Package: SPoRC (via r-universe)

June 4, 2026

Type Package

Title A Generalized Stochastic (S) Population (Po) Model Over Regional (R) Components (C)

Version 1.1.1

Description A generalized spatial population model for stock assessment, written in RTMB.

License GPL-3

URL https://chengmatt.github.io/SPoRC

Depends R (>= 3.5), RTMB

Imports abind, cowplot, data.table, dplyr, future, future.apply, ggplot2, grid, gridExtra, here, MASS, Matrix, methods, nleqslv, parallel, progressr, randtests, reshape2, stats, stringr, tidyr, TMB

Suggests knitr, purrr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev

Repository https://noaa-fisheries-integrated-toolbox.r-universe.dev

Date/Publication 2026-06-04 14:40:55 UTC

RemoteUrl https://github.com/chengmatt/SPoRC

RemoteRef HEAD

RemoteSha abc329bcd07446b23a4b85aea563360fe89d43dd

Contents

`catch_to_F_multifleet` *Solve for fishing mortality rates that achieve target catches for multiple fleets*

Description

Solve for fishing mortality rates that achieve target catches for multiple fleets

Usage

```
catch_to_F_multifleet(
  target_catch,
  NAA,
  WAA,
  natmort,
  fish_sel,
  f_init = 0.05,
  control = list(btol = 1e-06)
)
```

Arguments

`target_catch`	Numeric vector of target catch values for each fleet
`NAA`	Matrix of numbers-at-age (ages x sexes)
`WAA`	Matrix of weight-at-age (ages x sexes)
`natmort`	Matrix of natural mortality (ages x sexes)
`fish_sel`	3D array of fishery selectivity (ages x sexes x fleets)
`f_init`	Initial guess for F values (scalar or vector)
`control`	List of control parameters for nleqslv

Value

Numeric vector of F values for each fleet

See Also

Other Closed Loop Simulations: `catch_to_F_singlefleet()`, `condition_closed_loop_simulations()`, `get_closed_loop_reference_points()`

`catch_to_F_singlefleet` *Go from TAC to Fishing Mortality using bisection for when a single fishery fleet exists*

Description

Go from TAC to Fishing Mortality using bisection for when a single fishery fleet exists

Usage

```
catch_to_F_singlefleet(
  f_guess,
  catch,
  NAA,
  WAA,
  natmort,
  fish_sel,
  n.iter = 20,
  lb = 0,
  ub = 2
)
```

Arguments

`f_guess`	Initial guess of F
`catch`	Provided catch values
`NAA`	Numbers, dimensioned by ages, and sexes
`WAA`	Weight, dimensioned by ages and sexes
`natmort`	Natural mortality dimensioned by ages and sex
`fish_sel`	Fishery selectivity, dimesnioned by ages and sex
`n.iter`	Number of iterations for bisection
`lb`	Lower bound of F
`ub`	Upper bound of F

Value

Fishing mortality values for a single fleet

See Also

Other Closed Loop Simulations: `catch_to_F_multifleet()`, `condition_closed_loop_simulations()`, `get_closed_loop_reference_points()`

`condition_closed_loop_simulations`

Set up simulation list for closed-loop projections

Description

This function creates and initializes a simulation list for closed-loop projections of population dynamics. All components of the simulation list must match the expected names used internally by the setup functions. Users can provide **custom definitions** for any component by passing them through ‘...‘ using the correct name (e.g., ‘WAA_input‘, ‘fish_sel_input‘).

Usage

```
condition_closed_loop_simulations(
  closed_loop_yrs,
  n_sims,
  data,
  parameters,
  mapping,
  sd_rep,
  rep,
  random = random,
  FishIdx_SE_fill = "mean",
  SrvIdx_SE_fill = "mean",
```

```
  ISS_FishAgeComps_fill = "mean",
  ISS_FishLenComps_fill = "mean",
  ISS_SrvAgeComps_fill = "mean",
  ISS_SrvLenComps_fill = "mean",
  ...
)
```

Arguments

`closed_loop_yrs`	Integer. Number of years to project in the closed loop.
`n_sims`	Integer. Number of simulation replicates.
`data`	List. Observed data and configuration for the population.
`parameters`	List. Parameter values for the model.
`mapping`	List. Mapping of parameters for optimization.
`sd_rep`	List. Standard deviation reports from fitted model.
`rep`	List. Report from fitted model.
`random`	Character vector of random effects estimated
`FishIdx_SE_fill`	Character or numeric. Fill method for fishery index standard errors when extending to simulation years. Options are: - ‘"zeros"‘: fill with zeros - ‘"last"‘: repeat last non-NA slice - ‘"mean"‘: fill with the mean of the observed series - Numeric: constant scalar or array value
`SrvIdx_SE_fill`	Character or numeric. Fill method for survey index standard errors. Same options as ‘FishIdx_SE_fill‘.
`ISS_FishAgeComps_fill`	Character or numeric. Fill method for fishery age composition input sample sizes. Options are: - ‘"zeros"‘, ‘"last"‘, ‘"mean"‘ (as above) - ‘"F_pattern"‘: fill based on fishing mortality pattern in the closed-loop simulation - Numeric: constant scalar or array value
`ISS_FishLenComps_fill`	Character or numeric. Fill method for fishery length composition input sample sizes. Same options as ‘ISS_FishAgeComps_fill‘.
`ISS_SrvAgeComps_fill`	Character or numeric. Fill method for survey age composition input sample sizes. Options are ‘"zeros"‘, ‘"last"‘, ‘"mean"‘, or a numeric constant.
`ISS_SrvLenComps_fill`	Character or numeric. Fill method for survey length composition input sample sizes. Same options as ‘ISS_SrvAgeComps_fill‘.
`...`	Optional named arguments for custom inputs. Each name must correspond to a component expected by the simulation setup functions, and be dimensioned appropriately: ‘Setup_Sim_Fishing()‘, ‘Setup_Sim_Survey()‘, ‘Setup_Sim_Biologicals()‘, ‘Setup_Sim_Rec()‘, and ‘Setup_Sim_Tagging()‘. Examples include: - **Fishing**: ‘fish_sel_input‘, ‘ln_sigmaC‘, ‘Fmort_input‘, ‘fish_q_input‘, etc. - **Survey**: ‘srv_sel_input‘, ‘srv_q_input‘, ‘ObsSrvIdx_SE‘, etc. -

Biologicals: ‘WAA_input‘, ‘MatAA_input‘, ‘natmort_input‘, ‘AgeingError_input‘, ‘SizeAgeTrans_input‘ - **Recruitment inputs**: - ‘R0_input‘, ‘h_input‘, ‘sexratio_input‘, ‘ln_InitDevs_input‘, ‘Rec_input‘ - ‘Rec_input‘: - If shorter than the number of projection years, new recruitment deviates will be simulated based on ‘recruitment_opt‘, ‘R0_input‘, and ‘h_input‘ (supports changing regimes across years). - If you want fixed recruitment for all projection years, provide a ‘Rec_input‘ array that spans all years. - **Tagging**: ‘Tag_Reporting_input‘, ‘ln_Init_Tag_Mort‘, ‘ln_Tag_Shed‘, ‘tag_selex‘, ‘tag_natmort‘, ‘n_tags‘, ‘n_tags_rel_input‘ - **Movement**: ‘Movement‘ (must match the expected dimensions and be named exactly ‘Movement‘)

The values must have the correct dimensions expected by each component. If a component is not provided, default behavior will extend the last year (or zeros for fishing mortality, which can filled in subsequently)

See Also

Other Closed Loop Simulations: `catch_to_F_multifleet()`, `catch_to_F_singlefleet()`, `get_closed_loop_referenc`

`do_francis_reweighting`

Get Francis Weights

Description

Get Francis Weights

Usage

```
do_francis_reweighting(data, rep, age_labels, len_labels, year_labels)
```

Arguments

`data`	List of data inputs
`rep`	Report file list
`age_labels`	Age labels
`len_labels`	Length labels
`year_labels`	Year labels

Details

Function to get francis weights. Used inside the wrapper function run_francis(), or can be defined by the user as a loop to extract Francis weights (see example).

Value

A list object of francis weights (note that it will be NAs for some, if using jnt composition approaches - i.e., only uses one dimension), as well as a dataframe of francis mean fits

See Also

Other Francis Reweighting: `run_francis()`

Examples

```
## Not run:
for(j in 1:5) {

  if(j == 1) { # reset weights at 1
    data$Wt_FishAgeComps[] <- 1
    data$Wt_FishLenComps[] <- 1
    data$Wt_SrvAgeComps[] <- 1
    data$Wt_SrvLenComps[] <- 1
  } else {
    data$Wt_FishAgeComps[] <- wts$new_fish_age_wts
    data$Wt_FishLenComps[] <- wts$new_fish_len_wts
    data$Wt_SrvAgeComps[] <- wts$new_srv_age_wts
    data$Wt_SrvLenComps[] <- wts$new_srv_len_wts
  }

  sabie_rtmb_model <- fit_model(data,
                               parameters,
                               mapping,
                               random = NULL,
                               newton_loops = 3,
                               silent = TRUE
  )

  rep <- sabie_rtmb_model$report(sabie_rtmb_model$env$last.par.best) # Get report
  wts <- do_francis_reweighting(data = data, rep = rep, age_labels = 2:31,
                               len_labels = seq(41, 99, 2), year_labels = 1960:2024)
}

## End(Not run)
```

`do_jitter` *Run Jitter Analysis*

Description

Run Jitter Analysis

Usage

```
do_jitter(
  data,
  parameters,
  mapping,
  random = NULL,
```

```
  sd,
  n_jitter,
  n_newton_loops,
  do_par,
  n_cores,
  par_vec = NULL
)
```

Arguments

`data`	Data list to make obj
`parameters`	Parameter list to make obj
`mapping`	Mapping list to make obj
`random`	Character of random effects
`sd`	sd for jitter (additive)
`n_jitter`	Number of jitters to do
`n_newton_loops`	Number of newton loops to do
`do_par`	Whether to do paralleizaiton or not (boolean)
`n_cores`	Number of cores to use
`par_vec`	Vector of parameter starting values to use for jitter analysis. The default of this is NULL (jitters the starting value of the model). If a vector is provided, the jitter is initialized at the MLE parameters

Value

Dataframe of jitter values

See Also

Other Model Diagnostics: `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
   library(ggplot2)
   # get jitter values
   jit <- do_jitter(data = data,
                 parameters = parameters,
                 mapping = mapping,
                 random = NULL,
                 sd = 0.1,
                 n_jitter = 100,
                 n_newton_loops = 3,
                 do_par = TRUE,
                 n_cores = 8)
```

```
 # get proportion converged
 prop_converged <- jit %>%
 filter(Year == 1, Type == 'Recruitment') %>%
   summarize(prop_conv = sum(Hessian) / length(Hessian))

 # get final model results
 final_mod <- reshape2::melt(sabie_rtmb_model$rep$SSB) %>%
 rename(Region = Var1, Year = Var2) %>%
 mutate(Type = 'SSB') %>%
   bind_rows(reshape2::melt(sabie_rtmb_model$rep$Rec) %>%
   rename(Region = Var1, Year = Var2) %>% mutate(Type = 'Recruitment'))

