Package: mmsyFM (via r-universe)

November 5, 2026

Title Multispecies MSY Fleet Model for Mixed-Fishery Harvest Strategies

Version 0.1.0

Description Simulates mixed fisheries in which fleet segments (e.g. demersal, pelagic and mixed or small-scale) allocate their fishing days across metiers by expected profit, on top of Schaefer surplus-production stocks with seasonal availability and JABBA-type process error. Finds the multispecies maximum sustainable yield (MMSY) subject to a safeguard for every stock, tunes the fleet to an open-access status quo, and tunes and tests effort- and closure-based harvest strategies with assessment and implementation error, data lags, species complexes and nominal assignment of stocks to fleets.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1)

Imports data.table, ggplot2 (>= 3.4.0), stats, utils

Suggests commonmark, ggrepel, knitr, patchwork, rmarkdown, shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/Henning-Winker/mmsyFM

BugReports https://github.com/Henning-Winker/mmsyFM/issues

Config/roxygen2/version 8.1.1

Repository https://henning-winker.r-universe.dev

Date/Publication 2026-10-23 11:33:37 UTC

RemoteUrl https://github.com/Henning-Winker/mmsyFM

RemoteRef HEAD

RemoteSha 25bf4a05443dca8c1bbb75c6e431e24341d61b07

Contents

atlantic_system . . . 3
ffmsy . . . 3
got_species . . . 4
got_system_draft . . . 5
hcr_effort . . . 6
ms_assessment_presets . . . 8
ms_closure . . . 9
ms_condition . . . 9
ms_landscape . . . 11
ms_open_access . . . 12
ms_optimise . . . 13
ms_perf . . . 14
ms_perf_table . . . 15
ms_shared_q_curve . . . 16
ms_simulate . . . 16
ms_statistics . . . 19
ms_summary . . . 20
ms_system . . . 20
ms_tune . . . 21
ms_tune_hcr . . . 23
ms_yield_curve . . . 24
perf_rows . . . 25
plot_bps . . . 26
plot_catchability . . . 26
plot_fleet . . . 27
plot_kobe . . . 27
plot_landscape . . . 29
plot_projection . . . 29
plot_quilt . . . 30
plot_tradeoffs . . . 31
plot_yield_curves . . . 31
proc_sd_from_r . . . 32
read_ms_system . . . 32
relsig_jabba . . . 33
run_app . . . 33
theme_ms . . . 34
write_ms_system . . . 34

Index **35**

atlantic_system *Atlantic example system*

Description

Ten stocks (six demersal, four pelagic), six métiers (reef handline, sediment longline and pelagic troll, each on a shallow and a deep ground) and three segments (demersal and pelagic specialists, and a mixed generalist fishing at 0.75 efficiency). The catchability matrix was derived from the habitat landscape of `cpueTargeting::simulate_logbook()` (seed 1). Daily costs are tuned so that the open-access equilibrium has 35/35/30% of days by segment and the weakest stock at 0.4 Bmsy (see `ms_tune()`).

Usage

```
atlantic_system()
```

Details

Dentex (narrow reef habitat, high catchability relative to its growth rate) is the choke stock.

Value

An `ms_system`.

Examples

```
sys <- atlantic_system()
sq <- ms_status_quo(sys, capacity = 1000, gamma = 1.5, years = 200)
round(sq$BBmsy, 2)
```

ffmsy *F relative to Fmsy in the discrete Schaefer model*

Description

Growth is computed on start-of-year biomass and a year of fishing removes B (1 - exp(-F)), so equilibrium is B/K = 1 - h/r with harvest rate h = 1 - exp(-F), hmsy = r/2 and B/Bmsy = 2 - h/hmsy exactly. `ffmsy()` returns h/hmsy. (Using r/2 for instantaneous F overstates F/Fmsy for fast stocks: for r = 1.4, Fmsy = -log(1 - r/2) = 1.2, not 0.7.)

Usage

```
ffmsy(F, r, h = FALSE)
```

Arguments

F	instantaneous annual F (vector, or matrix with stocks in columns).
r	intrinsic growth rate(s), one per stock.
h	if TRUE, F is already a harvest rate.

Value

h / hmsy, same shape as F.

Examples

```
ffmsy(-log(1 - 0.7), r = 1.4)   # 1: fished at MSY
```

---

got_species *Gulf of Thailand species groups (mmsyapp input)*

Description

The 18 species groups of the multispecies MSY app (mmsyapp; Amoroso, Pons and Sharma) for the Gulf of Thailand: K, r, 2024 depletion (Dep_ini, B/K), a placeholder catchability q and a group label. This is the starting point for a Gulf of Thailand system (trawl, purse seine and small-scale segments); métiers and catchability are still to be defined.

Usage

```
got_species()
```

Value

A data frame.

Examples

```
head(got_species())
```

got_system_draft *Gulf of Thailand system (draft)*

Description

Draft, indicative only. Twenty stocks (the 18 mmsyapp groups plus anchovies and sardines), ten métiers and three segments:

- trawl: inner (15-30 m, mud), mid (30-50 m) and outer shelf (50-80 m, sand) grounds;
- purse seine: mackerel/scad, and light-luring anchovy/small pelagics;
- small-scale: five coastal métiers (crab, shrimp, pelagic gillnet, hook and trap, cephalopod), confined to the coastal zone. Small-scale is the generalist, switching freely between its five tactics, with low efficiency (0.075 of an industrial vessel-day) and a matching low daily cost.

Usage

```
got_system_draft()
```

Details

Every segment has loyalty 1 (zoning): it switches only among its own métiers.

Catchability starts from an expert overlap matrix (main fishery 1, by-catch 0.2, scaled by r) and was then conditioned with `ms_condition()` so that each stock's status quo matches its mmsyapp 2024 depletion (`dep_2024`), with B/Bmsy limited to 0.05-1.5: shrimp is set to 1.5 (mmsyapp: 0.97 B/K), croakers end at 0.07 (target 0.12, at the F cap), anchovies and sardines are free (about 0.75 Bmsy). Daily costs keep the days at 12/6/82% (trawl / purse seine / small-scale; capacity 450 / 225 / 3000 days, `gamma = 1.5`). The median status is 0.46 Bmsy. All stocks have `F_cap = 1.95` (a floor at 0.05 Bmsy) and simulations start from the conditioned depletions (`dep_init`). Squid is shared by small-scale, trawl and the light-luring seine, and scads by both seines and trawl. Prices are relative and indicative.

