plots the newploiomass target corresponding to Fbrp***Description**

plotWKREF Plots the newploiomass target corresponding to Fbrp

Usage

```
plotWKREF(
  ftgt = 1,
  btgt = 1,
  blim = 0.2,
  btrigger = 0.9 * btgt,
  bthresh = 0.8 * btgt,
  bclose = 0,
  fmin = 0,
  obs = "missing",
  kobe = TRUE,
  alpha = 1,
  xmax = 1.3,
  ymax = 1.5,
  ylab = "missing",
  xlab = "missing",
  rel = FALSE,
  expand = TRUE,
  labels = TRUE,
  critical = kobe
)
```

Arguments

blim	biomass limit
btrigger	biomass trigger below which F is linearly reduced
bthresh	biomass threshold beyond which biomass is classified sustainable
bclose	ratio biomass/blim that invokes fishing closure relative to blim
fmin	minimum allowable (bycatch) fishing mortality under closure
obs	option to show observation with input class 'FLStock'
kobe	add kobe colour-coding
alpha	transparency of shading
xmax	multiplier for upper default xlim
ymax	multiplier for upper default ylim
ylab	option customize ylab
xlab	option customize xlab

- rel option to denote x,y labs as relative B/Btgt and F/Ftgt
- expand option to expand the plot area to border - default TRUE
- labels annotate reference point labels
- critical option to highlight critical zone below blim

Value

ggplot

Examples

```
plotWKREF()  
# Close fishery at Blim and adjust axis labels to relative  
plotWKREF(blim=0.2,bclose=0.2,rel=TRUE)  
# Close fishery at Blim, but allow fmin (e.g. bycatch)  
plotWKREF(blim=0.2,bclose=0.2,fmin=0.1,rel=TRUE)  
# Change Btrigger above Btgt  
plotWKREF(blim=0.2,bclose=0.2,fmin=0.1,btrigger=0.80,rel=TRUE)  
# Plot stock data  
data(ple4)  
plotWKREF(ftgt=0.25,btgt=8e+05,btrigger = 0.9*8e+05, blim=2e5,bclose=3e5,fmin=0.03,obs=ple4)
```

pop_n_season	<i>Calculate population numbers-at-age by season</i>
--------------	--

Description

Propagates annual beginning-of-year numbers-at-age through equal within-year seasons under constant total instantaneous mortality.

Usage

```
pop_n_season(stock, n_seasons)
```

Arguments

- stock An 'FLStock' or 'FLStockR' containing annual stock numbers, fishing mortality, and natural mortality.
- n_seasons Integer number of equal within-year seasons.

Value

An 'FLQuant' containing numbers-at-age at the start of each season.

Examples

```
## Not run:
seasonal_numbers <- pop_n_season(om_hist, n_seasons = 4)

## End(Not run)
```

rc4	<i>r4sscol</i>
-----	----------------

Description

r4sscol

Usage

```
rc4(n, alpha = 1)
```

Arguments

n	number of colors
alpha	translucency

Value

vector of color codes

ref.bisect	<i>Bisection approach to find target F for B (sex-structured)</i>
------------	---

Description

The plain bisection algorithm (Burden & Douglas, 1985) is employed here to find the value of a given forecast target quantity (e.g. ‘fbar’) for which a selected value of a performance statistic is obtained over a chosen period.

Usage

```
ref.bisect(
  stock,
  sr,
  deviances = rec(stock) %=% 1,
  metrics,
  refpts,
  statistic,
  years,
```

```
pyears = years,
tune,
target = 1,
tol = 0.01,
maxit = 15,
verbose = TRUE
)
```

Arguments

stock	object class FLStock
sr	object class FLSR
metrics	FLQuant of FLStock to be defined
statistic	
years	years to be evaluated
tune	range for input x
target	target with default 1 for ratios
tol	tolerance level
maxit	number of optimisation steps
log	if TRUE, optimise on log-scale

Author(s)

Credits to Iago Mosqueira

References

Burden, Richard L.; Faires, J. Douglas (1985), "2.1 The Bisection Algorithm", Numerical Analysis (3rd ed.), PWS Publishers, ISBN 0-87150-857-5

rffwd	<i>rffwd()</i> Project forward an FLStock with evolutionary Fbar
-------	--

Description

rffwd() Project forward an FLStock with evolutionary Fbar

Usage

```
rffwd(object, sr, fbar = control, control = fbar, deviances = "missing")
```

Arguments

object	An <i>*FLStock*</i>
sr	A stock-recruit relationship, <i>*FLSR*</i> or <i>*predictModel*</i> .
fbar	Yearly target for average fishing mortality, <i>*FLQuant*</i> .
control	Yearly target for average fishing mortality, <i>*FLPar*</i> .
deviances	Deviances for the stock-recruit relationship, <i>*FLQuant*</i> .

Value

The projected **FLStock** object.

Examples

```
data(ple4)
sr <- srrTMB(as.FLSR(ple4,model=bevholtsV),spr0=mean(spr0y(ple4)))
brp = computeFbrp(ple4,sr,proxy="msy")
fbar(brp) = FLQuant(rep(0.01,70))
stk = as(brp,"FLStock")
units(stk) = standardUnits(stk)
its = 100
stk <- FLStockR(propagate(stk, its))
stk@refpts= Fbrp(brp)
b0=an(Fbrp(brp)["B0"])
control = FLPar(Feq=0.15,Frate=0.1,Fsigma=0.15,SB0=b0,minyear=2,maxyear=70,its=its)
run <- rffwd(stk, sr=sr,control=control,deviances=ar1lnorm(0.3, 1:70, its, 0, 0.6))
plotAdvice(run)
```

rGclass	<i>Function to characterize Productivity and refpts based on r and Generation</i>
---------	---

Description

Function to characterize Productivity and refpts based on r and Generation

Usage

```
rGclass(r = NULL, gt = NULL)
```

Arguments

r	value of the intrinsic rate of population increase
gt	generation time G

Value

list with Productivity category and suggest Fbrps

rmultinom_crn	<i>Draw a multinomial sample using pre-generated uniform deviates (CRN-safe)</i>
---------------	--

Description

Draws a multinomial sample of size `length(u)` from probability vector `p` by inverse-transform sampling: each element of `u` is mapped to a category by comparing it against the cumulative probability vector `cumsum(p)`. This is the standard device for common random numbers (CRN) in simulation studies – as long as the same `u` is reused, the sampled category for a given draw only changes when `p` itself changes, never because of an unrelated call to the random number generator elsewhere in the program.