 # comparison of SSB and recruitment
ggplot() +
  geom_line(jit, mapping = aes(x = Year + 1959, y = value,
   group = jitter, color = Hessian), lwd = 1) +
  geom_line(final_mod, mapping = aes(x = Year + 1959, y = value),
  color = "black", lwd = 1.3 , lty = 2) +
  facet_grid(Type~Region, scales = 'free',
             labeller = labeller(Region = function(x) paste0("Region ", x),
  Type = c("Recruitment" = "Age 2 Recruitment (millions)",
   "SSB" = 'SSB (kt)'))) +
  labs(x = "Year", y = "Value") +
  theme_bw(base_size = 20) +
  scale_color_manual(values = c("red", 'grey')) +
  geom_text(data = jit %>% filter(Type == 'SSB', Year == 1, jitter == 1),
            aes(x = Inf, y = Inf, label =
            paste("Proportion Converged: ",
            round(prop_converged$prop_conv, 3))),
            hjust = 1.1, vjust = 1.9, size = 6, color = "black")

 # compare jitter of max gradient and hessian PD
 ggplot(jit, aes(x = jitter, y = jnLL,
 color = Max_Gradient, shape = Hessian)) +
   geom_point(size = 5, alpha = 0.3) +
   geom_hline(yintercept = min(sabie_rtmb_model$rep$jnLL),
   lty = 2, size = 2, color = "blue") +
   facet_wrap(~Hessian, labeller = labeller(
     Hessian = c("FALSE" = "non-PD Hessian", "TRUE" = 'PD Hessian')
   )) +
   scale_color_viridis_c() +
   theme_bw(base_size = 20) +
   theme(legend.position = "bottom") +
   guides(color = guide_colorbar(barwidth = 15, barheight = 0.5)) +
   labs(x = 'Jitter') +
   geom_text(data = jit %>% filter(Hessian == TRUE,
   Year == 1, jitter == 1),
             aes(x = Inf, y = Inf, label =
             paste("Proportion Converged: ",
             round(prop_converged$prop_conv, 3))),
             hjust = 1.1, vjust = 1.9, size = 6, color = "black")
```

```
## End(Not run)
```

`do_likelihood_profile` *Run Likelihood Profile*

Description

Run Likelihood Profile

Usage

```
do_likelihood_profile(
  data,
  parameters,
  mapping,
  random = NULL,
  what,
  idx = NULL,
  min_val,
  max_val,
  inc = 0.05,
  do_par = FALSE,
  n_cores = NULL
)
```

Arguments

`data`	data list from model
`parameters`	parameter list from model
`mapping`	mapping list from model
`random`	character vector of random effects to estimate
`what`	parameter name we want to profile
`idx`	Index for an parameter array, pointing to the value we want to map off (index is relative to a flattened array)
`min_val`	minimum value of profile
`max_val`	maximum value of profile
`inc`	increment value between min and max value
`do_par`	logical, whether to use parallel processing (default FALSE)
`n_cores`	integer, number of cores to use for parallel processing (default is detectCores() - 1)

Value

Returns a list of likelihood profiled values for each data component with their respective dimensions (e.g., likelihood profiles by fleet, region, year, etc.) as well likelihood profiles for each data component, aggregated across all their respective dimensions.

See Also

Other Model Diagnostics: `do_jitter()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Do_Population_Projection

Do Population Projections

Description

Do Population Projections

Usage

```
Do_Population_Projection(
  n_proj_yrs = 2,
  n_regions,
  n_ages,
  n_sexes,
  sexratio,
  n_fish_fleets,
  do_recruits_move = 0,
  recruitment,
  terminal_NAA,
  terminal_NAA0,
  terminal_F,
  natmort,
  WAA,
  WAA_fish,
  MatAA,
  fish_sel,
  Movement,
  f_ref_pt = NULL,
  b_ref_pt = NULL,
  HCR_function = NULL,
  recruitment_opt = "inv_gauss",
  fmort_opt = "HCR",
  t_spawn,
  bh_rec_opt = NULL,
  init_F = 0
)
```

Arguments

`n_proj_yrs`	Number of projection years

`n_regions`	Number of regions
`n_ages`	Number of ages
`n_sexes`	Number of sexes
`sexratio`	Array of recruitment sex ratio (n_regions, n_proj_yrs, n_sexes)
`n_fish_fleets`	Number of fishery fleets
`do_recruits_move`	Whether recruits move (0 == don't move, 1 == move)
`recruitment`	Recruitment matrix dimensioned by n_regions, and n_yrs that we want to summarize across, or condition our projection on
`terminal_NAA`	Terminal Numbers at Age dimensioned by n_regions, n_ages, n_sexes
`terminal_NAA0`	Terminal Unfished Numbers at Age dimensioned by n_regions, n_ages, n_sexes
`terminal_F`	Terminal fishing mortality rate, dimensioned by n_regions, n_fish_fleets
`natmort`	Natural mortality, dimensioned by n_regions, n_proj_yrs, n_ages, n_sexes
`WAA`	Weight at age, dimensioned by n_regions, n_proj_yrs, n_ages, n_sexes
`WAA_fish`	Weight at age for the fishery, dimensioned by n_regions, n_proj_yrs, n_ages, n_sexes, n_fish_fleets
`MatAA`	Maturity at age, dimensioned by n_regions, n_proj_yrs, n_ages, n_sexes
`fish_sel`	Fishery selectivity, dimensioned by n_regions, n_proj_yrs, n_ages, n_sexes, n_fish_fleets
`Movement`	Movement, dimensioned by n_regions, n_regions, n_proj_yrs, n_ages, n_sexes
`f_ref_pt`	Fishing mortality reference point dimensioned by n_regions and n_proj_yrs
`b_ref_pt`	Biological reference point dimensioned by n_regions and n_proj_yrs
`HCR_function`	Function describing a harvest control rule. The function should always have the following arguments: x, which represents SSB, frp, which takes inputs of fishery reference points, and brp, which takes inputs of biological reference points. Any additional arguments should be specified with defaults or hard coded / fixed within the function.
`recruitment_opt`	Recruitment simulation option, where options are "inv_gauss", which simulates future recruitment based on the the recruitment values supplied using an inverse gaussian distribution, "mean_rec", which takes the mean of the recruitment values supplied for a given region, and "zero", which assumes that future recruitment does not occur
`fmort_opt`	Fishing mortality option. Choices are: * **"HCR"** – Applies the user-supplied ‘HCR_function‘ using region-specific SSB, F reference point, and biomass reference point. * **"HCR_global"** – Applies the ‘HCR_function‘ using global SSB (summed across regions) and a global biomass reference point (sum of the region-specific biomass reference points). Each region's biomass reference point should be defined individually; the function performs the summation. * **"Input"** – Uses user-supplied projected fishing mortality values directly.
`t_spawn`	Fraction time of spawning used to compute projected SSB

`bh_rec_opt`	A list object containing the following arguments: **recruitment_dd** A value (0 or 1) indicating global (1) or local density dependence (0). In the case of a single region model, either local or global will give the same results **rec_lag** A value indicating the number of years lagged that a given year's SSB produces recruits **R0** The virgin recruitment parameter **Rec_Prop** Recruitment apportionment values. In a single region model, this should be set at a value of 1. Dimensioned by n_regions **h** Steepness values for the stock recruitment curve. Dimensioned by n_regions **WAA** A weight-at-age array dimensioned by n_regions, n_ages, and n_sexes, where the reference year should utilize values from the first year **MatAA** A maturity at age array dimensioned by n_regions, n_ages, and n_sexes, where the reference year should utilize values from the first year **natmort** A natural mortality at age array dimensioned by n_regions, n_ages, and n_sexes, where the reference year should utilize values from the first year **SSB** All SSB values estimated from a given model, dimensioned by n_regions and n_yrs
`init_F`	Scalar of initial F value to apply for deriving beverton holt recruitment; default is set at 0.

Value

A list containing projected F, catch, SSB (and dynamic unfished), and Numbers at Age (and dynamic unfished). (Objects are generally dimensioned in the following order: n_regions, n_yrs, n_ages, n_sexes, n_fleets)

See Also

Other Reference Points and Projections: `Get_Reference_Points()`, `get_key_quants()`

`do_retrospective` *Run retrospective analyses for RTMB models*

Description

Performs retrospective peels by truncating the input data, optionally applying Francis reweighting and parallelization, and returns estimates of spawning stock biomass (SSB) and recruitment for each peel.

Usage

```
do_retrospective(
  n_retro,
  data,
  parameters,
  mapping,
  random = NULL,
  do_par,
  n_cores,
  newton_loops = 3,
  do_francis = FALSE,
  n_francis_iter = NULL,
  nlminb_control = list(iter.max = 1e+05, eval.max = 1e+05, rel.tol = 1e-15),
  do_sdrep = FALSE,
  fishidx_datalag = array(0, dim = c(data$n_regions, data$n_fish_fleets)),
  fishage_datalag = array(0, dim = c(data$n_regions, data$n_fish_fleets)),
  fishlen_datalag = array(0, dim = c(data$n_regions, data$n_fish_fleets)),
  srvidx_datalag = array(0, dim = c(data$n_regions, data$n_srv_fleets)),
  srvage_datalag = array(0, dim = c(data$n_regions, data$n_srv_fleets)),
  srvlen_datalag = array(0, dim = c(data$n_regions, data$n_srv_fleets)),
  tag_datalag = 0
)
```

Arguments

`n_retro`	Integer. Number of retrospective peels to perform.
`data`	List. Data input for the RTMB model.
`parameters`	List. Parameter values for the RTMB model.
`mapping`	List. Mapping information for the RTMB model.
`random`	Character vector. Names of random effects in the model. Default is `NULL`.
`do_par`	Logical. Whether to run retrospective peels in parallel. Default is `FALSE`.
`n_cores`	Integer. Number of cores to use for parallel execution if `do_par = TRUE`.
`newton_loops`	Integer. Number of Newton loops to run during model fitting. Default is 3.
`do_francis`	Logical. Whether to apply Francis reweighting within each retrospective peel. Default is `FALSE`.
`n_francis_iter`	Integer. Number of Francis reweighting iterations. Required if `do_francis = TRUE`.
`nlminb_control`	List. Control parameters passed to `nlminb` during model fitting. Default is `list(iter.max = 1e5, eval.max = 1e5, rel.tol = 1e-15)`.
`do_sdrep`	Logical. Whether to return standard errors from `sdreport`. Default is `FALSE`.
`fishidx_datalag`	Integer array. Lags for fishery index data [regions x fleets]. Default is zeros.
`fishage_datalag`	Integer array. Lags for fishery age composition data [regions x fleets]. Default is zeros.

`fishlen_datalag`	Integer array. Lags for fishery length composition data [regions x fleets]. Default is zeros.
`srvidx_datalag`	Integer array. Lags for survey index data [regions x fleets]. Default is zeros.
`srvage_datalag`	Integer array. Lags for survey age composition data [regions x fleets]. Default is zeros.
`srvlen_datalag`	Integer array. Lags for survey length composition data [regions x fleets]. Default is zeros.
`tag_datalag`	Integer. Lag for tagging data. Default is 0.

Value

A `data.frame` containing retrospective estimates of SSB and recruitment. Columns include:

- `Region`: Region index.
- `Year`: Year index.
- `Type`: "SSB" or "Recruitment".
- `peel`: Peel number (0 = full data, 1 = 1-year peel, etc.).
- `value`: Estimated value of SSB or recruitment.
- `pdHess` and `max_grad` (optional): Information from `sdreport` if `do_sdrep` = `TRUE`.

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

`do_runs_test` *Runs test function taken from SS3 diags.*

Description

Runs test function taken from SS3 diags.

Usage