Value

An `ms_system`.

Examples

```
sys <- got_system_draft()
sys

sq <- ms_status_quo(sys, capacity = c(450, 225, 3000), gamma = 1.5, years = 300)
round(sort(sq$BBmsy), 2)
```

`hcr_effort` *Segment effort rule with an optional choke-stock closure*

Description

Segment effort rule with an optional choke-stock closure

Usage

```
hcr_effort(
  sys,
  Btrig = 0.5,
  Blim = 0.25,
  assess_cv = 0.2,
  assess_rho = 0,
  own_q = 0.5,
  close = NULL,
  close_q = 0.5,
  max_change = Inf,
  Emin = 0.1,
  Btrig_close = Btrig,
  assessed = NULL,
  assign = NULL,
  complexes = NULL,
  complex_index = c("biomass", "cpue"),
  aggregate = c("weighted", "min"),
  seed = 1
)
```

Arguments

`sys`	an `ms_system`.
`Btrig`, `Blim`	trigger and limit relative to Bmsy (effort rule). The default trigger is 0.5 Bmsy, i.e. at Bpa, as is common in ICES and GFCM advice. `Btrig` may be named by segment to set triggers by segment (e.g. `c(trawl = 0.5, purse_seine = 0.8, small_scale = 0.5)`; unnamed segments get 0.5), or be a vector with one value per simulation row (a grid of triggers, as in `ms_tune_hcr()`).
`assess_cv`, `assess_rho`	assessment error CV and its AR(1).
`own_q`	a segment answers for a stock it catches as a main fishery: one of its metiers has >= `own_q` (default 0.5) of the stock's highest catchability. Lower values (e.g. 0.2) also count by-catch.
`close`	stocks or complexes protected by a closure.
`close_q`	métiers with >= `close_q` of the stock's highest catchability are closed.

`max_change` limit on the annual change of each segment's days, relative to the previous year (0.2 = at most 20% up or down), as for TAC changes. The first year starts from the reference days (fraction 1); a segment at zero (with `Emin = 0`) can reopen from 5% of its reference days.

`Emin` minimum effort: the lowest fraction of its reference days a segment is set to (at and below Blim), default 0.1. F scales with days, so the floor puts about `Emin` x the reference F on each stock; 0 lets the rule close a segment completely.

`Btrig_close` trigger of the closure.

`assessed`, `assign`, `complexes`, `complex_index` see Details.

`aggregate` how a segment's status is formed from the assessed B/Bmsy of the units it answers for: `"weighted"` (default), the mean weighted by each unit's potential revenue for the segment (catchability on its metiers x price x K), so the stocks that make up its fishery drive its effort; or `"min"`, the weakest link (the most depleted unit sets the segment's effort), a special, most precautionary case.

`seed` seed of the rule's own random stream (assessment errors), kept apart from the simulation's, so rules compared with the same simulation seed see the same process and implementation errors.

Details

Each year the assessed B/Bmsy of every stock is the true value times a lognormal assessment error (CV `assess_cv`, AR(1) `assess_rho`). Each segment's days are a fraction of its reference days, the caps passed to ms_simulate(): normally the safeguarded-MMSY effort from `ms_optimise()` (the "full" effort above the trigger), or the status-quo days. The fraction ramps from `Emin` at Blim (and below) to 1 at Btrig (both relative to Bmsy) of the segment's status: the value-weighted mean (`aggregate = "weighted"`) or the minimum (`"min"`, weakest link) of the assessed B/Bmsy of the units it catches as a main fishery (`own_q`). With `max_change`, the first year starts from the current days relative to the reference days, so the move from the status quo to the reference effort is phased.

Stocks in `close` are protected by a closure instead: the open share of the metiers that drive them (catchability >= `close_q` of the stock's highest) follows the same ramp, and they are left out of the effort rule. `max_change` limits the annual change of each segment's days relative to the previous year.

`Btrig` (effort rule) and `Btrig_close` (closure) may be vectors with one value per row of the simulation, so a grid of triggers runs in one call.

`assign`: nominal assignment of stocks to segments for management, a named list segment -> stock names (as in practice: a species is "managed" through one fleet although others catch it). It replaces the catchability-based responsibility; segments without stocks are not constrained by the rule.

`complexes`: named list of stock complexes assessed as one unit (pooled taxa, e.g. reef fish), the remaining stocks are single units. The complex index is pooled B/Bmsy (`complex_index = "biomass"`: sum B / sum Bmsy, as a production model fitted to pooled catch and biomass) or a pooled catch rate (`"cpue"`: taxa weighted by their catchability). `close`, `assign` and `assessed` then refer to unit names (complexes or stocks).

`assessed`: names of the stocks with an assessment (default all). The rule sees only these; a stock in `close` must be assessed.

Create a new rule for every simulation run (it keeps the last fraction and the assessment error between years).

Value

A rule function for the `hcr` argument of `ms_simulate()`, with attributes `own`, `drive` and `units`.

`ms_assessment_presets` *Default assessment strategies: indicator species, complexes and assignment*

Description

Starting points for the `assessed`, `complexes` and `assign` arguments of `hcr_effort()`, derived from the system:

Usage

```
ms_assessment_presets(sys, own_q = 0.5)
```

Arguments

`sys`	an `ms_system`.
`own_q`	as in `hcr_effort()`.

Details

- `indicators`: one indicator species per segment, the stock of highest potential value to that segment (catchability on its metiers x price x K) among the stocks it catches as a main fishery, without repeats;
- `complexes`: the species groups of the system with two or more stocks (stocks alone in their group stay single units);
- `assign`: each stock assigned to the segment for whose fishery it matters most (its largest share of the segment's potential value), its "main" fleet.

Value

A list with `indicators` (stock names), `complexes` (named list of stock names) and `assign` (named list segment -> stock names).

Examples

```
ms_assessment_presets(atlantic_system())
```

ms_closure *Fixed metier closures*

Description

Builds the open share of each metier by month (a 12 x metier matrix) for closures of one or more metiers, all year or for some months, and of all or part of their ground. Pass it as `open` to `ms_simulate()` (or multiply it with another open matrix). Displaced days move to the open metiers by expected profit.