Usage

```
rmultinom_crn(u, p)
```

Arguments

<code>u</code>	Numeric vector of independent ‘Uniform(0,1)’ deviates, one per individual to be classified. Typically pre-generated once for an entire MSE run and reused across scenarios.
<code>p</code>	Numeric vector of category probabilities. Need not sum to 1 (renormalised internally); values that are ‘NA’, non-finite, or negative are treated as 0.

Details

Contrast with `stats::rmultinom(n, size=1, prob=p)`, which draws against R’s live RNG stream and gives no control over which “slice” of that stream is consumed by which computation – two calls with different `p` will in general use different underlying randomness even if seeded identically, once anything else in the program has touched the RNG in between.

Value

Integer vector of length `length(p)`, the count landing in each category. Sums to `length(u)`, or to 0 if `p` sums to a non-positive value (all-zero true proportions, e.g. no catch that year/iter).

Examples

```
set.seed(1)
u <- runif(1000)
p <- c(0.1, 0.3, 0.6)
rmultinom_crn(u, p)                # ~ c(100, 300, 600)

## CRN property: same u, slightly different p -> only the boundary shifts
p2 <- c(0.12, 0.28, 0.60)
rbind(rmultinom_crn(u, p), rmultinom_crn(u, p2))
```

```
## degenerate case: all-zero probability vector
rmultinom_crn(u, c(0, 0, 0))          # c(0, 0, 0), no error
```

rUnif_len	<i>Generate common random numbers for length sampling</i>
-----------	---

Description

Generates one array of uniform deviates for each within-year timing event for use by [len.sim()].

Usage

```
rUnif_len(gear, timing, years, nsim, seed = NULL)
```

Arguments

gear	Gear object returned by [build_gear()]. ‘gear\$ess_len’ defines the total length-sample size.
timing	Numeric vector of within-year sampling times.
years	Vector of years.
nsim	Integer number of simulation iterations.
seed	Optional integer random seed.

Value

A list with one ‘[draw, year, iter]’ array per timing event.

Examples

```
## Not run:
u_len <- rUnif_len(trawl, timing = seq(0, 11/12, 1/12),
                  years = 2025:2054, nsim = 100, seed = 456)
length(u_len)      # one array per timing event
dim(u_len[[1]])    # ess_t x years x nsim

## End(Not run)
```

rUnif_lfd	<i>Generate common random numbers for two-stage length sampling</i>
-----------	---

Description

Generates uniform-deviate arrays for the age and length stages of [lfd.sim()] for every gear.

Usage

```
rUnif_lfd(om_gears, years, nsim, seed = NULL)
```

Arguments

om_gears	Named list of gear objects returned by [build_gear()].
years	Vector of years.
nsim	Integer number of simulation iterations.
seed	Optional integer random seed.

Value

A list with components ‘age’ and ‘len’. Each component is a named list of ‘[draw, year, iter]’ arrays, one per gear.

Examples

```
## Not run:
om_gears <- list(Trawl = trawl, Gillnet = gillnet)
devs <- rUnif_lfd(om_gears, years = 2025:2054, nsim = 100, seed = 456)

## End(Not run)
```

sam2FLQuant	<i>sam2FLQuant()</i>
-------------	----------------------

Description

sam2FLQuant()

Usage

```
sam2FLQuant(
  object,
  metric = c("ssb", "fbar", "catch", "rec")[1],
  what = c("mle")
)
```

Arguments

object fit from FLSAM
forecast TRUE/FALSE

Value

FLQuant

Author(s)

adopted from Laurie Kell (biodyn)

sam2FLStockR	<i>sam2FLStockR()</i>
--------------	-----------------------

Description

sam2FLStockR()

Usage

sam2FLStockR(res, itsCI = 1000)

Arguments

res fit from FLSAM
itsCI number of iterations to depict uncertainty in plots

Value

FLStockR with refpts

sam2stars	<i>sam2stars()</i>
-----------	--------------------

Description

sam2stars()

Usage

sam2stars(sam, refpts = NULL, quantiles = c(0.05, 0.95))

Arguments

sam	fit from FLSAM
refpts	optional FLPar object with reference points

Value

STARS list with \$timeseris and \$refpts

sample_proc_dev	<i>Sample SPM-equivalent biomass process deviations from one projection</i>
-----------------	---

Description

Calculates biomass process deviations from a projected FLStock by comparing the realised biomass transition with the deterministic surplus-production expectation implied by asem2spm().

Usage

```
sample_proc_dev(stk, brp, biomass = c("vb", "ssb"), eps = 1e-12)
```

Arguments

stk	A projected FLStock object, usually one element from the FLStocks object returned by Fsim_grid().
brp	An FLBRP object used to derive the equivalent surplus-production parameters via asem2spm().
biomass	Character. Biomass metric to use. Either "vb" for vulnerable biomass or "ssb" for spawning-stock biomass.
eps	Numeric. Small positive value used to avoid taking the logarithm of zero or negative predicted biomass.

Details

The process deviation is calculated as:

$$\epsilon_t = \log(B_{t+1}) - \log \left[B_t + B_t \frac{r}{m-1} \left(1 - \left(\frac{B_t}{B_0} \right)^{m-1} \right) - C_t \right]$$

where r , m , B_0 , and B_{MSY} are obtained from asem2spm(brp, quant = biomass).

This corresponds to sampling the residual process variation around the SPM-equivalent biomass transition:

$$B_{t+1} = (B_t + P(B_t) - C_t) \exp(\epsilon_t)$$

Value

A list with elements:

output A one-row data.frame with median F, process-error standard deviation, median B/Bmsy, and median B/B0.

deviations An FLQuant of log biomass process deviations.

ptpar The equivalent surplus-production parameters from `asem2spm()`.

sigma Standard deviation of the sampled process deviations.

bbmsy Biomass relative to Bmsy.

bb0 Median biomass relative to B0.

biomass The biomass metric used.