```
do_runs_test(x, type = NULL, mixing = "two.sided")
```

Arguments

`x`	Vector of residuals
`type`	Whether to use mean 0 assumption of mean of residuals (default = use mean 0)
`mixing`	Type of test to do, less = left tailed test that detects positive autocorrelation, two.sided = two sided test that tests whether there is positive and/or negative autocorrealtion. The null is that there isn't any, rejecting the null (<0.05) indictes that there is some non-randomness.

Value

List object with p value and limits for a three-sigma limit - (potential data outlier, where residual is > 3 standard deviations away from a mean of 0)

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
 idx_fits <- get_idx_fits(data = data, rep = rep,
 year_labs = seq(1960, 2024, 1))
  idx_fits <- idx_fits %>%
    mutate(
      Idx = case_when(
        Type == "Fishery" & Year < 1995 ~
         "Japanese Fishery CPUE Index",
        Type == "Fishery" & Year >= 1995 ~
         "Domestic Fishery CPUE Index",
        Type == 'Survey' & Fleet == 1 ~
        "Domestic LL Survey Relative Population Numbers",
        Type == 'Survey' & Fleet == 2 ~
        "GOA Trawl Survey Biomass (kt)",
        Type == 'Survey' & Fleet == 3 ~
        'Japanese LL Survey Relative Population Numbers'
      )
    )

  unique_idx <- unique(idx_fits$Idx)
  runs_all <- data.frame()
  for(i in 1:length(unique(idx_fits$Idx))) {
    tmp <- idx_fits %>% filter(Idx == unique_idx[i])
    runstest <- do_runs_test(x=as.numeric(tmp$resid),
    type="resid", mixing = "less")
    tmp_runs <- data.frame(p = runstest$p.runs,
     lwr = runstest$sig3lim[1], upr = runstest$sig3lim[2],
     Idx = unique_idx[i])
    runs_all <- rbind(runs_all, tmp_runs)
  } # end i

  ggplot() +
    geom_point(idx_fits, mapping = aes(x = Year, y = resid)) +
    geom_segment(idx_fits, mapping =
    aes(x = Year, xend = Year, y = 0, yend = resid)) +
    geom_smooth(idx_fits, mapping = aes(x = Year, y = resid), se = F) +
    geom_hline(yintercept = 0, lty = 2) +
    geom_hline(runs_all, mapping = aes(yintercept = upr), lty = 2) +
    geom_hline(runs_all, mapping = aes(yintercept = lwr), lty = 2) +
    geom_text(data = runs_all, aes(x = -Inf, y = Inf,
```

```
    label = paste("p = ", round(p, 3))), hjust = -0.5,
    vjust = 8.2, size = 7)+
    labs(x = "Year", y = 'Residuals') +
    theme_bw(base_size = 20) +
    facet_wrap(~Idx, scales = 'free', ncol = 2)

## End(Not run)
```

`dusky_rtmb_model` *Dusky model outputs from single regino model*

Description

A list containing inputs and outputs for the 2024 GOA Dusky Rockfish Assessment

Usage

```
dusky_rtmb_model
```

Format

Data list for single region dusky rockfish assessment

Source

Omori, K. L., Williams, B. C., Hulson, P.-J., Ferriss, B. 2024. Assessment of the dusky rockfish stock in the Gulf of Alaska. North Pacific Fishery Management Council, Anchorage, AK.

`fit_model` *Run RTMB model*

Description

Run RTMB model

Usage

```
fit_model(
  data,
  parameters,
  mapping,
  random = NULL,
  newton_loops = 3,
  silent = FALSE,
  do_optim = TRUE,
  nlminb_control = list(iter.max = 1e+05, eval.max = 1e+05, rel.tol = 1e-15),
  ...
)
```

Arguments

`data`	Data list
`parameters`	Parameter list
`mapping`	Mapping list
`random`	Character of random effects to integrate out
`newton_loops`	Number of newton loops to run to get gradients down
`silent`	Boolean on whether or not model run is silent
`do_optim`	Boolean on whether or not model is optimized
`nlminb_control`	List argument controls by nlminb
`...`	Additional arguments taken by MakeADFun

Value

Returns a list object that is optimized, with results outputted from the RTMB model

See Also

Other Utility: `get_logistN_Sigma()`, `get_par_est_info()`, `post_optim_sanity_checks()`, `rho_trans()`, `set_data_indicator_unused()`

Examples

```
## Not run:
model <- fit_model(data,
                   parameters,
                   mapping,
                   random = NULL,
                   newton_loops = 3)

## End(Not run)
```

`get_biological_plot` *Get Plots of Biological Quantities*

Description

Get Plots of Biological Quantities

Usage

```
get_biological_plot(data, rep, model_names)
```

Arguments

`data`	List of n_models of ‘SPoRC‘ data lists
`rep`	List of n_models of ‘SPoRC‘ report lists
`model_names`	Vector of model names

Value

A list of plots for terminal year movement, natural mortality, weight-at-age, and maturity at age across models

See Also

Other Plotting: `get_data_fitted_plot()`, `get_selex_plot()`, `get_ts_plot()`, `plot_all_basic()`, `theme_sablefish()`

Examples

```
## Not run:
get_biological_plot(list(data1, data2), list(rep1, rep2), c("Model1", "Model2"))

## End(Not run)
```

`get_catch_fits_plot` *Title Get Catch Fits Plot*

Description

Title Get Catch Fits Plot

Usage

```
get_catch_fits_plot(data, rep, model_names)
```

Arguments

`data`	List of n_models of ‘SPoRC‘ data lists
`rep`	List of n_models of ‘SPoRC‘ report lists
`model_names`	Vector of model names

Value

A plot of fitted values to various catch time series across models

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
get_catch_fits_plot(list(data1, data2), list(rep1, rep2), c("Model1", "Model2"))

## End(Not run)
```

`get_closed_loop_reference_points`

Get Closed Loop Reference Points

Description

Computes fishery and biological reference points either using "true" simulated values from the operating model or using assessment-derived data and report objects. Supports single-region and multi-region reference points.

Usage

```
get_closed_loop_reference_points(
  use_true_values,
  sim_env,
  asmt_data = NULL,
  asmt_rep = NULL,
  y,
  sim,
 reference_points_opt = list(n_avg_yrs = 1, SPR_x = 0.4, calc_rec_st_yr = 1, rec_age =
    1, type = "single_region", what = "SPR"),
  n_proj_yrs
)
```

Arguments

`use_true_values`
: Logical. If TRUE, uses values from the simulation environment (‘sim_env‘) for calculating reference points. If FALSE, uses ‘asmt_data‘ and ‘asmt_rep‘.

`sim_env`
: Simulation environment

`asmt_data`
: Optional list. Assessment data object (from RTMB) if not using true values.

`asmt_rep`
: Optional list. Assessment report object (from RTMB) if not using true values.

`y`
: Integer. Number of years to include in calculations (usually the last year of the assessment or simulation).

`sim`
: Integer. Index of the simulation replicate in ‘sim_env‘.

`reference_points_opt`
: List. Options for reference point calculations:

  n_avg_yrs Number of years to average over demographic rates. Default is 1.

  SPR_x Target SPR fraction for reference point calculations. Default is 0.4.

  calc_rec_st_yr Year to start calculating mean recruitment. Default is 1.

  rec_age Age at recruitment. Default is 1.

  type Reference point type: "single_region" or "multi_region". Default is "single_region".

	what Method for reference point calculation. Options include "SPR", "BH_MSY", "independent_SPR", "independent_BH_MSY", "global_SPR", "global_BH_MSY". Default is "SPR".
`n_proj_yrs`	Number of projection years

Value

A list with elements:

f_ref_pt Array of fishing reference points by region and projection year.

b_ref_pt Array of biological reference points by region and projection year.

virgin_b_ref_pt Array of unfished biological reference points by region and projection year.

See Also

Other Closed Loop Simulations: `catch_to_F_multifleet()`, `catch_to_F_singlefleet()`, `condition_closed_loop_si`

`get_comp_prop` *Gets composition data proportions normalized according to the assessment specifications from RTMB*

Description

Gets composition data proportions normalized according to the assessment specifications from RTMB

Usage

```
get_comp_prop(data, rep, age_labels, len_labels, year_labels)
```

Arguments

`data`	list of data inputs
`rep`	report file from RTMB
`age_labels`	vector of observed age labels in assessment
`len_labels`	vector of length labels in assessment
`year_labels`	vector of years

Value

List of fishery age, lengths, survey age, lengths dataframe as well as in matrix form (dimensioned by region, year, bin, sex, fleet)

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
comp_props <- get_comp_prop(data = data, rep = rep,
age_labels = 2:31, len_labels = seq(41, 99, 2), year_labels = 1960:2024)
comp_props$Fishery_Ages %>%
  filter(Fleet == 1, Sex == 1) %>%
  ggplot() +
  geom_col(aes(x = Age, y = obs)) +
  geom_line(aes(x = Age, y = pred)) +
  facet_wrap(~Year, ncol = 3)

  comp_props$Survey_Ages %>%
    group_by(Region, Age, Sex, Fleet) %>%
    summarize(lwr_obs = quantile(obs, 0.1),
              upr_obs = quantile(obs, 0.9),
              lwr_pred = quantile(pred, 0.1),
              upr_pred = quantile(pred, 0.9),
              obs = mean(obs),
              pred = mean(pred)) %>%
    ggplot() +
    geom_line(mapping = aes(x = Age, y = obs,
    color = 'Obs', lty = 'Obs'), lwd = 1.3) +
    geom_ribbon(mapping = aes(x = Age, y = obs,
    ymin = lwr_obs, ymax = upr_obs, fill = 'Obs'), alpha = 0.3) +
    geom_line(mapping = aes(x = Age, y = pred,
     color = 'Pred', lty = 'Pred'), lwd = 1.3) +
    geom_ribbon(mapping = aes(x = Age, y = pred,
     ymin = lwr_pred, ymax = upr_pred, fill = 'Pred'), alpha = 0.3) +
    facet_grid(Region~Fleet, labeller = labeller(
      Region = c('1' = ”Region 1”),
      Fleet = c('1' = 'Domestic LL Survey', '3' = 'JP LL Survey')
    )) +
    labs(x = 'Age', y = 'Proportion', color = '', linetype = '', fill = '') +
    theme_bw(base_size = 20) +
    theme(legend.position = 'top')

## End(Not run)
```