Usage

```
ms_closure(sys, metiers, months = 1:12, share = 1)
```

Arguments

`sys`	an `ms_system`.
`metiers`	names of the metiers to close.
`months`	months closed (1-12; default all year), e.g. `2:5` for February to May.
`share`	share of each metier closed in those months (1 = fully; 0.5 = half its ground, as for an area closure).

Value

A 12 x metier matrix of open shares (1 = open).

Examples

```
sys <- got_system_draft()
op <- ms_closure(sys, c("TR_inner", "TR_mid"), months = 2:5)   # trawl closed Feb-May
op[1:6, c("TR_inner", "TR_mid", "TR_outer")]
```

ms_condition *Condition stock status on target depletions*

Description

Rescales each stock's catchability (its column of `sys$Q`, across all metiers) so that the open-access equilibrium reaches a target B/Bmsy per stock, while `ms_tune()` keeps the segment shares of days at `sys$segments` with total days held fixed (the overall effort level and the catchability scale are not separately identifiable).

Usage

```
ms_condition(
  sys,
  target,
  range = c(0.05, 1.5),
  cap = 1000,
  gamma = 1.5,
  outer = 4,
  inner = 12,
  tol = 0.02,
  damp = 0.8,
  years = 300,
  verbose = TRUE,
  recost = TRUE,
  ...
)
```

Arguments

sys	an ms_system (costs already roughly tuned, e.g. by ms_tune()).
target	named vector of target B/Bmsy (names = stock names or codes); NA or missing stocks are free.
range	lower and upper limits of the targets.
cap	fleet capacity (days) by segment.
gamma	profit preference.
outer, inner	number of outer (cost tuning) and inner (catchability) iterations.
tol	tolerance on |B/Bmsy - target| for the inner loop.
damp	damping exponent of the catchability update.
years	years to equilibrium.
verbose	print progress.
recost	if TRUE (default), each metier's daily cost keeps its ratio to the metier's unfished revenue rate as catchability changes, so a metier whose main stock gets a lower catchability does not keep a cost sized for the old one (which would leave it unfished).
...	passed to ms_simulate().

Details

In the discrete Schaefer model B/Bmsy = 2 - F/Fmsy (with F/Fmsy as h/hmsy, see ffmsy()), so each inner step multiplies a stock's catchability by ((2 - target) / current F/Fmsy)^damp. Stocks with an NA target are left free. Targets are clamped to range.

Value

The conditioned ms_system, with attribute conditioning: a data frame of targets and achieved B/Bmsy and F/Fmsy, and the catchability multipliers.

`ms_landscape` *Catchability from a habitat landscape*

Description

Builds the catchability matrix from where stocks live and where (and how) metiers fish, instead of writing it by hand:

Usage

```
ms_landscape(cells, species, metiers, selectivity, q0 = 0.001)
```

Arguments

`cells`	data frame with `depth`, `substrate` and `area` (any units).
`species`	data frame with `species`, `depth_mean`, `depth_sd` and one column per substrate level (weights, e.g. `mud`, `sand`, `hard`; missing levels get weight 0, all missing = 1).
`metiers`	data frame with `metier`, `depth_min`, `depth_max` and optional `substrate` (levels separated by `;`, empty = all).
`selectivity`	gear selectivity: a matrix (metiers x species, with dimnames) or a data frame with a `metier` column and one column per stock. Values 0-1.
`q0`	catchability at average density and selectivity 1.

Details

- the seabed is a set of `cells` with a depth, a substrate and an area;
- each stock's habitat suitability in a cell is a normal depth preference times a substrate weight; its distribution over the cells is suitability x area, normalised to 1;
- each metier fishes a ground (a depth range and a set of substrates) with a gear selectivity for each stock.

The catchability of metier j for stock s is

```
q[j, s] = q0 x sel[j, s] x (share of s on ground j / area share of ground j)
```

i.e. q0 x selectivity x the stock's density on the ground relative to its average density. A stock spread evenly over the whole seabed has q = q0 x sel on every ground. The level of each stock's column is usually set afterwards by `ms_condition()`; the landscape gives its shape (which metiers catch the stock, and how strongly relative to each other).

Value

A catchability matrix (metiers x stocks) with attributes `distribution` (cells x stocks), `ground` (cells x metiers, 0/1) and `overlap` (share of each stock on each ground).

See Also

plot_landscape()

Examples

```
cells <- expand.grid(depth = seq(2.5, 77.5, 5), substrate = c("mud", "sand"))
cells$area <- 1
sp <- data.frame(species = c("A", "B"), depth_mean = c(15, 60), depth_sd = 10,
                 mud = c(1, 0.2), sand = c(0.2, 1))
mt <- data.frame(metier = c("inner", "outer"), depth_min = c(0, 40), depth_max = c(40, 80))
sel <- matrix(1, 2, 2, dimnames = list(mt$metier, sp$species))
round(ms_landscape(cells, sp, mt, sel), 5)
```

---

ms_open_access *Open-access and status-quo equilibria*

Description

ms_open_access() runs the system to equilibrium with the caps set to the fleet capacity, so the days fished are set by profit (the vessels that break even). ms_status_quo() is the same with capacity days per segment.

Usage

```
ms_open_access(sys, years = 500, cap = 1000, ...)

ms_status_quo(sys, capacity = 1000, ...)
```

Arguments

sys	an ms_system.
years	years to equilibrium.
cap	capacity (days) by segment.
...	passed to ms_simulate().
capacity	days by segment (scalar or vector).

Value

A list with E_seg (days by segment), B, summary, BBmsy and FFmsy.

`ms_optimise` *Maximise an equilibrium objective over segment effort caps*

Description

Finds the effort caps that maximise total yield (or revenue, or profit) at equilibrium subject to every constrained stock staying at or above Bpa: B_i / Bmsy_i >= Bpa for all i in `species`.