Examples

```
## Not run:
sim <- Fsim_grid(
  brp,
  sigmaR = 0.5,
  rho = 0.3,
  nyears = 100,
  iters = 100
)

one <- sample_proc_dev(
  stk = sim[[1]],
  brp = brp,
  biomass = "vb"
)

one$output
one$sigma

## End(Not run)
```

Description

An ‘FLoem’ method for ‘mse::mp()’ that generates fishery-dependent length-frequency (LFD) samples and a reported catch split by gear (and, optionally, by area), as a structural analogue of ‘mse::sampling.oem()’. The abundance-index block of ‘sampling.oem()’ is replaced by an LFD block: true catch-at-age is partitioned by gear with `[catch_n_gear()]` and passed through the two-stage sampler `[lfd.sim()]`.

An ‘FLoem’ method for ‘mse::mp()’ that generates fishery-dependent length-frequency (LFD) samples and a reported catch split by gear (and, optionally, by area), as a structural analogue of

`'mse::sampling.oem()'`. The abundance-index block of `'sampling.oem()'` is replaced by an LFD block: true catch-at-age is partitioned by gear with `[catch_n_gear()]` and passed through the two-stage sampler `[lfd.sim()]`.

Usage

```
sampling.lfd.oem(
  stk,
  deviances,
  observations,
  args,
  tracking,
  om_gears,
  f_share = NULL,
  lfd.raising = FALSE,
  lfd.scale = c("sample", "catch")
)

sampling.lfd.oem(
  stk,
  deviances,
  observations,
  args,
  tracking,
  om_gears,
  f_share = NULL,
  lfd.raising = FALSE,
  lfd.scale = c("sample", "catch")
)
```

Arguments

<code>stk</code>	'FLStock' passed by <code>'mp()'</code> (the OM stock).
<code>deviances</code>	List with optional elements 'stk' 'FLQuants' of multiplicative deviances on 'FLStock' slots (e.g. <code>'catch.n'</code>), applied exactly as in <code>'sampling.oem()'</code>. 'lfd' Uniform common-random-number streams for the two sampling stages. Single area: the list returned by <code>[rUnif_lfd()]</code>, i.e. <code>'list(age = list(<gear> = array), len = list(<gear> = array))'</code>, each array <code>'[draw, year, iter]'</code>. Multiple areas: a list named by area (unit) of such lists. If <code>'NULL'</code>, draws use the live RNG. 'catch_gear' 'FLQuants' named by gear of multiplicative reporting deviances on catch weight (year x unit x iter). If <code>'NULL'</code>, the catch split is reported without error.
<code>observations</code>	List with elements 'stk' Observed 'FLStock', as in <code>'sampling.oem()'</code>.

	‘lfd’ ‘FLQuants’ (or list) named by gear: len x year x unit x iter, spanning at least ‘y0:fy’, seeded with historical samples up to the first data year. Areas where a gear is absent stay 0. ‘catch_gear’ ‘FLQuants’ (or list) named by gear: catch weight by year x unit x iter, seeded like ‘lfd’.
args	‘mp()’ arguments; uses ‘y0’, ‘dy’, ‘dys’. ‘y0’ must be set in ‘mp()’s own ‘args’ (the first year of the LFD series), not in ‘FLoem(args =)’.
tracking	‘mp()’ tracking object, returned unchanged.
om_gears	Gear objects from [build_gear()] (built with ‘fbar_range’, so each gear carries ‘f_share’). Single area: the named list of gears. Multiple areas: a list named by area (matching ‘dimnames(stk)\$unit’) of named gear lists; a gear may be absent from an area. A plain gear list with a multi-unit stock is applied to every area.
f_share	Optional ‘FLQuants’ named by gear of time-varying shares of total F-at-age (age x year x unit x iter, covering the projection years), e.g. from [f_share_gear()]. Default ‘NULL’ uses each gear’s static ‘f_share’, which is exact whenever gear Fbar trajectories are proportional and needs no extension into the projection period.
lfd.raising	Logical. If ‘TRUE’ (intended for more than one unit), the per-unit samples passed to ‘est’ are raised to each unit’s reported gear catch by sum of products with the gear’s weight-at-length (‘gear\$wt_len’, from ‘a’ and ‘b’ in ‘lhpars’; see [lfd_sop()]) and summed over units, so the pooled length composition is weighted by catch per unit rather than by sample size per unit. The reported catch by gear is summed over units as well. ‘observations\$lfd’ always keeps the raw samples by unit. Default ‘FALSE’ passes the samples by unit unchanged.
lfd.scale	With ‘lfd.raising = TRUE’: “sample” (default) rescales the pooled, catch-weighted LFD to the total number of fish measured across units, so the sample size seen by the likelihood is the real one; “catch” keeps the raised numbers caught (requires ‘wt_len’ and catch in the same units).

Details

A single-area stock (one ‘unit’, typically “unique”) is the default case. Multiple areas are represented by the ‘unit’ dimension of the operating-model stock, each area with its own gear set.

Each call is incremental, as in ‘sampling.oem()’: only years ‘dys’ are simulated and written into ‘observations’, so each CRN deviate is used once and the same ‘FLoem’ gives identical sampling noise to every MP.

For area u , gear g and year $y \in \text{‘dys’}$:

1. gear catch-at-age by the partial Baranov equation, $C_{g,a,y,u} = N_{a,y,u} \phi_{g,a,y,u} F_{a,y,u} (1 - e^{-Z_{a,y,u}}) / Z_{a,y,u}$, using the OM’s realised F and the (possibly noised) stock;
2. two-stage LFD sample with [lfd.sim()]: a multinomial age sample of size ‘ess_age’, projected through the gear’s $P(l \mid a, g)$ and sampled with size ‘ess_len’;
3. reported catch weight $\sum_a C_{g,a,y,u} w_{a,y,u} \varepsilon_{g,y,u}$.