`get_data_fitted_plot` *Get Data Fitted to Plot*

Description

Get Data Fitted to Plot

Usage

```
get_data_fitted_plot(data, model_names)
```

Arguments

`data`	List of n_models of ‘SPoRC‘ data lists
`model_names`	Character vector of model names

Value

A plot of data that were fitted to across models

See Also

Other Plotting: `get_biological_plot()`, `get_selex_plot()`, `get_ts_plot()`, `plot_all_basic()`, `theme_sablefish()`

Examples

```
## Not run:
get_data_fitted_plot(list(data1, data2), c("Model1", "Model2"))

## End(Not run)
```

`get_idx_fits` *Gets index fits results*

Description

Gets index fits results

Usage

```
get_idx_fits(data, rep, year_labs)
```

Arguments

`data`	Data list fed into RTMB
`rep`	Report list output from RTMB
`year_labs`	Year labels to use (vector)

Value

Fits to indices as a dataframe

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
idx_fits <- get_idx_fits(data = data, rep = rep,
 year_labs = seq(1960, 2024, 1))

idx_fits <- idx_fits %>%
  mutate(
    Idx = case_when(
      Type == "Fishery" & Year < 1995 ~
      "Japanese Fishery CPUE Index",
      Type == "Fishery" & Year >= 1995 ~
      "Domestic Fishery CPUE Index",
      Type == 'Survey' & Fleet == 1 ~
      "Domestic LL Survey Relative Population Numbers",
      Type == 'Survey' & Fleet == 2 ~
      "GOA Trawl Survey Biomass (kt)",
      Type == 'Survey' & Fleet == 3 ~
       'Japanese LL Survey Relative Population Numbers'
    )
  )
ggplot() +
  geom_line(idx_fits, mapping =
  aes(x = Year, y = value), lwd = 1.3, col = 'red') +
  geom_pointrange(idx_fits, mapping =
  aes(x = Year, y = obs, ymin = lci, ymax = uci), color = 'blue', pch = 1) +
  labs(x = "Year", y = 'Index') +
  theme_bw(base_size = 20) +
  facet_wrap(~Idx, scales = 'free', ncol = 2)


## End(Not run)
```

`get_idx_fits_plot` *Get Index Fits Plot*

Description

Get Index Fits Plot

Usage

```
get_idx_fits_plot(data, rep, model_names)
```

Arguments

`data`	List of n_models of ‘SPoRC‘ data lists
`rep`	List of n_models of ‘SPoRC‘ report lists
`model_names`	Vector of model names

Value

A plot of fitted values to various indices across models

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
get_idx_fits_plot(list(data1, data2), list(rep1, rep2), c("Model1", "Model2"))

## End(Not run)
```

`get_key_quants` *Generate Key Projection Quantities and Table Plot*

Description

Calculates biological and fishery reference points and performs population projections to estimate terminal spawning biomass, catch advice, and reference point values by model and region. Also returns a formatted table plot of key quantities.

Usage

```
get_key_quants(data, rep, reference_points_opt, proj_model_opt, model_names)
```

Arguments

`data`	A list of model input data objects, one for each model (i.e., a list of SPoRC-formatted data lists). Each element should contain information on regions, years, ages, fleets, and biological inputs (e.g., weight-at-age, maturity, mortality).
`rep`	A list of model output objects, one for each model (i.e., a list of SPoRC-formatted report lists). Each element must include recruitment, selectivity, mortality, and numbers-at-age.

`reference_points_opt`	A named list specifying options for reference point calculations. See `Get_Reference_Points` for more details. Must include: **SPR_x** Spawning potential ratio (e.g., 0.4) for calculating F reference points. May be `NULL` if using `bh_rec`. **t_spwn** Fraction of year when spawning occurs (e.g., 0.5). **sex_ratio_f** Proportion of recruits that are female. **calc_rec_st_yr** Start year for averaging recruitment. **rec_age** Recruitment age. **type** Reference point calculation method (e.g., "multi_region"). **what** Type of output requested from the reference point function.
`proj_model_opt`	A named list of projection settings. See `Do_Population_Projection` for details. Must include: **n_proj_yrs** Number of years to project forward. **HCR_function** Harvest control rule function to use. **recruitment_opt** Recruitment assumption (e.g., "mean_rec", "bh_rec", "inv_gauss"). **fmort_opt** Fishing mortality assumption (e.g., "input", "HCR"). **n_avg_yrs** Number of years to average over for projection inputs.
`model_names`	A character vector of model identifiers (e.g., c("Base", "Alt1", "Alt2")), one for each element in `data` and `rep`.

Details

This function checks input list completeness, calculates reference points using `Get_Reference_Points()`, performs population projections with `Do_Population_Projection()`, and assembles both tabular and visual summaries.

If `recruitment_opt` is set to "inv_gauss", a warning is issued since only a single simulation will be run. This is typically not appropriate and an alternative assumption is recommended.

Value

A list with two elements:

`[[1]]`	A data.frame of key quantities by model and region, including terminal SSB, catch advice, and reference points.
`[[2]]`	A cowplot table plot (ggdraw object) of the same key quantities.

See Also

`Get_Reference_Points`, `Do_Population_Projection`

Other Reference Points and Projections: `Do_Population_Projection()`, `Get_Reference_Points()`

Examples

```
## Not run:
reference_points_opt <- list(SPR_x = 0.4,
                             t_spwn = 0,
                             sex_ratio_f = 0.5,
                             calc_rec_st_yr = 20,
                             rec_age = 2,
                             type = "multi_region",
                             what = "global_SPR")

proj_model_opt <- list(
  n_proj_yrs = 2,
  n_avg_yrs = 1,
  HCR_function = function(x, frp, brp, alpha = 0.05) {
    stock_status <- x / brp
    if (stock_status >= 1) f <- frp
   if (stock_status > alpha && stock_status < 1) f <- frp * (stock_status - alpha) / (1 - alpha)
    if (stock_status < alpha) f <- 0
    return(f)
  },
  recruitment_opt = "mean_rec",
  fmort_opt = "HCR"
)

out <- get_key_quants(list(mlt_rg_sable_data),
                      list(mlt_rg_sable_rep),
                      reference_points_opt,
                      proj_model_opt,
                      "Model 1")
out[[1]]  # key quantities data.frame
out[[2]]  # table plot

## End(Not run)
```

`get_logistN_Sigma` *Construct logistic-normal covariance matrix*

Description

Helper function to generate the covariance matrix (Σ) used in logistic-normal composition models. The structure depends on the specification of `comp_like`:

- `comp_like` = 2: independent and identically distributed (iid) across categories ($n_{\text{categories}}$).
- `comp_like` = 3: first-order autoregressive (AR1) correlation across categories ($n_{\text{categories}}$).
- `comp_like` = 4: two-dimensional AR1 correlation across categories and sexes ($n_{\text{categories}} \times n_{\text{sexes}}$).

Usage

```
get_logistN_Sigma(
  comp_like,
  n_bins,
  n_sexes,
  theta,
  corr_b = NULL,
  corr_s = NULL
)
```

Arguments

`comp_like`	Integer specifying the logistic-normal correlation structure: • 2 = iid across categories • 3 = AR1 across categories • 4 = AR1 across categories and sexes
`n_bins`	Number of composition categories (e.g., age or length bins). For `comp_like` = `2, 3`, the covariance matrix is dimensioned `n_bins`. For `comp_like` = `4`, it is dimensioned `n_bins` `*` `n_sexes`.
`n_sexes`	Number of sexes. Required when `comp_like` = `4`.
`theta`	Standard deviation parameter controlling the overall scale of the covariance.
`corr_b`	Correlation parameter across categories, in the interval $(-1, 1)$. Used when `comp_like` = `3` or `4`.
`corr_s`	Correlation parameter across sexes, in the interval $(-1, 1)$. Used when `comp_like` = `4`.

Value

A covariance matrix Σ with dimension:

- `n_bins` (`comp_like` = `2, 3`)
- `n_bins` `*` `n_sexes` (`comp_like` = `4`)

See Also

Other Utility: `fit_model()`, `get_par_est_info()`, `post_optim_sanity_checks()`, `rho_trans()`, `set_data_indicator_unused()`

Examples

```
## Not run:
n_cat <- 5
n_sexes <- 2

# iid example (categories only)
get_logistN_Sigma(comp_like = 2, n_bins = n_cat, n_sexes = NULL, theta = 0.5)
```

```
# AR1 across categories
get_logistN_Sigma(comp_like = 3, n_bins = n_cat, n_sexes = NULL, theta = 0.5,
                  corr_b = 0.3)

# AR1 across categories and sexes
get_logistN_Sigma(comp_like = 4, n_bins = n_cat, n_sexes = n_sexes, theta = 0.5,
                  corr_b = 0.3, corr_s = 0.2)

## End(Not run)
```

`get_model_rep_from_mcmc` *Extract model report from MCMC posterior samples*

Description

This function collapses MCMC chains from an RTMB/ADNUTS object, generates model reports for each posterior draw, and extracts specified components of the report.

Usage

```
get_model_rep_from_mcmc(rtmb_obj, adnuts_obj, what, n_cores)
```

Arguments

`rtmb_obj`	An RTMB object created via ‘ADFun‘.
`adnuts_obj`	An ‘adnuts‘ object containing MCMC samples.
`what`	Character vector specifying the names of components in the model report to extract.
`n_cores`	Number of cores to use

Value

A named list of ‘data.table‘s, one for each element in ‘what‘. Each table contains the melted report component across all posterior samples, with an additional column ‘posterior_sample‘ indicating the MCMC draw index.

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
model_reports <- get_model_rep_from_mcmc(rtmb_obj, adnuts_obj,
                                         what = c("SSB", "Rec"))

## End(Not run)
```

`get_nLL_plot` *Get plot of negative log likelihood values*

Description

Get plot of negative log likelihood values

Usage

```
get_nLL_plot(data, rep, model_names)
```

Arguments

`data`	List of data from ‘SPoRC‘
`rep`	List of n_models of ‘SPoRC‘ report lists
`model_names`	Vector of model names

Value

Plot and tables of negative log likelihood values across models

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
get_nLL_plot(list(data1, data2), list(rep1, rep2), c("Model1", "Model2"))

## End(Not run)
```

`get_osa` *Compute OSA residuals for composition data*

Description

Formats observed and expected composition data and calculates one-step-ahead (OSA) residuals using multinomial, Dirichlet-multinomial, or logistic-normal likelihoods. This function is the main interface for residual diagnostics, internally calling [run_osa()] to perform the residual calculations.