Usage

```
ms_optimise(
  sys,
  objective = "yield",
  levers = names(sys$segments),
  fixed = NULL,
  ratio = rep(1/sys$G, sys$G),
  upper = 3000,
  Bpa = 0.5,
  species = sys$species,
  min_share = 0,
  n = NULL,
  zoom = 5,
  years = 150,
  ...
)
```

Arguments

`sys`	an `ms_system`.
`objective`	what is maximised: `"yield"` (total catch), `"revenue"`, `"profit"`, or the catch of one or more stocks given by code or name (e.g. `"ANC"` to maximise the anchovy catch), always subject to the safeguard for the stocks in `species`.
`levers`	segments whose caps are optimised, or "total" for one global cap split in proportion `ratio` across all segments
`fixed`	named caps for segments not in `levers` (e.g. the free agent: mixed = 40)
`ratio`	split of the global cap when levers = "total"
`upper`	upper bound of each cap
`Bpa`	safeguard relative to Bmsy.
`species`	names of the stocks under the safeguard (default all)
`min_share`	viability floor: every segment's cap must be at least this share of the total cap (e.g. 0.2). Applies to caps by segment only (with `levers = "total"` the split is fixed by `ratio`).
`n, zoom`	grid points per dimension and number of zoom steps.
`years`	years to equilibrium.
`...`	passed to ms_simulate() (gamma, behaviour, phi, open, ...)

Details

The search is a vectorised zooming grid in log1p(effort) space: a grid of n^d candidates is run in one call, the best feasible one is kept, and the box shrinks around it. This is robust to the kinks and small cycles of a behavioural fleet, which defeat gradient-based optimisers.

Value

list: caps, summary row, B/Bmsy, F/Fmsy and catch by stock, the value of the objective, effort by segment and metier, the binding stock(s), and the search history

---

ms_perf *Performance statistics in the mseviz format*

Description

Computes performance statistics of one or more projections (named list of `ms_simulate()` runs, one per management procedure or rule) and returns them as a long `data.table` in the format of the mseviz package: one row per run (`mp`), unit (`biol`: a stock code, a segment, or `"All"` for the whole system), statistic and iteration, with the value in `data`. The table can be passed to `plot_bps()`, `plot_tradeoffs()` and `plot_quilt()`, or to `mseviz::plotBPs()`, `mseviz::plotTOs()` and `mseviz::perfTable()`.

Usage

```
ms_perf(runs, years = NULL, Blim = 0.25, Bpa = 0.5)
```

Arguments

`runs`	named list of `ms_simulate()` results (with `keep = "all"`), one per management procedure; all from the same system.
`years`	evaluation years (default the last 20).
`Blim, Bpa`	limit and precautionary biomass relative to Bmsy.

Details

Statistics are computed per iteration over the evaluation `years`:

statistic	unit	definition
`BBmsy`	stock	median B/Bmsy
`FFmsy`	stock	median F/Fmsy
`CMSY`	stock	mean catch / MSY (MSY = rK/4)
`PBlim`	stock	share of years with B < Blim
`PBpa`	stock	share of years with B < Bpa
`PB80`	stock	share of years with B > 0.8 Bmsy
`green`	stock	share of years in the Kobe green zone (B > Bmsy, F < Fmsy)
`Days`	segment, All	mean days fished
`C`	All	mean total catch

Rev	All	mean revenue
Profit	All	mean profit
CMMSY	All	mean total catch / sum of single-stock MSYs
AAVC	All	mean annual absolute change of total catch (relative)
AAVE	All	mean annual absolute change of total days (relative)
NBpa	All	mean number of stocks below Bpa
PanyBlim	All	share of years with any stock below Blim
minBBmsy	All	lowest B/Bmsy of any stock in any year

Probabilities (statistics starting with P and green) are best shown by their mean over iterations, which is the probability over years and iterations.

Value

A data.table with columns mp, biol, statistic, name (plotmath label), desc, type ("value" or "prob"), year (the evaluation period), iter and data.

See Also

ms_perf_table(), plot_bps(), plot_tradeoffs(), plot_quilt()

---

ms_perf_table *Summary (decision) table of performance statistics*

Description

Medians (or means, for probabilities) of each statistic by management procedure, with the 10th and 90th percentiles.

Usage

```
ms_perf_table(
  perf,
  statistics = NULL,
  biol = "All",
  probs = c(0.1, 0.9),
  digits = 3
)
```

Arguments

perf	output of ms_perf().
statistics	statistics to include (default all in perf for biol).
biol	unit: "All" (system), a segment or a stock code; several values give one block per unit.
probs	lower and upper quantiles.
digits	significant digits.

Value

A data frame: one row per mp (and biol), one column per statistic, cells "median (lower-upper)".

---

ms_shared_q_curve *Shared-q yield curve (the mmsyapp model)*

Description

The classical multispecies model with one effort and a fixed catchability vector, calibrated so that it reproduces the fishing mortality of a reference state: q_i = F_i / E, with B_i/K_i = 1 - q_i E / r_i at equilibrium. The F cap of the system (F_cap) applies as in ms_simulate(), so no stock falls below 2 - F_cap Bmsy.

Usage

```
ms_shared_q_curve(
  sys,
  ref,
  mult = 10^seq(-1.5, log10(4), length.out = 60),
  Bpa = 0.5
)
```

Arguments

sys	an ms_system.
ref	a reference state from ms_open_access() or ms_status_quo() (uses its FFmsy and E_seg).
mult	multipliers of the reference total days.
Bpa	safeguard relative to Bmsy.

Value

A data frame as ms_yield_curve().

---

ms_simulate *Simulate the system forward*

Description

Annual Schaefer growth on start-of-year biomass; catches removed monthly (Baranov-type, C = B (1 - exp(-F))). Each month every segment allocates its effort among metiers by expected profit computed from the biomass it perceives (info: "current" = true current biomass, "lagged" = last year's realised CPUE by metier).

Usage

```
ms_simulate(
  sys,
  caps,
  years = 100,
  gamma = 1.5,
  behaviour = c("dynamic", "fixed"),
  info = c("current", "lagged"),
  phi = 0.5,
  open = 1,
  hcr = NULL,
  hcr_lag = 2,
  B_init = NULL,
  E_init = NULL,
  capacity = NULL,
  cost_seg = NULL,
  proc_cv = 0,
  rho = 0,
  proc_r_power = 1,
  proc_rho_sp = 0,
  proc_sd = NULL,
  proc_dep = c("none", "jabba", "power"),
  proc_dep_pars = c(a = 0.128, c = 0.223, d = 1.326),
  proc_dep_power = 0,
  proc_dep_ref = 0.5,
  proc_sd_max = 1,
  proc_center = c("mean", "median"),
  impl_cv = 0,
  impl_bias = 0,
  effort_cv = 0,
  effort_rho = 0,
  seed = NULL,
  crn = NULL,
  keep = c("final", "all"),
  avg_years = 10,
  F_cap = sys$F_cap %||% Inf,
  F_cap_p = NULL
)
```

Arguments

sys	an ms_system.
caps	n x G matrix of annual effort caps by segment
years	number of years.
gamma	preference exponent (0 = profit-blind)
behaviour	"dynamic" (reallocate every month) or "fixed" (shares frozen at the unfished allocation: the classical shared-q model)

`info` expected value from the current biomass or last year's catch rates.