Raising across units ('lfd.raising = TRUE'). For gear g , year y and unit u with sampled counts $n_{l,y,u}$ and reported catch weight $\hat{C}_{g,y,u}$:

$$N_{l,y} = \sum_u n_{l,y,u} \frac{\hat{C}_{g,y,u}}{\sum_{l'} n_{l',y,u} W(l')}, \quad N_{l,y}^{(sample)} = N_{l,y} \frac{\sum_u \sum_l n_{l,y,u}}{\sum_l N_{l,y}}.$$

Without raising, a unit that is heavily sampled relative to its catch would dominate the pooled composition. With one unit, raising followed by 'lfd.scale = "sample"' returns the sample unchanged.

Zero catch-at-age entries are set to 'sqrt(.Machine\$double.eps)' to avoid 0/0 in [lfd.sim()]; a gear with no catch at all in an area-year leaves a zero LFD.

A single-area stock (one 'unit', typically "unique") is the default case. Multiple areas are represented by the 'unit' dimension of the operating-model stock, each area with its own gear set.

Each call is incremental, as in 'sampling.oem()': only years 'dys' are simulated and written into 'observations', so each CRN deviate is used once and the same 'FLoem' gives identical sampling noise to every MP.

For area u , gear g and year $y \in \text{'dys'}$:

1. gear catch-at-age by the partial Baranov equation, $C_{g,a,y,u} = N_{a,y,u} \phi_{g,a,y,u} F_{a,y,u} (1 - e^{-Z_{a,y,u}}) / Z_{a,y,u}$, using the OM's realised F and the (possibly noised) stock;
2. two-stage LFD sample with [lfd.sim()]: a multinomial age sample of size 'ess_age', projected through the gear's $P(l | a, g)$ and sampled with size 'ess_len';
3. reported catch weight $\sum_a C_{g,a,y,u} w_{a,y,u} \varepsilon_{g,y,u}$.

Raising across units ('lfd.raising = TRUE'). For gear g , year y and unit u with sampled counts $n_{l,y,u}$ and reported catch weight $\hat{C}_{g,y,u}$:

$$N_{l,y} = \sum_u n_{l,y,u} \frac{\hat{C}_{g,y,u}}{\sum_{l'} n_{l',y,u} W(l')}, \quad N_{l,y}^{(sample)} = N_{l,y} \frac{\sum_u \sum_l n_{l,y,u}}{\sum_l N_{l,y}}.$$

Without raising, a unit that is heavily sampled relative to its catch would dominate the pooled composition. With one unit, raising followed by 'lfd.scale = "sample"' returns the sample unchanged.

Zero catch-at-age entries are set to 'sqrt(.Machine\$double.eps)' to avoid 0/0 in [lfd.sim()]; a gear with no catch at all in an area-year leaves a zero LFD.

Value

A list with 'stk' (the observed stock, window 'y0:dy', with attributes "lfd" and "catch_gear": 'FLQuants' by gear holding the accumulated series 'y0:dy', by unit or, with 'lfd.raising = TRUE', pooled over units), 'idx' (empty 'FLIndices'), and the updated 'observations' and 'tracking'. The attributes are how the samples reach the 'est' module (e.g. 'flicc.sa()'), because 'mp()' passes only 'stk' and 'idx' to it.

A list with 'stk' (the observed stock, window 'y0:dy', with attributes "lfd" and "catch_gear": 'FLQuants' by gear holding the accumulated series 'y0:dy', by unit or, with 'lfd.raising = TRUE', pooled over units), 'idx' (empty 'FLIndices'), and the updated 'observations' and 'tracking'. The attributes are how the samples reach the 'est' module (e.g. 'flicc.sa()'), because 'mp()' passes only 'stk' and 'idx' to it.

See Also

```
[build_gear()], [catch_n_gear()], [lfd.sim()], [rUnif_lfd()], 'mse::sampling.oem()'
[build_gear()], [catch_n_gear()], [lfd.sim()], [rUnif_lfd()], 'mse::sampling.oem()'
```

Examples

```
## Not run:
lfd.devs <- rUnif_lfd(om_gears, years = 2010:2034, nsim = 20, seed = 456)
oem <- FLoem(
  method = sampling.lfd.oem,
  observations = list(stk = stock(om), lfd = lfd_obs, catch_gear = cg_obs),
  deviances = list(stk = stk.devs, lfd = lfd.devs, catch_gear = cg.devs),
  args = list(om_gears = om_gears)
)
run <- mp(om, oem = oem, ctrl = ctrl,
  args = list(iy = 2024, fy = 2034, y0 = 2012, it = 20))

## two areas (units): gear sets by area, independent CRN streams by area,
## LFDs raised to catch by area and pooled for the assessment
oem2 <- FLoem(
  method = sampling.lfd.oem,
  observations = list(stk = stock(om2), lfd = lfd_obs2, catch_gear = cg_obs2),
  deviances = list(stk = stk.devs2,
    lfd = list(North = rUnif_lfd(gears_N, 2010:2034, 20, 1),
      South = rUnif_lfd(gears_S, 2010:2034, 20, 2)),
    catch_gear = cg.devs2),
  args = list(om_gears = list(North = gears_N, South = gears_S),
    lfd.raising = TRUE)
)

## End(Not run)
## Not run:
lfd.devs <- rUnif_lfd(om_gears, years = 2010:2034, nsim = 20, seed = 456)
oem <- FLoem(
  method = sampling.lfd.oem,
  observations = list(stk = stock(om), lfd = lfd_obs, catch_gear = cg_obs),
  deviances = list(stk = stk.devs, lfd = lfd.devs, catch_gear = cg.devs),
  args = list(om_gears = om_gears)
)
run <- mp(om, oem = oem, ctrl = ctrl,
  args = list(iy = 2024, fy = 2034, y0 = 2012, it = 20))

## two areas (units): gear sets by area, independent CRN streams by area,
## LFDs raised to catch by area and pooled for the assessment
oem2 <- FLoem(
  method = sampling.lfd.oem,
  observations = list(stk = stock(om2), lfd = lfd_obs2, catch_gear = cg_obs2),
  deviances = list(stk = stk.devs2,
    lfd = list(North = rUnif_lfd(gears_N, 2010:2034, 20, 1),
      South = rUnif_lfd(gears_S, 2010:2034, 20, 2)),
    catch_gear = cg.devs2),
  args = list(om_gears = list(North = gears_N, South = gears_S),
```

```

        lfd.raising = TRUE)
    )

    ## End(Not run)