Usage

```
get_osa(
  obs_mat,
  exp_mat,
  N = NULL,
  DM_theta = NULL,
  LN_Sigma = NULL,
  years,
  fleet,
  bins,
  comp_type,
  bin_label,
  comp_like = 0,
  addtocomp = 0
)
```

Arguments

`obs_mat`	Array of observed compositions, dimensioned by `[region, year, bin, sex, fleet]`. May contain `NA`s, which are removed when filtering by `years`.
`exp_mat`	Array of expected compositions, dimensioned the same as `obs_mat`. May contain `NA`s, which are removed when filtering by `years`.
`N`	Input (or effective if Multinomial) sample size. Dimensions depend on `comp_type`: • `comp_type = 0` (aggregated): vector of length `n_years`. • `comp_type = 1` (split by region and sex): array `[n_regions, n_years, n_sexes]`. • `comp_type = 2` (split by region, joint by sex): matrix `[n_regions, n_years]`. For years without data, users can simply input an NA or any abritary number (it gets filtered out within the function).
`DM_theta`	Dirichlet-multinomial overdispersion parameter(s). Dimensions must match `N`: • aggregated: scalar • split by sex: matrix `[n_regions, n_sexes]` • joint by sex: vector of length `n_regions`
`LN_Sigma`	Logistic-normal covariance matrix. Dimensions depend on `comp_type`:

	• aggregated: matrix `[n_bins, n_bins]` • split by region and sex: array `[n_regions, n_bins, n_bins, n_sexes]` • joint by sex: array `[n_regions, n_bins, n_bins]` Use [get_logistN_Sigma()] to help construct this input.
`years`	Vector of years to filter to if composition type is aggregated (0). Otherwise, this expects a list where each list element is a vector of years for each region where compositions are available for use (split by region and sex, or split by region, joint by sex).
`fleet`	Fleet identifier (character or numeric) to filter to.
`bins`	Vector of age or length bin labels corresponding to the composition categories.
`comp_type`	Integer specifying how compositions are structured: • 0 = aggregated across regions and sexes • 1 = split by region and sex • 2 = split by region, joint by sex
`bin_label`	Character label describing whether bins represent ages or lengths.
`comp_like`	Integer specifying the likelihood type (defaults to 0): • 0 = multinomial • 1 = Dirichlet-multinomial • 2–4 = logistic-normal variants
`addtocomp`	Constant that is added to compositions

Value

A list with one element:

res Data frame of OSA residuals. Columns include: `fleet`, `index_label`, `year`, `index`, `resid`, `region`, `sex`, and `comp_type`.

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`, `plot_resids()`

`get_par_est_info` *Helper function for extracting parameter information and names from TMB*

Description

Helper function for extracting parameter information and names from TMB

Usage

```
get_par_est_info(parameters, mapping, sd_rep)
```

Arguments

`parameters`	Parameter list from setting up TMB object
`mapping`	Mapping list from setting up TMB object
`sd_rep`	SD Report from TMB obj

Value

A list of dataframes for estimated and non-estimated parameter values.

See Also

Other Utility: `fit_model()`, `get_logistN_Sigma()`, `post_optim_sanity_checks()`, `rho_trans()`, `set_data_indicator_unused()`

`Get_Reference_Points` *Wrapper function to get reference points*

Description

Wrapper function to compute fishing and biological reference points given data and report objects from an assessment or simulation. Supports both single-region and multi-region calculations with options for SPR or Beverton–Holt MSY reference points.

Usage

```
Get_Reference_Points(
  data,
  rep,
  SPR_x = NULL,
  t_spwn = 0,
  sex_ratio_f = rep(0.5, data$n_regions),
  calc_rec_st_yr = 1,
  rec_age = 1,
  type,
  what,
  n_avg_yrs = 1,
  local_bh_msy_newton_steps = 6
)
```

Arguments

`data`	List. Data object containing ages, years, weight-at-age, maturity, natural mortality, and other simulation/assessment info.
`rep`	List. Report object from RTMB containing estimated parameters like Fmort, selectivity, recruitment, steepness.
`SPR_x`	Numeric. Target Spawning Potential Ratio fraction. Required for SPR-based reference points.
`t_spwn`	Numeric. Mortality time until spawning.
`sex_ratio_f`	Numeric vector. Female sex ratio by region.
`calc_rec_st_yr`	Integer. First year used to compute mean recruitment.
`rec_age`	Integer. Age at recruitment.
`type`	Character. "single_region" or "multi_region".
`what`	Character. Type of reference point: **SPR** Single-region SPR reference point **independent_SPR** Multi-region SPR without movement **global_SPR** Multi-region SPR with movement **BH_MSY** Single-region Beverton–Holt MSY **independent_BH_MSY** Multi-region BH-MSY without movement **global_BH_MSY** Multi-region global BH-MSY with movement **local_BH_MSY** Multi-region local BH-MSY with movement
`n_avg_yrs`	Integer. Number of years to average demographic rates when calculating reference points.
`local_bh_msy_newton_steps`	Number of newton steps to take to solve for equilibrium recruitment in the origin region when local_BH_MSY is assumed.

Value

A list with elements:

f_ref_pt Vector of fishing reference points for each region.

b_ref_pt Vector of biological reference points for each region.

virgin_b_ref_pt Vector of virgin biomass reference points for each region.

See Also

Other Reference Points and Projections: `Do_Population_Projection()`, `get_key_quants()`

`get_retrospective_plot`

Get Retrospective Plot

Description

Get Retrospective Plot

Usage

```
get_retrospective_plot(retro_output, Rec_Age)
```

Arguments

`retro_output`	Dataframe generated from do_retrospective
`Rec_Age`	Age in which recruitment occurs

Value

A retrospective plot of recruitment and SSB in relative and absolute scales, as well as a retrospective plot of recruitment by cohort (squid plot)

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_relative_difference()`, `plot_resids()`

Examples

```
## Not run:
# do retrospective
retro <- do_retrospective(n_retro = 7, # number of retro peels to run
data = data, # rtmb data
parameters = parameters, # rtmb parameters
mapping = mapping, # rtmb mapping
random = NULL, # if random effects are used
do_par = TRUE, # whether or not to parralleize
n_cores = 7, # if parallel, number of cores to use
do_francis = F, # if we want tod o Francis
n_francis_iter = NULL # Number of francis iterations to do
)
get_retrospective_plot(retro, Rec_Age = 2)

## End(Not run)
```

`get_retrospective_relative_difference` *Derive relative difference from terminal year from a retrospective analysis.*

Description

Derive relative difference from terminal year from a retrospective analysis.

Usage

```
get_retrospective_relative_difference(retro_data)
```

Arguments

`retro_data`	Dataframe outputted from do_retrospective function

Value

Returns a data frame with relative difference of SSB and recruitment from the terminal year

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `plot_resids()`

`get_selex_plot` *Get Fishery and Survey Selectivity Plots*

Description

Get Fishery and Survey Selectivity Plots

Usage

```
get_selex_plot(rep, model_names, Selex_Type = "age", year_indx = NULL)
```

Arguments

`rep`	List of n_models of ‘SPoRC‘ report lists
`model_names`	Vector of model names
`Selex_Type`	Character vector specifying whether to output age or length-based selectivity (age, length)
`year_indx`	Year index for which selectivity year to plot (can be an integer or vector)

Value

Plots of terminal year fishery and survey selectivity by fleet, region, and sex across models

See Also

Other Plotting: `get_biological_plot()`, `get_data_fitted_plot()`, `get_ts_plot()`, `plot_all_basic()`, `theme_sablefish()`

Examples

```
## Not run:
get_selex_plot(list(rep1, rep2), c("Model1", "Model2"), year_indx = c(1:30))

## End(Not run)
```

`get_ts_plot` *Get Time Series Plots*

Description

Get Time Series Plots

Usage

```
get_ts_plot(rep, sd_rep, model_names, do_ci = TRUE)
```

Arguments

`rep`	List of n_models of ‘SPoRC‘ report lists
`sd_rep`	List of n_models of ‘SPoRC‘ sdreport lists
`model_names`	Vector of model names
`do_ci`	Boolean for whether confidence intervals are plotted

Value

Plots of spawning biomass, dynamic b0, total biomass, recruitment, and fishing mortality timeseries across models

See Also

Other Plotting: `get_biological_plot()`, `get_data_fitted_plot()`, `get_selex_plot()`, `plot_all_basic()`, `theme_sablefish()`

Examples

```
## Not run:
 get_ts_plot(list(rep1, rep2), list(sd_rep1, sd_rep2), c("Model1", "Model2"), do_ci = TRUE)

## End(Not run)
```

`marg_AIC` *Calculate the Corrected marginal AIC (AICc) from Optimization Results*

Description

Computes the corrected marginal Akaike Information Criterion (AICc) for model selection using optimization results. It supports objects returned from different optimizers, such as ‘optim‘ or ‘nlminb‘.

Usage

```
marg_AIC(opt, p = 2, n = Inf)
```

Arguments

`opt`	A list containing optimization results. Must include either: • ‘"par"‘ and ‘"objective"‘ (e.g., from ‘optim‘), or • ‘"par"‘ and ‘"value"‘ (e.g., from ‘nlminb‘)
`p`	Numeric. Penalty multiplier for the number of parameters. Default is 2.
`n`	Numeric. Sample size. Default is ‘Inf‘.

Value

Numeric. The corrected AIC (AICc) value.

`mlt_rg_sable_data` *Sablefish data for multi region (5 area) case study*

Description

A dataset containing the necessary elements for the Alaska sablefish spatial case study.

Usage

```
mlt_rg_sable_data
```

Format

A list with multiple components needed for the multi (5) region sablefish model

Source

Cheng et al. 2025, Fish and Fisheries; Panmictic Panacea? Demonstrating Good Practices for Developing Spatial Stock Assessments through Application to Alaska Sablefish (Anoplopoma fimbria)

`mlt_rg_sable_rep` *Sablefish report for 5 region case study*

Description

A report containing results for the spatial Alaska sablefish case study.

Usage

```
mlt_rg_sable_rep
```

Format

Report file from the 5 region sablefish case study

Source

Cheng et al. 2025, Fish and Fisheries; Panmictic Panacea? Demonstrating Good Practices for Developing Spatial Stock Assessments through Application to Alaska Sablefish (Anoplopoma fimbria)

`plot_all_basic` *Plotting function for all basic quantities*

Description

Plotting function for all basic quantities

Usage

```
plot_all_basic(data, rep, sd_rep, model_names, out_path)
```

Arguments

`data`	List of n_models of ‘SPoRC‘ data lists
`rep`	List of n_models of ‘SPoRC‘ report lists
`sd_rep`	List of n_models of sd report lists from ‘SPoRC‘
`model_names`	Character vector of model names
`out_path`	Path to the output directory. Users only need to specify the path.

Value

A series of plots compared across models outputted as a pdf in the specified directory

See Also

Other Plotting: `get_biological_plot()`, `get_data_fitted_plot()`, `get_selex_plot()`, `get_ts_plot()`, `theme_sablefish()`

Examples

```
## Not run:
plot_all_basic(
  data = list(data1, data2),
  rep = list(rep1, rep2),
  sd_rep = list(sd_rep1, sd_rep2),
  model_names = c("Model1", "Model2"),
  out_path = here::here()
)

## End(Not run)
```

`plot_resids` *Plots OSA residuals from outputs from get_osa. Much of this code is taken from the afscOM package, but with modificaitons to plot features.*

Description

Plots OSA residuals from outputs from get_osa. Much of this code is taken from the afscOM package, but with modificaitons to plot features.

Usage

```
plot_resids(osa_results)
```

Arguments

`osa_results`	List object obtained from get_osa, that contains a dataframe of residuals and aggregated fits.