`phi` entry/exit rate of each segment: E_t+1 = min(cap, E_t * exp(phi * margin_g)), margin_g = segment profit / segment revenue; 0 = effort always at the cap

`open` métier access, 0-1 (the open share of each métier's ground or season): a vector of length M (all months), a 12 x M matrix (by month, e.g. a seasonal closure), or a function of the year returning either.

`hcr` optional function(state) -> n x G multiplier on caps

`hcr_lag` years between the data a rule sees and the year its advice applies (default 2: data of year y - 2, assessed in y - 1, advice for y). Before the projection the data years are taken to be the initial state.

`B_init` initial biomass (n x S, or one value per stock); default `sys$dep_init * K` (the system's starting depletion, 1 if not set)

`E_init` initial effort by segment (vector of length G or n x G; default the caps)

`capacity` fleet days by segment (registered vessels x days); default the caps. With vessel heterogeneity (sys$vessel_cv > 0) and phi > 0, the days used move towards capacity x the share of vessels that break even (lognormal vessel efficiency, mean 1), and the active vessels' mean efficiency is the mean of that upper tail. phi is then the annual adjustment rate (0-1).

`cost_seg` daily cost by segment (length G or n x G), overrides sys$cost_seg

`proc_cv` process error CV at the median r (JABBA form: the whole process equation is multiplied by exp(eta), eta ~ N(-sigma^2/2, sigma^2))

`rho` AR(1) autocorrelation of the process error in time

`proc_r_power` the process CV of species i is proc_cv * (r_i / median r)^power: more productive species are more variable (0 = same CV for all)

`proc_rho_sp` correlation of the process error across species (shared environment)

`proc_sd` log-scale process SD by species (length S), overrides proc_cv and proc_r_power; e.g. proc_sd_from_r(sys$r) (0.05 for sharks, 0.4 for anchovy)

`proc_dep` depletion-dependent process error: "jabba" = JABBA proc.dyn (jabba2jags.R): sd_t = sd_i * relsig(B/Bmsy), relsig(x) = raw(x) / raw(1), raw(x) = a + (1 - a) / (1 + (x / c)^d), empirical ICES a = 0.128, c = 0.223, d = 1.326 (`proc_dep_pars`); relsig = 1 at Bmsy, ~1.5 at 0.5 Bmsy, -> 4.3 at 0. "power": see proc_dep_power. Capped at proc_sd_max.

`proc_dep_pars` parameters a, c, d of the JABBA `proc.dyn` function.

`proc_dep_power`, `proc_dep_ref`, `proc_sd_max`
power form (proc_dep = "power"): sd_i,t = sd_i * ((B_i,t / K_i) / proc_dep_ref)^(-proc_dep_power), capped at proc_sd_max; sd_i applies at B/K = proc_dep_ref (Bmsy). proc_dep_power > 0: stocks become more volatile at low abundance; < 0: less volatile

`proc_center` "mean": exp(sd eps - sd^2/2) (mean-unbiased, the median drifts down as sd grows: stochastic depensation); "median": exp(sd eps), as in JABBA's dlnorm(Pmean, iPV) process equation

`impl_cv`, `impl_bias`
implementation error on the effort caps by segment: realised cap = cap * exp(bias + N(-s^2/2, s^2)), s from impl_cv

`effort_cv`, `effort_rho`
: effort/catchability noise by segment, in every run (regulated or not): an annual lognormal multiplier on effective effort (weather, fuel, markets, catchability), AR(1) effort_rho, mean 1. Unlike the implementation error it acts when no cap binds.

`seed`
: random seed.

`crn`
: common random numbers: rows i and i + crn share the same process (and rule) error draws, e.g. crn = nsim when the rows are nsim simulations repeated for each of several control settings

`keep`
: "final" (equilibrium summaries) or "all" (trajectories)

`avg_years`
: years kept when `keep = "final"`.

`F_cap`
: cap on each stock's realised F, in units of Fmsy (scalar or one per stock; default `sys$F_cap`, else `Inf`). F/Fmsy is measured as the harvest-rate ratio h/hmsy (see `ffmsy()`); the cap on h is converted to instantaneous F and applied monthly as one twelfth. Extra effort on a capped stock catches nothing more, so its catch per day falls (economic extinction: the stock is left to rare by-catch). In Schaefer equilibrium B/Bmsy = 2 - F/Fmsy, so a cap below 2 sets a floor: `F_cap = 1.95` keeps every stock at or above 0.05 Bmsy.

`F_cap_p`
: shape of the cap: `Inf` = hard cap, min(F, F_cap); a finite p = smooth cap, (F^-p + F_cap^-p)^(-1/p), which approaches F_cap without a kink (p = 8 lowers F by about 1.5% at 1.5 Fmsy when F_cap = 1.95). Default `NULL`: hard without process error (equilibrium and tuning), p = 8 with process error (stochastic runs).

Value

An object of class `ms_sim`: arrays `B`, `C`, `F` (years x rows x stocks), `E_metier`, `E_seg_metier`, `E_seg`, matrices `revenue`, `profit`, the system, and `init` (the year-0 biomass and days, rows x stocks / segments).

`ms_statistics` *Definitions of the performance statistics*

Description

Definitions of the performance statistics

Usage

```
ms_statistics()
```

Value

A data frame with `statistic`, `name` (plotmath), `desc`, `type` (`"value"` or `"prob"`) and `higher_better` (direction used to colour decision tables; `NA` = no preferred direction).

ms_summary *Summaries over the kept years*

Description

Means over the kept years of an `ms_simulate()` run (the equilibrium when `keep = "final"` and the run is long enough).

Usage

```
ms_summary(x, Bpa = 0.5, Blim = 0.25)
```

Arguments

`x`	an `ms_sim` object.
`Bpa`, `Blim`	reference points relative to Bmsy.