```

schaefer.sim

schaefer.sim()

Description

generates a Schaefer surplus production model with process and observation error

Usage

```

schaefer.sim(
  k = 10000,
  r = 0.3,
  q = 0.5,
  pe = 0.1,
  oe = 0.2,
  bk = 0.9,
  years = 1980:2022,
  f0 = 0.2,
  fhi = 2.2,
  flo = 0.8,
  sigmaF = 0.15,
  iters = 1,
  blim = 0.3,
  bthr = 0.5,
  rel = FALSE
)

```

Arguments

k	carrying capacity
r	intrinsic rate of population increase
q	catchability coefficient
pe	process error
oe	process error
bk	initial fraction of b/k
years	time horizon
f0	factor for initial year as $f_0 = f/f_{msy}$
fhi	factor for high F as $f_{hi} = f/f_{msy}$
flo	factor for low F as $f_{lo} = f_{bar}/f_{msy}$
sigmaF	variation on f trajectory
iters	number of iterations
rel	if TRUE metrics B/B _{msy} and F/F _{msy} are produced

Value

FLQuants

Examples

```
stk = schaefer.sim(iters=100,q=0.5)
plotAdvice(stk)
plot(FLIndex(index=iter(stk@stock,1))) # index
```

seasonalize_catch_n_gear

Disaggregate gear catch-at-age by season

Description

Disaggregates annual gear-specific catch-at-age into equal within-year seasons under constant instantaneous fishing and natural mortality.

Usage

```
seasonalize_catch_n_gear(stock, f_age_gear, n_seasons)
```

Arguments

stock	An 'FLStock' or 'FLStockR' containing annual stock numbers, fishing mortality, and natural mortality.
f_age_gear	Named list of gear-specific fishing-mortality-at-age 'FLQuant' objects.
n_seasons	Integer number of equal within-year seasons.

Details

Population numbers at the start of season 's' are $N_s = N_0 \exp[-Z(s-1)/S]$, where 'S' is 'n_seasons'. Gear catch in that season is calculated with the corresponding partial fishing mortality.

Value

A named list of gear-specific 'FLQuant' objects with a populated season dimension.

Examples

```
## Not run:
seasonal_catch <- seasonalize_catch_n_gear(om_hist, f_age_gear, n_seasons = 4)

## End(Not run)
```

sel_a_seasonal_avg	<i>Calculate seasonally averaged selectivity-at-age</i>
--------------------	---

Description

Integrates selectivity-at-length over the length-at-age distribution at seasonal midpoint timings and averages the resulting selectivity-at-age curves across seasons.

Usage

```
sel_a_seasonal_avg(
  lhpar,
  sel_len,
  age,
  n_seasons,
  model = vonbert,
  reflen = NULL,
  cv = 0.1,
  lmin = 5,
  lmax_mult = 1.2,
  bin = 1,
  scale = TRUE
)
```

Arguments

lhpar	Life-history parameters containing 'linf', 'k', and 't0'.
sel_len	Selectivity-at-length as an 'FLQuant' or named numeric vector.
age	Numeric vector of ages.
n_seasons	Integer number of equal within-year seasons.
model	Growth function or model object. Defaults to 'vonbert'.
reflen	Optional reference length passed to [iALK()].
cv	Numeric coefficient of variation in length-at-age.
lmin, lmax_mult	Unused; the length grid is taken from the names of 'sel_len' (e.g. 'gear\$sel_len'), so it always matches the gear.
bin	Numeric length-bin width.
scale	Logical; if 'TRUE', scale the averaged curve to a maximum of one.

Details

Seasonal midpoint 's' is evaluated at $t = (s - 0.5) / n_seasons$. Within-year growth is represented by replacing 't0' with 't0 - t' in the growth parameters.

Value

A named numeric vector of seasonally averaged selectivity-at-age.

Examples

```
## Not run:
sel_a <- sel_a_seasonal_avg(
  lhpars,
  om_gears$Trawl$sel_len,
  age = 0:5,
  n_seasons = 4
)

## End(Not run)
```

sel_dnormal_len	<i>Double-normal selectivity at length</i>
-----------------	--

Description

Computes asymmetric dome-shaped selectivity as a function of length. The curve peaks at 'Lpeak', with separate standard deviations for the ascending and descending limbs.

Usage

```
sel_dnormal_len(L, Lpeak, sd_left, sd_right)
```

Arguments

L	Numeric vector of lengths.
Lpeak	Length at maximum selectivity.
sd_left	Standard deviation for lengths below or equal to 'Lpeak'.
sd_right	Standard deviation for lengths above 'Lpeak'.

Value

Numeric vector of selectivity values scaled to a maximum of 1.

sel_la

*Construct selectivity-at-length and selectivity-at-age***Description**

Evaluates a logistic, normal, or double-normal selectivity curve on a length grid and maps the same curve to predicted length-at-age.

Usage

```
sel_la(
  lhpar,
  amin = 0,
  amax = 20,
  type = c("logistic", "normal", "dnormal"),
  lmin = 3,
  lmax = NULL,
  binwidth = 1,
  lmax_mult = 1.1,
  L50 = NULL,
  L95 = NULL,
  Lpeak = NULL,
  sd = NULL,
  sd_left = 0.3,
  sd_right = 0.6,
  scale = TRUE
)
```

Arguments

lhpar	Life-history parameters containing 'linf', 'k', and 't0'.
amin, amax	Integer minimum and maximum ages. Defaults to '0' and '20'.
type	Selectivity curve: "logistic", "normal", or "dnormal".
lmin	Numeric lower limit of the length grid. Defaults to '3'.
lmax	Optional numeric upper limit of the length grid. When 'NULL', it is calculated from 'linf * lmax_mult'.
binwidth	Numeric length-bin width. Defaults to '1'.
lmax_mult	Numeric multiplier applied to 'linf' when 'lmax' is 'NULL'. Defaults to '1.1'.
L50, L95	Lengths at 50 and 95 percent selectivity for a logistic curve.
Lpeak	Length at maximum selectivity for normal and double-normal curves.
sd	Standard deviation of a normal curve.
sd_left, sd_right	Standard deviations of the ascending and descending limbs of a double-normal curve.
scale	Logical; if 'TRUE', scale the length and age curves separately to a maximum of one.