Value

A vareity of plots for OSA residuals (list)

See Also

Other Model Diagnostics: `do_jitter()`, `do_likelihood_profile()`, `do_retrospective()`, `do_runs_test()`, `get_catch_fits_plot()`, `get_comp_prop()`, `get_idx_fits()`, `get_idx_fits_plot()`, `get_model_rep_from_mcmc()`, `get_nLL_plot()`, `get_osa()`, `get_retrospective_plot()`, `get_retrospective_relative_difference()`

Examples

```
## Not run:
comp_props <- get_comp_prop(data = data, rep = sabie_rtmb_model$rep,
 age_labels = 2:31, len_labels = seq(41, 99, 2), year_labels = 1960:2024)
plot_resids(get_osa(obs_mat = comp_props$Obs_FishAge_mat,
                    exp_mat = comp_props$Pred_FishAge_mat,
                    N = rep(16.52215, length(1999:2023)),
                    years = which(1960:2024 %in% 1999:2023),
                    LN_Sigma = LN_Sigma,
                    fleet = 1,
                    bins = 2:31,
                    comp_type = 0,
                    comp_like = 0,
                    bin_label = "Age"))
osa_plot <- plot_resids(osa_results)

## End(Not run)
```

`post_optim_sanity_checks` *Post Optimization Model Convergence Checks*

Description

Post Optimization Model Convergence Checks

Usage

```
post_optim_sanity_checks(
  sd_rep,
  rep,
  gradient_tol = 0.001,
  se_tol = 100,
  corr_tol = 0.99
)
```

Arguments

`sd_rep`	sd report list from a ‘SPoRC‘ model
`rep`	report list from a ‘SPoRC‘ model
`gradient_tol`	Value for maximum gradient tolerance to use
`se_tol`	Value for maximum standard error tolerance to use
`corr_tol`	Value for maximum correlation tolerance to use

See Also

Other Utility: `fit_model`(), `get_logistN_Sigma`(), `get_par_est_info`(), `rho_trans`(), `set_data_indicator_unused`

`rho_trans` *Title Constrains value between -1 and 1*

Description

Title Constrains value between -1 and 1

Usage

```
rho_trans(x)
```

Arguments

`x`	Numeric value to constrain

Value

Constrained value between -1 and 1

See Also

Other Utility: `fit_model()`, `get_logistN_Sigma()`, `get_par_est_info()`, `post_optim_sanity_checks()`, `set_data_indicator_unused()`

`run_annual_cycle` *Run Annual Cycle in Simulation Environment*

Description

Run Annual Cycle in Simulation Environment

Usage

```
run_annual_cycle(y, sim, sim_env)
```

Arguments

`y`	Year index
`sim`	Simulation index
`sim_env`	Simulation environment will all the necessary elements to run the annual cycle

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`run_francis` *Run Iterative Francis Reweighting Procedure*

Description

Runs an iterative Francis reweighting procedure for composition data (fishery and survey age- and length-compositions). The function reweights input data, repeatedly fits the model, and computes updated Francis weights.

Usage

```
run_francis(
  data,
  parameters,
  mapping,
  random = NULL,
  n_francis_iter = 10,
  newton_loops = 0
)
```

Arguments

`data`	A list of model input data, including at least observed compositions (‘ObsFishAgeComps‘, ‘ObsFishLenComps‘, ‘ObsSrvAgeComps‘, ‘ObsSrvLenComps‘) and corresponding weights (‘Wt_FishAgeComps‘, ‘Wt_FishLenComps‘, ‘Wt_SrvAgeComps‘, ‘Wt_SrvLenComps‘).
`parameters`	A list of model parameters to be passed to [fit_model()].
`mapping`	A list or mapping object used to specify fixed or estimated parameters in [fit_model()].
`random`	A character string of random effects passed to [fit_model()].
`n_francis_iter`	Integer. Number of Francis reweighting iterations to perform. Default is ‘10‘.
`newton_loops`	Integer. Number of Newton loops passed to [fit_model()]. Default is ‘0‘.

Value

A list with three elements:

obj The fitted model object returned by [fit_model()], including all elements of a TMB object, data, parameters, mapping, random effects specified, and report.

end_mean_francis A summary of the mean Francis weights from the first iteration.

end_mean_francis A summary of the mean Francis weights from the final iteration.

recorded_weights A summary of recorded francis weights from each iteartion.

See Also

Other Francis Reweighting: `do_francis_reweighting()`

Examples

```
## Not run:
  out <- run_francis(data = data,
                     parameters = parameters,
                     mapping = mapping,
                     random = NULL,
                     n_francis_iter = 5,
                     newton_loops = 3)
  out$obj
  out$mean_francis

## End(Not run)
```

`set_data_indicator_unused`

Set Data Indicators to Unused for Specified Years

Description

Set Data Indicators to Unused for Specified Years

Usage

```
set_data_indicator_unused(
  data,
  unused_years,
 what = c("Catch", "FishIdx", "FishAgeComps", "FishLenComps", "SrvIdx", "SrvAgeComps",
    "SrvLenComps", "Tagging")
)
```

Arguments

`data`	Data list for RTMB model
`unused_years`	Integer vector specifying which years to mark as unused. Only years present in `data$years` are considered.
`what`	Character vector specifying which data types to modify. Possible values include: **"Catch"** Catch data indicators. **"FishIdx"** Fishery index data indicators. **"FishAgeComps"** Fishery age composition data indicators. **"FishLenComps"** Fishery length composition data indicators. **"SrvIdx"** Survey index data indicators. **"SrvAgeComps"** Survey age composition data indicators. **"SrvLenComps"** Survey length composition data indicators. **"Tagging"** Tagging data and associated cohorts.

Value

The modified `data` object, with indicators set to 0 for the specified years and tagging cohorts removed if relevant.

See Also

Other Utility: `fit_model()`, `get_logistN_Sigma()`, `get_par_est_info()`, `post_optim_sanity_checks()`, `rho_trans()`

`Setup_Mod_Biologicals` *Setup biological inputs for estimation model*

Description

Setup biological inputs for estimation model

Usage

```
Setup_Mod_Biologicals(
  input_list,
  WAA,
  WAA_fish = NULL,
  WAA_srv = NULL,
  MatAA,
  addtocomp = 0.001,
  addtofishidx = 1e-04,
  addtosrvidx = 1e-04,
  addtotag = 1e-10,
  AgeingError = NULL,
  Use_M_prior = 0,
  M_prior = NA,
  fit_lengths = 0,
  SizeAgeTrans = NA,
  Selex_Type = "age",
  M_spec = "est_ln_M",
  M_ageblk_spec = "constant",
  M_regionblk_spec = "constant",
  M_yearblk_spec = "constant",
  M_sexblk_spec = "constant",
  Fixed_natmort = NULL,
  ...
)
```

Arguments

`input_list`	List containing data, parameter, and map lists for the model.
`WAA`	Numeric array of weight-at-age (spawning), dimensioned `[n_regions, n_years, n_ages, n_sexes]`.
`WAA_fish`	Numeric array of weight-at-age (fishery), dimensioned `[n_regions, n_years, n_ages, n_sexes, n_fish_fleets]`.
`WAA_srv`	Numeric array of weight-at-age (survey), dimensioned `[n_regions, n_years, n_ages, n_sexes, n_srv_fleets]`.
`MatAA`	Numeric array of maturity-at-age, dimensioned `[n_regions, n_years, n_ages, n_sexes]`.
`addtocomp`	Numeric value for a constant to add to composition data. Default is 1e-3. Not used if logistic normal likelihoods are utilized.
`addtofishidx`	Numeric value for a constant to add to composition data. Default is 1e-4.
`addtosrvidx`	Numeric value for a constant to add to composition data. Default is 1e-4.
`addtotag`	Numeric value for a constant to add to composition data. Default is 1e-10
`AgeingError`	Numeric matrix or array representing the ageing error transition matrix. If a matrix (2D), dimensions should be `[number of modeled ages, number of observed composition ages]` and the ageing error is assumed to be constant over time. If an array (3D), dimensions should be `[number of years, number of modeled ages, number of observed composition ages]` allowing ageing error to vary by year. Defaults to an identity matrix (no ageing error) if not specified, assuming observed age bins exactly match modeled age bins. **Note:** If the observed age composition bins differ from the modeled age bins (e.g., observed ages 2–10 while modeled ages are 1–10), the default identity matrix will cause a dimensional mismatch and misalignment. In such cases, users should provide a custom ageing error matrix mapping modeled to observed ages. For example, to drop the first modeled age bin, supply a matrix like `diag(1, 10)[, 2:10]`. This ensures proper alignment of age bins for likelihood calculations.
`Use_M_prior`	Integer flag indicating whether to apply a natural mortality prior (`0` = no, `1` = yes).
`M_prior`	Numeric vector of length two giving the mean (in normal space) and standard deviation of the natural mortality prior.
`fit_lengths`	Integer flag indicating whether to fit length data (`0` = no, `1` = yes).
`SizeAgeTrans`	Numeric array of size-at-age transition probabilities, dimensioned `[n_regions, n_years, n_lens, n_ages, n_sexes]`.
`Selex_Type`	Character string specifying whether selectivity is age or length-based. Default is age-based • `"length"`: Length-based selectivity. • `"age"`: Age-based selectivity
`M_spec`	Character string specifying natural mortality estimation approach. Defaults to `est_ln_M`, which estimates mortality to be invariant, if blocks are not specified. Options:

- `"est_ln_M"`: Estimates natural mortality across the defined natural mortality blocks.
- `"fix"`: Fix all natural mortality parameters using the provided array.

`M_ageblk_spec`	Specification of age blocking for natural mortality estimation. Either a character string ("constant") or a list of index vectors, e.g., `list(1:10, 11:30)`, which specifies 2 age blocks for M.
`M_regionblk_spec`	Specification of regional blocking for natural mortality. Either a character string ("constant") or a list of index vectors, e.g., `list(1:3, 4:5)`, which specifies 2 region blocks for M.
`M_yearblk_spec`	Specification of year blocking for natural mortality. Either a character string ("constant") or a list of index vectors, e.g., `list(1:10, 11:30)`, which specifies 2 year blocks for M.
`M_sexblk_spec`	Specification of sex blocking for natural mortality. Either a character string ("constant") or a list of index vectors, e.g., `list(1,2)`, which specifies sex-invariant M.
`Fixed_natmort`	Numeric array of fixed natural mortality values, dimensioned `[n_regions, n_years, n_ages, n_sexes]`. Required if `M_spec` = `"fix"`.
`...`	Additional arguments for starting values such as `ln_M` and `M_offset`. These are ignored if `M_spec` = `fix`.