Value

A data frame with one row per simulation row (yield, revenue, profit, days, lowest B/Bmsy, ...), with attributes `BBmsy`, `FFmsy` and `catch` (rows x stocks).

ms_system *Build a multispecies fleet system from tables*

Description

Defines the stocks, métiers, catchability, fleet segments and costs that all other functions use.

Usage

```
ms_system(
  species,
  metiers,
  catchability,
  segments,
  cost_share = 0.4,
  effort_season_amp = 0,
  F_cap = Inf
)
```

Arguments

species	data frame, one row per stock, with columns species, r, K and price; optional code (short label), group, season_amp (0 = resident, 1 = absent at the low point), season_peak (month of highest availability), F_cap (cap on realised F in units of Fmsy, see ms_simulate()) and dep_init (starting B/K of simulations, default 1; set by ms_condition()).
metiers	data frame, one row per métier, with column metier; optional tactic, ground and cost (cost per day of a vessel with efficiency 1). Without cost, costs are cost_share times each métier's unfished revenue rate.
catchability	catchability of each stock in each métier: a matrix (métiers x stocks, with dimnames) or a data frame with a metier column and one column per stock.
segments	data frame, one row per fleet segment, with columns segment, loyalty (share of days on home métiers) and home (home métiers separated by ;, empty = all); optional efficiency (on home métiers, default 1), efficiency_away (default efficiency), cost_multiplier (daily cost relative to a vessel of efficiency 1, default efficiency), vessel_cv (spread of vessel efficiency within the segment, default 0.25), share_target (target share of days, used by ms_tune()) and capacity (registered days, the default fleet capacity of the segment, e.g. for the app).
cost_share	used when metiers has no cost column.
effort_season_amp	seasonal amplitude of effort (0 = equal months; effort peaks in July).
F_cap	cap on realised F (units of Fmsy) for stocks without an F_cap column value; Inf = no cap, 1.95 = floor at 0.05 Bmsy.

Value

An object of class ms_system.

See Also

atlantic_system() for a ready-made example.

Examples

```
sys <- atlantic_system()
sys
```

---

ms_tune *Tune daily costs so that a target fleet composition is an open-access equilibrium*

Description

At open access each segment's active days are the share of its vessels that break even (heterogeneous vessel efficiency), so the composition of the active fleet is set by the daily costs. The G log cost multipliers are solved so that G residuals vanish: the shares of days of the first G - 1 segments and one status target for the stocks, r = (log share_g / target_g, g < G; log status / target). Newton steps with a forward-difference Jacobian; the base point and the G perturbed points run in one vectorised call (warm-started).

Usage

```
ms_tune(
  sys,
  shares = sys$segments,
  status = c(median = 0.9),
  cap = 1000,
  max_iter = 25,
  tol = 0.02,
  h = 0.05,
  max_step = 0.5,
  years_first = 400,
  years = 250,
  avg_years = 30,
  verbose = TRUE,
  ...
)
```

Arguments

sys	an ms_system.
shares	target share of active days by segment (default the share_target of the segment table)
status	c(median = x): median B/Bmsy; c(min = x): the weakest stock; c(E_total = x): total active days
cap	fleet capacity (registered days) by segment
max_iter, tol, h, max_step	Newton settings.
years_first, years, avg_years	years of the first and later runs, and years averaged.
verbose	print progress.
...	passed to ms_simulate() (gamma, ...)

Value

the tuned system (cost_seg replaced), with attributes "tuning" (history), "status_quo" (the open-access equilibrium) and "converged"

`ms_tune_hcr` *Tune the triggers of hcr_effort() to a risk criterion*

Description

Runs a grid of effort-rule triggers (and closure triggers, if `close` is given) in one vectorised call with common random numbers: every grid point sees the same process and assessment errors. The tuned rule maximises the objective among the grid points that meet the risk limit.

Usage

```
ms_tune_hcr(
  sys,
  triggers = seq(0.5, 1.2, by = 0.05),
  close = NULL,
  close_triggers = NULL,
  risk = c(worst_P_below_Bpa = 0.05),
  objective = "yield",
  E_ref,
  B_init,
  E_init = E_ref,
  nsim = 200,
  years = 50,
  eval_years = 31:50,
  seed = 1,
  Blim = 0.25,
  assess_cv = 0.2,
  max_change = Inf,
  Emin = 0.1,
  assessed = NULL,
  assign = NULL,
  complexes = NULL,
  complex_index = "biomass",
  aggregate = "weighted",
  own_q = 0.5,
  ...
)
```

Arguments

`sys`	an `ms_system`.
`triggers`	effort-rule triggers (B/Bmsy)
`close`	stock or complex protected by a closure (see `hcr_effort()`).
`close_triggers`	closure triggers; NULL = same as the effort trigger

risk	named limit on a perf_rows() column, e.g. c(worst_P_below_Bpa = 0.05) (every stock below Bpa at most 5% of the time) or c(worst_P_assessed = 0.05) (only the assessed stocks)
objective	perf_rows() column to maximise ("yield" or "profit")
E_ref	reference days by segment (the caps the rule scales), e.g. the safeguarded-MMSY effort from ms_optimise() or the status-quo days
B_init, E_init	initial biomass and days (the status quo).
nsim, years, eval_years, seed	simulations, years, years evaluated, seed.
Blim, assess_cv, max_change, Emin, complexes, complex_index, assign, aggregate, own_q	passed to hcr_effort().
assessed	stocks with an assessment (default all)
...	passed to ms_simulate() (gamma, capacity, proc_cv, ...)

Value

A list with table (performance by grid point), best (the tuned triggers) and the criteria.

---

ms_yield_curve *Equilibrium yield curve over effort*

Description

Scales a reference effort vector (days by segment) by mult and runs each scaled vector to equilibrium. The scaled days are the caps (registered days); the days fished are those of the vessels that break even, so the curve ends where effort is limited by profit.

Usage

```
ms_yield_curve(
  sys,
  E_ref,
  mult = c(0, 10^seq(-1.5, log10(8), length.out = 60)),
  years = 300,
  Bpa = 0.5,
  ...
)
```

Arguments

sys	an ms_system.
E_ref	reference days by segment (e.g. the status quo).
mult	multipliers of E_ref.
years	years to equilibrium.
Bpa	safeguard relative to Bmsy.
...	passed to ms_simulate() (gamma, open, ...).

Value

A data frame with E (days fished), yield, profit, min_BBmsy and n_below_Bpa, with attribute BBmsy.