Details

Length-dimension labels contain lower bin limits, while selectivity is evaluated at bin midpoints. The returned age curve is a point evaluation at predicted length-at-age. Use [build_gear()] when selectivity-at-age must be integrated over a length-at-age distribution.

Value

A list with components 'sel_len', an 'FLQuant' of selectivity-at-length; 'sel_a', an 'FLQuant' of selectivity-at-age; and 'len_bins', a data frame of lower, upper, and midpoint lengths.

Examples

```
## Not run:
linefish <- sel_la(
  lhpar = lhpar,
  amin = 0,
  amax = 20,
  type = "logistic",
  L50 = 35,
  L95 = 50
)
gillnet <- sel_la(
  lhpar = lhpar,
  type = "normal",
  Lpeak = 35,
  sd = 10
)
traps <- sel_la(
  lhpar = lhpar,
  type = "dnormal",
  Lpeak = 30,
  sd_left = 8,
  sd_right = 25
)
plot_sel_la(traps)

## End(Not run)
```

sel_logistic_len	<i>Logistic selectivity at length</i>
------------------	---------------------------------------

Description

Computes asymptotic logistic selectivity as a function of length. The curve is parameterised by the length at 50 and the length at 95

Usage

```
sel_logistic_len(L, L50, L95)
```

Arguments

- L Numeric vector of lengths.
- L50 Length at 50% selectivity.
- L95 Length at 95% selectivity.

Value

Numeric vector of selectivity values between 0 and 1.

sel_normal_len	<i>Normal dome-shaped selectivity at length</i>
----------------	---

Description

Computes symmetric dome-shaped selectivity as a function of length. The curve peaks at ‘Lpeak’ and declines symmetrically around the peak.

Usage

```
sel_normal_len(L, Lpeak, sd)
```

Arguments

- L Numeric vector of lengths.
- Lpeak Length at maximum selectivity.
- sd Standard deviation controlling the width of the dome.

Value

Numeric vector of selectivity values scaled to a maximum of 1.

sops	<i>scales catch-at-age to total catch with error (optional)</i>
------	---

Description

scales catch-at-age to total catch with error (optional)

Usage

```
sops(object, stock, sigma = 0.1, what = c("catch", "landings", "discards")[1])
```

Arguments

object	FLQuant catch.n, discard.n, landings.n
stock	FLStock
sigma	observation error
what	type c("catch", "landings", "discards")

Value

FLQuant

spict2FLQuant	<i>spict2FLQuant()</i>
---------------	------------------------

Description

spict2FLQuant()

Usage

```
spict2FLQuant(  
  x,  
  metric = c("ssb", "fbar", "catch", "stock", "harvest")[1],  
  osa = FALSE,  
  forecast = F,  
  what = c("mle")  
)
```

Arguments

x	fit from SPICIT
osa	add one-step-ahead forecast
forecast	TRUE/FALSE
what	mle or log.sd

Value

FLQuant

Author(s)

adopted from Laurie Kell (biodyn)

spict2FLStockR	<i>spict2FLStockR()</i>
----------------	-------------------------

Description

spict2FLStockR()

Usage

```
spict2FLStockR(  
  res,  
  blim = 0.3,  
  bthr = 0.5,  
  rel = FALSE,  
  osa = FALSE,  
  forecast = NULL,  
  itsCI = 1,  
  seed = 123  
)
```

Arguments

res	fit from SPICT
blim	biomass limit reference point as fraction of Bmsy
bthr	biomass precautionary reference point as fraction of Bmsy
rel	if TRUE ratios BBmsy and FFmsy are stored
osa	add one-step-ahead forecast
forecast	extract forecast TRUE/FALSE
itsCI	number of iterations to depict uncertainty in plots
seed	random seed for consistent sampling across scenarios

Value

FLStockR with refpts

spict2stars	<i>spict2stars()</i>
-------------	----------------------

Description

spict2stars()

Usage

```
spict2stars(spict, blim = 0.3, bthr = 0.5, quantiles = c(0.05, 0.95))
```

Arguments

spict	fit from fit.spict()
blim	biomass limit point as fraction of Bmsy, default 0.3Bmsy (ICES)
bthr	biomass precautionary point as fraction of Bmsy, default 0.5Bmsy (ICES)

Value

STARS list with \$timeseris and \$refpts

spr2depletion	<i>Equilibrium depletion implied by SPR under Beverton-Holt steepness</i>
---------------	---

Description

Maps a spawning potential ratio (SPR, spawning biomass per recruit relative to unfished) to equilibrium spawning biomass relative to unfished, B/B_0 , for a Beverton-Holt stock-recruit relationship with steepness s :

$$\frac{B}{B_0} = \frac{4s \text{SPR} - (1 - s)}{5s - 1},$$

truncated at zero below $\text{SPR}_{\text{crash}} = (1 - s)/(4s)$. The mapping is the identity at $s = 1$ (constant recruitment).

Usage

```
spr2depletion(spr, s)
```

Arguments

spr	Numeric vector or FLQuant of SPR (0-1).
s	Beverton-Holt steepness, $0.2 < s \leq 1$.

Details

With $R(B) = 4sR_0B/[\phi_0R_0(1-s) + (5s-1)B]$ and the equilibrium condition $B = R\phi(F)$, solving for B and dividing by $B_0 = R_0\phi_0$ gives the equation above. Use it to express per-recruit status estimates (e.g. from length-based SPR methods) on the biomass scale when recruitment depends on stock size. Relative to an SPR target, $\text{spr2depletion}(\text{spr}, s) / \text{spr2depletion}(\text{spr_tgt}, s)$ gives B/B_{SPRtgt} .

Value

Object of the same class as `spr`.