See Also

Other Model Setup: `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Catch_and_F` *Setup fishing mortality and catch observations*

Description

Setup fishing mortality and catch observations

Usage

```
Setup_Mod_Catch_and_F(
  input_list,
  ObsCatch,
  catch_units = array("biom", dim = c(input_list$data$n_regions,
    input_list$data$n_fish_fleets)),
  Catch_Type = array(1, dim = c(length(input_list$data$years),
    input_list$data$n_fish_fleets)),
  UseCatch,
  Use_F_pen = 1,
```

```
  est_all_regional_F = 1,
  sigmaC_spec = "fix",
  sigmaC_agg_spec = "fix",
  sigmaF_spec = "fix",
  sigmaF_agg_spec = "fix",
  ...
)
```

Arguments

`input_list`	A list containing data, parameters, and map lists used by the model.
`ObsCatch`	Numeric array of observed catches, dimensioned `[n_regions, n_years, n_fish_fleets]`.
`catch_units`	Catch units - Array dimensioned by n_regions x n_fish_fleets • `"abd"`: Catch units in abundance • `"biom"`: Catch units in biomass (default)
`Catch_Type`	Integer matrix with dimensions `[n_years, n_fish_fleets]`, specifying catch data types: • `0`: Use aggregated catch data for the year. • `1`: Use region-specific catch data for the year.
`UseCatch`	Indicator array `[n_regions, n_years, n_fish_fleets]` specifying whether to include catch data in the fit: • `0`: Do not use catch data. • `1`: Use catch data and fit.
`Use_F_pen`	Integer flag indicating whether to apply a fishing mortality penalty: • `0`: Do not apply penalty. • `1`: Apply penalty.
`est_all_regional_F`	Integer flag indicating whether all regional fishing mortality deviations are estimated: • `0`: Some fishing mortality deviations are aggregated across regions. • `1`: All fishing mortality deviations are regional.
`sigmaC_spec`	Character string specifying observation error structure for catch data. Default behavior fixes `sigmaC` at a starting value of `1e-3` (log-scale `ln_sigmaC = log(1e-3)`) for all regions, years, and fleets. Other options include: • `"est_shared_f"`: Estimate `sigmaC` shared across fishery fleets, unique by region and year. • `"est_shared_r"`: Estimate `sigmaC` shared across regions, unique by fleet and year. • `"est_shared_y"`: Estimate `sigmaC` shared across years, unique by region and fleet. • `"est_shared_r_f"`: Estimate `sigmaC` shared across regions and fleets, unique by year. • `"est_shared_f_y"`: Estimate `sigmaC` shared across fleets and years, unique by region.

- `"est_shared_r_y"`: Estimate `sigmaC` shared across regions and years, unique by fleet.
- `"est_shared_r_y_f"`: Estimate single `sigmaC` shared across regions, years, and fleets.
- `"fix"`: Fix `sigmaC` at the starting value.
- `"est_all"`: Estimate separate `sigmaC` for each region, year, and fleet combination.

`sigmaC_agg_spec`

Character string specifying process error structure for aggregated catch observation error. Default fixes `sigmaC_agg` at the starting value (log-scale `sigmaC_agg`). Other options include:

- `"est_shared_f"`: Estimate `sigmaC_agg` shared across fishery fleets, unique by year.
- `"est_shared_y"`: Estimate `sigmaC_agg` shared across years, unique by fleet.
- `"est_shared_y_f"`: Estimate single `sigmaC_agg` shared across years and fleets.
- `"fix"`: Fix at the starting value.
- `"est_all"`: Estimate separate parameters for each year and fleet combination.

`sigmaF_spec` Character string specifying process error structure for fishing mortality. Default fixes `sigmaF` at `1` on the log scale (i.e., `ln_sigmaF = 0`). Other options include:

- `"est_shared_f"`: Estimate `sigmaF` shared across fishery fleets.
- `"est_shared_r"`: Estimate `sigmaF` shared across regions but unique by fleet.
- `"est_shared_r_f"`: Estimate `sigmaF` shared across regions and fleets.
- `"fix"`: Fix `sigmaF` at the starting value.
- `"est_all"`: Estimate separate `sigmaF` for each region and fleet.

`sigmaF_agg_spec`

Character string specifying process error structure for aggregated fishing mortality. Default fixes `sigmaF_agg` at the starting value (log-scale `ln_sigmaF_agg`). Other options include:

- `"est_shared_f"`: Estimate `sigmaF_agg` shared across fishery fleets.
- `"fix"`: Fix at the starting value.
- `"est_all"`: Estimate separate parameters for each fishery fleet.

`...` Additional arguments specifying starting values for `ln_sigmaC`, `ln_sigmaF`, and `ln_sigmaF_agg`.

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Dim` *Set up model dimensions*

Description

Set up model dimensions

Usage

```
Setup_Mod_Dim(
  years,
  ages,
  lens,
  n_regions,
  n_sexes,
  n_fish_fleets,
  n_srv_fleets,
  n_proj_yrs_devs = 0,
  verbose = FALSE,
  store_config = FALSE
)
```

Arguments

`years`	Numeric vector of years.
`ages`	Numeric vector of age classes.
`lens`	Numeric vector of length bins; can be set to `1` if length data are not modeled.
`n_regions`	Integer specifying the number of spatial regions.
`n_sexes`	Integer specifying the number of sexes.
`n_fish_fleets`	Integer specifying the number of fishery fleets.
`n_srv_fleets`	Integer specifying the number of survey fleets.
`n_proj_yrs_devs`	Number of projection years for deviation parameters (ln_RecDevs, move_devs, ln_fishsel_devs, ln_srvsel_devs)
`verbose`	Logical flag indicating whether to print progress messages (default `FALSE`).
`store_config`	Logical flag indicating whether or not to store configuration (default `FALSE`).

Value

A list containing three named elements:

`data` List of data inputs dimensioned by the model dimensions.

`parameters` List of model parameters initialized according to dimensions.

`map` List of parameter mappings for model fitting.

`config` List of arguments being supplied into the Setup_Mod_* functions.

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

Setup_Mod_FishIdx_and_Comps

Setup observed fishery indices and composition data (age and length comps)

Description

Setup observed fishery indices and composition data (age and length comps)

Usage

```
Setup_Mod_FishIdx_and_Comps(
  input_list,
  ObsFishIdx,
  ObsFishIdx_SE,
  fish_idx_type,
  UseFishIdx,
  ObsFishAgeComps,
  UseFishAgeComps,
  ISS_FishAgeComps,
  ObsFishLenComps,
  UseFishLenComps,
  ISS_FishLenComps,
  FishAgeComps_LikeType,
  FishLenComps_LikeType,
  FishAgeComps_Type,
  FishLenComps_Type,
  ...
)
```

Arguments

`input_list`	List containing a data list, parameter list, and map list
`ObsFishIdx`	Observed fishery index data as a numeric array with dimensions `[n_regions, n_years, n_fish_fleets]`.
`ObsFishIdx_SE`	Standard errors associated with `ObsFishIdx`, also dimensioned `[n_regions, n_years, n_fish_fleets]`.
`fish_idx_type`	Character vector of length `n_fish_fleets` specifying the type of index data. Options are `"abd"` for abundance, `"biom"` for biomass, and `"none"` if no index is available.

`UseFishIdx`	Logical or binary indicator array (`[n_regions, n_years, n_fish_fleets]`) specifying whether to include a fishery index in the likelihood (`1`) or ignore it (`0`).
`ObsFishAgeComps`	Observed fishery age composition data as a numeric array with dimensions `[n_regions, n_years, n_ages, n_sexes, n_fish_fleets]`. Values should reflect counts or proportions (not required to sum to 1, but should be on a comparable scale).
`UseFishAgeComps`	Indicator array (`[n_regions, n_years, n_fish_fleets]`) specifying whether to fit fishery age composition data (`1`) or ignore it (`0`).
`ISS_FishAgeComps`	Input sample size for age compositions, array dimensioned `[n_regions, n_years, n_sexes, n_fish_fleets]`. Required if observed age comps are normalized (i.e., sum to 1), to correctly scale the contribution to the likelihood.
`ObsFishLenComps`	Observed fishery length composition data as a numeric array with dimensions `[n_regions, n_years, n_lens, n_sexes, n_fish_fleets]`. Values should reflect counts or proportions.
`UseFishLenComps`	Indicator array (`[n_regions, n_years, n_fish_fleets]`) specifying whether to fit fishery length composition data (`1`) or ignore it (`0`).
`ISS_FishLenComps`	Same as `ISS_FishAgeComps`, but for length compositions.
`FishAgeComps_LikeType`	Character vector of length `n_fish_fleets` specifying the likelihood type used for fishery age composition data. Options include `"Multinomial"`, `"Dirichlet-Multinomial"`, and `"iid-Logistic-Normal"`. Use `"none"` to omit the likelihood.
`FishLenComps_LikeType`	Same as `FishAgeComps_LikeType`, but for fishery length composition data.
`FishAgeComps_Type`	Character vector specifying how age compositions are structured by fleet and year range. Options include: • `"agg"`: Aggregated across regions and sexes. • `"spltRspltS"`: Split by region and by sex (compositions sum to 1 within region-sex group). • `"spltRjntS"`: Split by region but summed jointly across sexes. • `"none"`: No composition data used. Format each element as `"<type>_Year_<start>-<end>_Fleet_<fleet number>"` (e.g., `"agg_Year_1-10_Fleet_1"`).
`FishLenComps_Type`	Same as `FishAgeComps_Type`, but for length compositions.
`...`	Additional arguments specifying starting values for overdispersion parameters (e.g., `ln_FishAge_theta`, `ln_FishLen_theta`, `ln_FishAge_theta_agg`, `ln_FishLen_theta_agg`).

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Fishsel_and_Q`

Setup fishery selectivity and catchability specifications

Description

Setup fishery selectivity and catchability specifications

Usage

```
Setup_Mod_Fishsel_and_Q(
  input_list,
  cont_tv_fish_sel = paste("none_Fleet_", 1:input_list$data$n_fish_fleets, sep = ""),
  fish_sel_blocks = paste("none_Fleet_", 1:input_list$data$n_fish_fleets, sep = ""),
  fish_sel_model,
  Use_fish_q_prior = 0,
  fish_q_prior = NA,
  fish_q_blocks = paste("none_Fleet_", 1:input_list$data$n_fish_fleets, sep = ""),
  fishsel_pe_pars_spec = NULL,
  fish_fixed_sel_pars_spec = NULL,
  fish_q_spec = NULL,
  fish_sel_devs_spec = NULL,
  corr_opt_semipar = NULL,
  Use_fish_selex_prior = 0,
  fish_selex_prior = NULL,
  cont_tv_fish_sel_penalty = TRUE,
  fishsel_devs_shared_ages = NULL,
  ...
)
```

Arguments

`input_list`	List containing a data list, parameter list, and map list
`cont_tv_fish_sel`	Character vector specifying the form of continuous time-varying selectivity for each fishery fleet. The vector must be length `n_fish_fleets`, and each element must follow the structure: `"<time variation type>_Fleet_<fleet number>"`. Valid time variation types include: • `"none"`: No continuous time variation (default) • `"iid"`: Independent and identically distributed deviations across years. • `"rw"`: Random walk in time.

- `"3dmarg"`: 3D marginal time-varying selectivity.
- `"3dcond"`: 3D conditional time-varying selectivity.
- `"2dar1"`: Two-dimensional AR1 process.

For example:

- `"iid_Fleet_1"` applies an iid time-varying structure to Fleet 1.
- `"none_Fleet_2"` means no time variation is used for Fleet 2.

`fish_sel_blocks`

Character vector specifying the fishery selectivity blocks for each region and fleet.