---

perf_rows *Performance statistics of projection runs*

Description

perf_rows() summarises rows (simulations) of one run over years; ms_performance() does this for a named list of runs.

Usage

```
perf_rows(
  x,
  rows = NULL,
  years = NULL,
  Bpa = 0.5,
  Blim = 0.25,
  assessed = NULL
)

ms_performance(runs, years = NULL, Bpa = 0.5, Blim = 0.25)
```

Arguments

x	an ms_sim run with keep = "all".
rows	simulation rows (default all).
years	years to summarise (default the last 20).
Bpa, Blim	reference points relative to Bmsy.
assessed	stocks for worst_P_assessed.
runs	named list of ms_sim runs.

Value

A one-row data frame of performance statistics.

plot_bps *Boxplots of performance statistics by management procedure*

Description

The style of `mseviz::plotBPs()`: one panel per statistic, a box per management procedure (10th, 25th, 50th, 75th and 90th percentiles over iterations) with a point at the median; probabilities are shown as a point at their mean.

Usage

```
plot_bps(
  perf,
  statistics = NULL,
  biol = "All",
  target = c(BBmsy = 0.8, FFmsy = 1, CMSY = 1, PB80 = 0.5),
  limit = c(PBlim = 0.05, PanyBlim = 0.05),
  ncol = 3
)
```

Arguments

`perf`	output of `ms_perf()`.
`statistics`	statistics to include (default all in `perf` for `biol`).
`biol`	a single unit (`"All"`, a segment or a stock code) or several stock codes (one row of panels per stock).
`target`, `limit`	named vectors of reference lines by statistic (green and red dashed lines), drawn for the statistics shown. Defaults: targets B/Bmsy = 0.8 (the FAO threshold for overfished stocks), F/Fmsy = 1, Catch/MSY = 1 and P(B > 0.8 Bmsy) = 0.5; limits P(B < Blim) = 0.05 and P(any B < Blim) = 0.05. `NULL` draws none.
`ncol`	columns of panels (single unit).

Value

A ggplot.

plot_catchability *Catchability heatmap*

Description

Catchability of each stock in each métier, relative to the stock's highest catchability.

Usage

```
plot_catchability(sys)
```

Arguments

sys	an ms_system.

Value

A ggplot.

---

plot_fleet *Projections of days by segment and total yield*

Description

Projections of days by segment and total yield

Usage

```
plot_fleet(runs)
```

Arguments

runs	named list of ms_sim runs with keep = "all".

Value

A ggplot.

---

plot_kobe *Kobe plot of equilibrium states*

Description

One point per stock, one panel per scenario. With zones = "hcr" the background follows the WKREF/GFCM convention of FLRef::plotWKREF(), with zones cut by the harvest control rule: below Bthresh, red above the HCR line and yellow below it; above Bthresh, orange above and green below; a dark overlay below Blim. The HCR line is Fmin up to Bclose, a linear ramp to Ftgt at Btrigger, and Ftgt above (as FLRef::plotGFCM(fmin = 0.1, bclose = 0.25, btrigger = 0.8)).

Usage

```
plot_kobe(
  states,
  sys,
  Bpa = 0.5,
  Blim = 0.25,
  Btrigger = 0.8,
  Ftgt = 1,
  Bclose = Blim,
  Fmin = 0.1,
  zones = c("hcr", "simple"),
  palette = c("pastel", "flref"),
  alpha = 1
)
```

Arguments

states	named list of states with BBmsy and FFmsy (e.g. from ms_status_quo() or ms_optimise()).
sys	an ms_system.
Bpa, Blim	reference points relative to Bmsy (Bpa is Bthresh).
Btrigger, Ftgt, Bclose, Fmin	HCR trigger, target F (relative to Fmsy), biomass at and below which F is at its minimum (default Blim, as in hcr_effort()) and that minimum F (default 0.1, the minimum effort Emin of hcr_effort()).
zones	"hcr" (coloured zones and the HCR line) or "simple".
palette	zone colours: "pastel" (default) or "flref" (the saturated colours of FLRef::plotWKREF()).
alpha	opacity of the zones.

Value

A ggplot.

Examples

```
sys <- atlantic_system()

sq <- ms_status_quo(sys, capacity = 1000, gamma = 1.5, years = 200)
plot_kobe(list("Status quo" = sq), sys)
```

plot_landscape *Plot a habitat landscape*

Description

Each stock's distribution over depth (summed over substrates).

Usage

```
plot_landscape(Q, cells, ncol = 5)
```

Arguments

Q	a catchability matrix from ms_landscape().
cells	the cells passed to ms_landscape().
ncol	columns of panels.

Value

A ggplot.

plot_projection *Projections by stock: B/Bmsy, F/Fmsy or catch over time*

Description

Median and 90% band by stock, for a named list of runs.

Usage

```
plot_projection(
  runs,
  what = c("BBmsy", "FFmsy", "yield"),
  Bpa = 0.5,
  Blim = 0.25,
  ncol = 5
)
```

Arguments

runs	named list of ms_sim runs with keep = "all".
what	"BBmsy", "FFmsy" or "yield".
Bpa, Blim	reference points relative to Bmsy.
ncol	columns of the panel grid.

Value

A ggplot.

`plot_quilt` *Quilt plot: a colour-coded decision table*

Description

A table of medians (means for probabilities) with each cell coloured by how well the management procedure does on that statistic relative to the others: green = best, red = worst within a column (or, with `layout = "stocks"`, within a stock), using the direction in `ms_statistics()`; statistics without a preferred direction (days) are grey. With `risk`, probabilities of being below a limit are coloured by absolute classes instead.

Usage

```
plot_quilt(
  perf,
  statistics = NULL,
  biol = "All",
  layout = c("statistics", "stocks"),
  statistic = "BBmsy",
  risk = NULL,
  digits = 2,
  palette = c("#e8857c", "#f6d38a", "#a9d99a")
)
```

Arguments

`perf`	output of `ms_perf()`.
`statistics`	statistics to include (default all in `perf` for `biol`).
`biol`	unit: `"All"` (system), a segment or a stock code; several values give one block per unit.
`layout`	`"statistics"` (rows = management procedures, columns = statistics, for one unit) or `"stocks"` (rows = stocks, columns = management procedures, for one statistic).
`statistic`	the statistic for `layout = "stocks"` (e.g. `"BBmsy"`, `"PBpa"`).
`risk`	optional risk limit for probabilities of being below a limit (`PBlim`, `PBpa`, `PanyBlim`), e.g. 0.05: cells at or below the limit are green, up to twice the limit amber, above red, regardless of rank.
`digits`	significant digits.
`palette`	colours from worst to best.