Examples

```
spr2depletion(c(0.2, 0.4, 0.6, 1), s = 0.75)
## status relative to an SPR40 target on the biomass scale
spr2depletion(0.3, 0.75) / spr2depletion(0.4, 0.75)
```

ss2FLStockR

ss2FLStockR()

Description

`ss2FLStockR()`

Usage

```
ss2FLStockR(mvln, thin = 1, output = NULL, rel = FALSE)
```

Arguments

<code>mvln</code>	output from <code>ssmvln()</code>
<code>thin</code>	thinnig rate of retained iters
<code>output</code>	expected outputs presented as "mle" or median of "iters"
<code>rel</code>	if TRUE ratios B/B_{tgt} and F/F_{tgt} are shown

Value

FLStockR with refpts

ss2ices	<i>ss2ices()</i>
---------	------------------

Description

ss2ices()

Usage

ss2ices(mvln, quantiles = c(0.05, 0.95))

Arguments

- mvln output of ssmvln()
- quantiles default is 95CIs as c(0.025,0.975)
- output choice c("iters","mle")[1]
- Fmsy if specified the ratio F/Fmsy is calculated (required for ensembles)
- Btrigger if specified the ratio SSB/Btrigger is calculated (required for ensembles)

Value

ICES list with \$timeseris and \$refpts

ss2stars	<i>ss2stars()</i>
----------	-------------------

Description

ss2stars()

Usage

ss2stars(mvln, output = c("iters", "mle")[2], quantiles = c(0.05, 0.95))

Arguments

- mvln output of ssmvln()
- output choice c("iters","mle")[1]
- quantiles default is 95CIs as c(0.025,0.975)

Value

STARS list with \$timeseris and \$refpts

ss3col	<i>ss3col</i>
--------	---------------

Description

ss3col

Usage

ss3col(n, alpha = 1)

Arguments

n	number of colors
alpha	translucency

Value

vector of color codes

ss3devs	<i>ss3devs</i>
---------	----------------

Description

function to generate MVN assessment error deviations of F and SSB

Usage

ss3devs(om, vcv, Fphi = 0.423, bias.correct = TRUE, ...)

Arguments

om	*FLom* or *FLStock* object
vcv	covariance matrix from ss3vcv()
Fphi	autocorrelation of F error
bias.correct	lognormal bias correction if TRUE

Value

FLQuants with devs of F and SSB

Author(s)

Henning Winker (GFCM)

ss3vcv	ss3vcv
--------	--------

Description

function to generate to extract variance-covariance matrix for F and SSB

Usage

```
ss3vcv(ss3rep)
```

Arguments

ss3rep from r4ss::SS_output

Value

covariance matrix for F and SSB end year

Author(s)

Henning Winker (GFCM)

ssmvln	ssmvln()
--------	----------

Description

function to generate uncertainty for Stock Synthesis

Usage

```
ssmvln(  
  ss3rep,  
  Fref = NULL,  
  years = NULL,  
  virgin = FALSE,  
  mc = 1000,  
  weight = 1,  
  run = "MVLN",  
  addprj = FALSE,  
  ymax = NULL,  
  xmax = NULL,  
  legendcex = 1,  
  verbose = TRUE,  
  seed = 123,  
  observed.catch = FALSE  
)
```

Arguments

ss3rep	from r4ss::SS_output
Fref	Choice of Fratio c("MSY","Btgt","SPR","F01"), corresponding to F_MSY and F_Btgt
years	single year or vector of years for mvln
virgin	if FALSE (default) the B0 base for Bratio is SSB_unfished
mc	number of monte-carlo simulations
weight	weighting option for model ensembles weight*mc
run	qualifier for model run
addprj	include forecast years
ymax	ylim maximum
xmax	xlim maximum
verbose	Report progress to R GUI?
seed	retains interannual correlation structure like MCMC
observed.catch	if FALSE expected catch is used
out	choice c("iters","mle")
plot	option to show plot
legendcex=1	Allows to adjust legend cex

Value

output list of quant posteriors and mle's

Author(s)

Henning Winker (GFCM)

stock2ratios	<i>stock2ratios()</i>
--------------	-----------------------

Description

stock2ratios()

Usage

stock2ratios(object)

Arguments

object of class *FLStockR*

Value

FLStockR with ratios F/Ftgt and B/Btgt

stock2spm	<i>stock2spm()</i>
-----------	--------------------

Description

stock2spm()

Usage

stock2spm(object, brps, bthr = 0.5, blim = 0.3, rel = T, relC = FALSE)

Arguments

object of class *FLStockR*

Value

FLStockR with ratios F/Ftgt and B/Btgt

stockMedians	<i>stockMedians()</i> Converts FLStock into simplified FLStock with Median FLQuants
--------------	---

Description

stockMedians() Converts FLStock into simplified FLStock with Median FLQuants

Usage

stockMedians(object, FUN = iterMedians, ssbQ = 1, recQ = 1)

Arguments

object of class *FLStock* or *FLStockR* or *FLStocks*
FUN computes iterMedians, iterMeans
ssbQ SSB quarter seasonal models
recQ recruitment quarter seasonal models

Value

FLStockR with *FLQuants*

<code>updCatch.n</code>	<i>computes catch.n for a given harvest</i>
-------------------------	---

Description

computes catch.n for a given harvest

Usage

`updCatch.n(stk)`

Arguments

object An **FLStock**

Value

FLQuant

<code>updsr</code>	<i>updsr()</i>
--------------------	----------------

Description

updates sr in brp after changing biology

Usage

`updsr(object, s = 0.7, v = NULL)`

Arguments

object An **FLBRP**
s assumed steepness s
v input option new SB0

Value

FLBRP

Examples

```
data(ple4)
sr <- srrTMB(as.FLSR(ple4,model=bevholtsV),spr0=mean(spr0y(ple4)))
brp = FLBRP(ple4,sr)
s = sr@SV[[1]]
params(brp)
# change
m(brp) = Mlorenzen(stock.wt(brp),Mref=0.15)
brpupd =updsr(brp,s)
params(brp)
```

updstars	<i>updstars()</i>
----------	-------------------

Description

```
updstars()
```

Usage

```
updstars(star, newrefpts)
```

Arguments

- star output of star list with new reftpoints
- newrefpts FLPar manually adjusted reference points

Value

STARS list with \$timeseris and \$refpts

vbbrp	<i>Calculate vulnerable biomass from an FLBRP or FLStock object</i>
-------	---

Description

Calculates vulnerable biomass as the stock numbers-at-age multiplied by stock weight-at-age and total selectivity. Total selectivity is defined as the sum of landings and discards selectivity, scaled to a maximum of one.