Value

A ggplot.

plot_tradeoffs *Trade-off plots of performance statistics*

Description

The style of `mseviz::plotTOs()`: medians (means for probabilities) of each statistic in `y` against statistic `x`, with 10th-90th percentile ranges, one point per management procedure.

Usage

```
plot_tradeoffs(
  perf,
  x = "C",
  y = c("NBpa", "PanyBlim", "Profit", "AAVC"),
  biol_x = "All",
  biol_y = "All",
  ncol = 2
)
```

Arguments

`perf`	output of `ms_perf()`.
`x`	statistic on the x axis (default total catch).
`y`	statistics on the y axes.
`biol_x, biol_y`	units of `x` and `y` (default the system, `"All"`).
`ncol`	columns of panels (single unit).

Value

A ggplot.

plot_yield_curves *Equilibrium yield curves*

Description

Equilibrium yield curves

Usage

```
plot_yield_curves(curves, ref = NULL, sys = NULL, Bpa = 0.5)
```

Arguments

curves	named list of curves from ms_yield_curve() or ms_shared_q_curve().
ref	optional reference state (e.g. the status quo) to mark.
sys	optional ms_system, to draw the sum of single-stock MSY.
Bpa	safeguard relative to Bmsy.

Value

A ggplot (yield over days; filled points: MMSY, open points: highest yield with every stock >= Bpa).

---

proc_sd_from_r *Process SD by productivity: a power law in r through two anchors*

Description

sd(r) = sd1 * (r / r1)^p with p = log(sd2 / sd1) / log(r2 / r1). Default anchors: sharks (r = 0.057, sd = 0.05) and anchovy (r = 1.2, sd = 0.4), giving sd = 0.355 * r^0.68.

Usage

```
proc_sd_from_r(r, r1 = 0.057, sd1 = 0.05, r2 = 1.2, sd2 = 0.4)
```

Arguments

r	intrinsic growth rates.
r1, sd1, r2, sd2	the two anchors.

Value

Process SD (log scale) for each r.

---

read_ms_system *Read a system from CSV tables*

Description

Reads <prefix>_species.csv, <prefix>_metiers.csv, <prefix>_catchability.csv and <prefix>_segments.csv from a folder and passes them to ms_system().

Usage

```
read_ms_system(dir, prefix, ...)
```

Arguments

dir	folder with the tables.
prefix	file prefix.
...	passed to ms_system().

---

relsig_jabba *JABBA* proc.dyn *relative process SD*

Description

relsig(x) = raw(x) / raw(1), raw(x) = a + (1 - a) / (1 + (x / c)^d), with x = B/Bmsy (empirical ICES a = 0.128, c = 0.223, d = 1.326), as in JABBA's jabba2jags.R. relsig = 1 at Bmsy.

Usage

```
relsig_jabba(BBmsy, a = 0.128, c = 0.223, d = 1.326)
```

Arguments

BBmsy	B/Bmsy.
a, c, d	parameters.

Value

relsig for each B/Bmsy.

---

run_app *Launch the mmsyFM Shiny app*

Description

Opens the app with six tabs: set-up (system, status quo, tuning), equilibrium explorer, yield curves, safeguarded MMSY, harvest strategies and trigger tuning.

Usage

```
run_app(...)
```

Arguments

...	passed to shiny::runApp().

Value

Called for its side effect.

theme_ms *Colours and theme of the package plots*

Description

`ms_cols` are the categorical colours, `ms_ink` the text and line colours, and `theme_ms()` the ggplot2 theme.

Usage

```
theme_ms(base_size = 10)

ms_cols

ms_ink
```

Arguments

`base_size`	base font size.

write_ms_system *Write a system to CSV tables*

Description

The inverse of `read_ms_system()`: writes the four input tables.

Usage

```
write_ms_system(sys, dir, prefix)
```

Arguments

`sys`	an `ms_system`.
`dir`	folder with the tables.
`prefix`	file prefix.

Index

atlantic_system, 3
atlantic_system(), *21*

ffmsy, 3
ffmsy(), *10*, *19*

got_species, 4
got_system_draft, 5

hcr_effort, 6
hcr_effort(), *8*, *23*, *24*, *28*

ms_assessment_presets, 8
ms_closure, 9
ms_cols *(*theme_ms*)*, 34
ms_condition, 9
ms_condition(), *5*, *11*, *21*
ms_ink *(*theme_ms*)*, 34
ms_landscape, 11
ms_landscape(), *29*
ms_open_access, 12
ms_open_access(), *16*
ms_optimise, 13
ms_optimise(), *7*, *28*
ms_perf, 14
ms_perf(), *15*, *26*, *30*, *31*
ms_perf_table, 15
ms_perf_table(), *15*
ms_performance *(*perf_rows*)*, 25
ms_shared_q_curve, 16
ms_shared_q_curve(), *32*
ms_simulate, 16
ms_simulate(), *8–10*, *12*, *14*, *16*, *20*, *21*, *24*
ms_statistics, 19
ms_statistics(), *30*
ms_status_quo *(*ms_open_access*)*, 12
ms_status_quo(), *16*, *28*
ms_summary, 20
ms_system, 20
ms_system(), *32*, *33*
ms_tune, 21
ms_tune(), *3*, *9*, *10*, *21*
ms_tune_hcr, 23
ms_tune_hcr(), *6*
ms_yield_curve, 24
ms_yield_curve(), *16*, *32*

perf_rows, 25
plot_bps, 26
plot_bps(), *14*, *15*
plot_catchability, 26
plot_fleet, 27
plot_kobe, 27
plot_landscape, 29
plot_landscape(), *12*
plot_projection, 29
plot_quilt, 30
plot_quilt(), *14*, *15*
plot_tradeoffs, 31
plot_tradeoffs(), *14*, *15*
plot_yield_curves, 31
proc_sd_from_r, 32

read_ms_system, 32
read_ms_system(), *34*
relsig_jabba, 33
run_app, 33

shiny::runApp(), *33*

theme_ms, 34

write_ms_system, 34