Usage

```
vbbrp(x)
```

Arguments

x An object with `stock.n`, `stock.wt`, `landings.sel`, and `discards.sel` slots, typically an FLBRP or FLStock object.

Details

Vulnerable biomass is calculated as:

$$VB_y = \sum_a N_{a,y} w_{a,y} s_a$$

where s_a is total selectivity-at-age, calculated as `landings.sel(x) + discards.sel(x)` and normalised to a maximum of one.

This helper is useful when deriving an SPM-equivalent biomass quantity based on vulnerable biomass rather than spawning biomass or total biomass.

Value

An FLQuant containing vulnerable biomass by year and iteration. Units are copied from `stock(x)`.

Examples

```
## Not run:
vb <- vbbrp(brp)
plot(vb)

## End(Not run)
```

wt_a_seasonal

Calculate seasonally weighted weight-at-age

Description

Calculates weight-at-age at seasonal midpoint timings and averages across seasons using seasonal numbers-at-age as weights.

Usage

```
wt_a_seasonal(
  n_season,
  params,
  gear = NULL,
  n_seasons,
  model = vonbert,
  reflen = NULL,
  cv = gear$grid$cv %||% 0.1,
```

```

    lmin = gear$grid$lmin %||% 5,
    lmax_mult = gear$grid$lmax_mult %||% 1.2,
    bin = gear$grid$bin %||% 1
  )

```

Arguments

n_season	An 'FLQuant' containing seasonal population or catch numbers-at-age.
params	Life-history parameters containing 'linf', 'k', 't0', 'a', and 'b'.
gear	Optional gear object returned by [build_gear()]. If supplied, the age-length key is conditioned on the gear's selectivity-at-length.
n_seasons	Integer number of equal within-year seasons.
model	Growth function or model object. Defaults to 'vonbert'.
reflen	Optional reference length passed to [iALK()].
cv, lmin, lmax_mult, bin	Length-at-age CV and length grid; default to 'gear\$grid' when 'gear' is supplied, so the key matches the gear.

Details

When 'gear' is supplied, seasonal catch numbers weight a capture-conditioned age-length key. When 'gear = NULL', the biological key is weighted by seasonal population numbers.

Value

An 'FLQuant' containing seasonally weighted weight-at-age by year and iteration.

Examples

```

## Not run:
catch_wt <- wt_a_seasonal(
  seasonal_catch$Trawl,
  lhpars,
  gear = om_gears$Trawl,
  n_seasons = 4
)

## End(Not run)

```

Index

ABItgt, [4](#)
ALK, [5](#)
alk.sample, [5](#)
ALKs, [6](#)
applyALK, [7](#)
applyAR, [7](#)
asem2spm, [8](#)

bioidx.sim, [11](#)
bisect, [11](#)
blag, [14](#)
build_gear, [15](#)

ca.sim, [17](#)
catch_by_gear, [18](#)
catch_gear, [19](#)
catch_n_gear, [20](#)
combine_sel_age, [21](#)
computeFbrp, [22](#)
computeFbrps, [24](#)
condition_alk, [25](#)

f_age_gear, [25](#)
f_share_gear, [26](#)
Fbrp, [27](#)
Fe40, [27](#)
flr2stars, [28](#)
Fmmy, [28](#)
Fp05, [29](#)
Fsim, [30](#)
Fsim_grid, [31](#)
fudc, [33](#)
fwd2ices, [34](#)
fwd2stars, [35](#)
fwd2type1, [35](#)
fwdF4B, [36](#)

gear_ialk, [37](#)
gear_par, [38](#)
getF, [38](#)

huecol, [39](#)

iALK, [39](#)
idx.sim, [40](#)

jabba2FLStockR, [41](#)
jabba2stars, [42](#)
joint_sel, [42](#)

len.sim, [43](#)
len_grid, [45](#)
lfd.sim, [45](#)
lfd.sim.season, [47](#)
lfd_sop, [48](#)

merge_catch, [49](#)
Mlorenzen, [50](#)

newselex, [50](#)

opt.bisect, [51](#)

pe_sampler, [53](#)
pgquant, [54](#)
plot_condALK, [55](#)
plot_condALK_bias, [55](#)
plot_lfd_ridgeline, [56](#)
plot_lfd_season, [57](#)
plot_pe, [58](#)
plot_pf, [58](#)
plot_sel_a_seasonal, [59](#)
plot_sel_age, [60](#)
plot_sel_la, [62](#)
plotAdvice, [63](#)
plotAR, [64](#)
plotbioage, [66](#)
plotbioyr, [67](#)
plotCECAF, [67](#)
plotdyn, [69](#)
ploteq, [69](#)
plotFsim, [70](#)

plotGFCM, 71
plotMajuro, 73
plotspr, 74
plotWKREF, 75
pop_n_season, 76

rc4, 77
ref.bisect, 77
rffwd, 78
rGclass, 79
rmultinom_crn, 80
rUnif_len, 81
rUnif_lfd, 82

sam2FLQuant, 82
sam2FLStockR, 83
sam2stars, 83
sample_proc_dev, 84
sampling.lfd.oem, 85
schaefer.sim, 90
seasonalize_catch_n_gear, 91
sel_a_seasonal_avg, 92
sel_dnormal_len, 93
sel_la, 94
sel_logistic_len, 95
sel_normal_len, 96
sops, 96
spict2FLQuant, 97
spict2FLStockR, 98
spict2stars, 99
spr2depletion, 99
ss2FLStockR, 100
ss2ices, 101
ss2stars, 101
ss3col, 102
ss3devs, 102
ss3vcv, 103
ssmvln, 103
stock2ratios, 104
stock2spm, 105
stockMedians, 105

updCatch.n, 106
updsr, 106
updstars, 107

vbbrp, 107

wt_a_seasonal, 108