Each element must follow one of the following structures:

- ‘"Block_<block number>_Year_<start>-<end>_Fleet_<fleet number>"‘
- ‘"Block_<block number>_Year_<start>-terminal_Fleet_<fleet number>"‘
- ‘"none_Fleet_<fleet number>"‘

This argument defines how fishery selectivity varies over time for each fleet:

- `"Block_..."` entries specify discrete time blocks during which selectivity parameters are assumed constant.
- `"none_..."` entries indicate that selectivity is constant across all years for the specified fleet.

If time-block-based selectivity is specified for a fleet (via `fish_sel_blocks`), its corresponding continuous selectivity option (in `cont_tv_fish_sel`) must be set to `"none_Fleet_<fleet number>"`. The two approaches—blocked and continuous time-varying selectivity—are mutually exclusive. The default for each fleet is `"none_Fleet_x"` (i.e., no selectivity blocks).

`fish_sel_model` Character vector specifying the fishery selectivity functional form for each fleet, and optionally by time block.

Each element must follow one of the following structures:

- `"<selectivity model>_Fleet_<fleet number>"`
- `"<selectivity model>_Fleet_<fleet number>_Block_<block number>"`

The first form applies a single selectivity model across all years for the specified fleet. The second form allows the user to assign a distinct selectivity model to a specific time block, as defined in `fish_sel_blocks`.

Available selectivity model types include:

- `"logist1"` — Logistic function with parameters `a50` and `k`.
- `"logist2"` — Logistic function with parameters `a50` and `a95`.
- `"gamma"` — Dome-shaped gamma function with parameters `amax` and `delta`.
- `"exponential"` — Exponential function with a power parameter.
- `"dbnrml"` — Double-normal function with six parameters.

If multiple selectivity time blocks are specified for a fleet (using `fish_sel_blocks`), then the corresponding selectivity model for each block must be explicitly defined using the `"<model>_Block_<block>_Fleet_<fleet>"` format. If blocks are not defined for a fleet, use the `"<model>_Fleet_<fleet number>"` format only. For mathematical definitions and implementation details of each selectivity form, refer to the model equations vignette.

`Use_fish_q_prior`

Integer (0 or 1). Flag to enable/disable fishery catchability priors. When set to 1, applies log-normal priors to fishery selectivity parameters as specified in `fish_q_prior`. When set to 0, no priors are applied.

`fish_q_prior` Data frame containing prior specifications for fishery catchability parameters. Must include columns: `region` (region index), `fleet` (fleet index), `block` (time block index), `mu` (prior mean on natural scale), and `sd` (prior standard deviation on log scale). Each row specifies a log-normal prior N(log(mu), sd) for a given catchability parameter. Only parameters with rows in this data frame will have priors applied.

`fish_q_blocks` Character vector specifying fishery catchability (q) blocks for each fleet. Each element must follow the structure: `"Block_<block number>_Year_<start>-<end>_Fleet_<fleet number>"` or `"none_Fleet_<fleet number>"`. Default is "none_Fleet_x".

This allows users to define time-varying catchability blocks independently of selectivity blocks. The blocks must be non-overlapping and sequential in time within each fleet.

For example:

- `"Block_1_Year_1-35_Fleet_1"` assigns block 1 to Fleet 1 for years 1–35.
- `"Block_2_Year_36-56_Fleet_1"` continues with block 2 for years 36–56.
- `"Block_3_Year_57-terminal_Fleet_1"` assigns block 3 from year 57 to the terminal year for Fleet 1.
- `"none_Fleet_2"` indicates no catchability blocks are used for Fleet 2.

Internally, these specifications are converted to a `[n_regions, n_years, n_fish_fleets]` array, where each block is mapped to the appropriate years and fleets.

`fishsel_pe_pars_spec`

Character string specifying how process error parameters for fishery selectivity are estimated across regions and sexes. This is only relevant if `cont_tv_fish_sel` is not set to `"none"`; otherwise, all process error parameters are treated as fixed.

Available options include:

- `"est_all"`: Estimates separate process error parameters for each region and sex.
- `"est_shared_r"`: Shares process error parameters across regions (sex-specific parameters are still estimated).
- `"est_shared_s"`: Shares process error parameters across sexes (region-specific parameters are still estimated).
- `"est_shared_r_s"`: Shares process error parameters across both regions and sexes, estimating a single set of parameters.
- `"est_shared_f_x"`: Shares process error parameters with another fleet, where `x` is the fleet number to share with. This option forces multiple fleets to have identical process error variance and correlation structures for their time-varying selectivity. For example, `"est_shared_f_2"` means the current fleet will use the same process error parameters as fleet 2. The reference fleet (fleet x) must use one of the other sharing options and cannot itself be sharing with another fleet.
- `"fix"` or `"none"`: Does not estimate process error parameters; all are treated as fixed.

`fish_fixed_sel_pars_spec`

Character string specifying the structure for estimating fixed-effect parameters of the fishery selectivity model (e.g., a50, k, amax). This controls whether selectivity parameters are estimated separately or shared across regions and sexes.

Available options include:

- `"est_all"`: Estimates separate fixed-effect selectivity parameters for each region and sex.
- `"est_shared_r"`: Shares parameters across regions (sex-specific parameters are still estimated).
- `"est_shared_s"`: Shares parameters across sexes (region-specific parameters are still estimated).
- `"est_shared_r_s"`: Shares parameters across both regions and sexes, estimating a single set of fixed-effect parameters.
- `"est_shared_f_x"`: Shares fixed-effect selectivity parameters with another fleet, where `x` is the fleet number to share with. This option forces multiple fleets to have identical selectivity curves by using the same underlying parameters (e.g., same a50, k, amax values). For example, `"est_shared_f_2"` means the current fleet will use the same fixed-effect selectivity parameters as fleet 2. The reference fleet (fleet x) must use one of the other sharing options and cannot itself be sharing with another fleet.
- `"fix"`: Fixes all selectivity parameters to their initial values (no estimation).
- `"none"`: No selectivity parameters are estimated (equivalent to `"fix"`).

`fish_q_spec`

Character string specifying the structure of fishery catchability (`q`) estimation across regions. This controls whether separate or shared parameters are used.

Available options include:

- `"est_all"`: Estimates separate catchability parameters for each region.
- `"est_shared_r"`: Estimates a single catchability parameter shared across all regions.

`fish_sel_devs_spec`

Character string specifying the structure of process error deviations in time-varying fishery selectivity dimensioned by the number of fishery fleets. This determines how deviations are estimated across regions and sexes.

Available options include:

- `"est_all"`: Estimates a separate deviation time series for each region and sex.
- `"est_shared_r"`: Shares deviations across regions (sex-specific deviations are still estimated).
- `"est_shared_s"`: Shares deviations across sexes (region-specific deviations are still estimated).
- `"est_shared_r_s"`: Shares deviations across both regions and sexes, estimating a single deviation time series.
- `"est_shared_a"`: Shares deviations across age blocks.
- `"est_shared_r_a"`: Shares deviations across regions and age shared blocks.
- `"est_shared_a_s"`: Shares deviations across age shared blocks and sexes.

- `"est_shared_r_a_s"`: Shares deviations across regions, age shared blocks, and sexes.
- `"est_shared_f_x"`: Shares deviations with another fleet, where `x` is the fleet number to share with. This option allows multiple fleets to use identical deviation parameters, reducing the number of parameters to estimate. For example, `"est_shared_f_2"` means the current fleet will use the same deviation parameters as fleet 2. The reference fleet (fleet x) must use one of the other sharing options (`"est_all"`, `"est_shared_r"`, `"est_shared_s"`, or `"est_shared_r_s"`) and cannot itself be sharing with another fleet.
- `"fix"`: Fixes all deviation parameters to zero (no time-variation).
- `"none"`: No deviation parameters are estimated (equivalent to `"fix"`).

This argument is only used when a continuous time-varying selectivity form is specified (e.g., via `cont_tv_fish_sel`).

`corr_opt_semipar`

Character string specifying which correlation structures to suppress when using semi-parametric time-varying selectivity models. Only used if `cont_tv_sel` is set to one of `"3dmarg"`, `"3dcond"`, or `"2dar1"`.

This option allows users to turn off estimation of specific correlation components in the time-varying selectivity model. This can improve stability or enforce assumptions about independence in the temporal or age structure.

Available options:

- `"corr_zero_y"`: Sets year (temporal) correlations to 0.
- `"corr_zero_b"`: Sets age correlations to 0.
- `"corr_zero_y_b"`: Sets both year and bin correlations to 0.
- `"corr_zero_c"`: Sets cohort correlations to 0. Only valid for `cont_tv_sel = "3dmarg"` or `"3dcond"`.
- `"corr_zero_y_c"`: Sets year and cohort correlations to 0. Only valid for `cont_tv_sel = "3dmarg"` or `"3dcond"`.
- `"corr_zero_b_c"`: Sets bin (age) and cohort correlations to 0. Only valid for `cont_tv_sel = "3dmarg"` or `"3dcond"`.
- `"corr_zero_y_b_c"`: Sets all correlations (year, bin (age), and cohort) to 0. Only valid for `cont_tv_sel = "3dmarg"` or `"3dcond"`; equivalent to an iid structure.

These correlation-suppression flags are ignored when `cont_tv_sel` is set to any other value.

`Use_fish_selex_prior`

Integer (0 or 1). Flag to enable/disable fishery selectivity priors. When set to 1, applies log-normal priors to fishery selectivity parameters as specified in `fish_selex_prior`. When set to 0, no priors are applied.

`fish_selex_prior`

Data frame containing prior specifications for fishery selectivity parameters. Must include columns: `region` (region index), `fleet` (fleet index), `block` (time block index), `sex` (sex index), `par` (parameter index), `mu` (prior mean on natural scale), and `sd` (prior standard deviation on log scale). Each row specifies a log-normal prior N(log(mu), sd) for one selectivity parameter. Only parameters with rows in this data frame will have priors applied.

`cont_tv_fish_sel_penalty`	Whether or not continuous fishery time varying selectivity penalties are applied (if cont_tv_fish_sel > 0)
`fishsel_devs_shared_ages`	List object for specifying which ages are shared when selectivity deviations are semi-parametric (e.g., list(1:5, 6:10, 11:30) specifies that ages 1-5, 6-10, and 11-30 have the same deviations.)
`...`	Additional arguments specifying starting values for fishery selectivity and catchability parameters (fishsel_pe_pars, ln_fishsel_devs, ln_fish_fixed_sel_pars, ln_fish_q, fish_q_coeff)

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Movement` *Setup Movement Processes for SPoRC*

Description

Configure movement model components (unstructured Markov or CTMC) and populate the model input and parameter lists with the appropriate data structures and starting values.

Usage

```
Setup_Mod_Movement(
  input_list,
  move_type = 0,
  do_recruits_move = 0,
  use_fixed_movement = 0,
  Fixed_Movement = NA,
  Use_Movement_Prior = 0,
  Movement_prior = NULL,
  Movement_ageblk_spec = "constant",
  Movement_yearblk_spec = "constant",
  Movement_sexblk_spec = "constant",
  cont_vary_movement = "none",
  Movement_cont_pe_pars_spec = "none",
  ctmc_move_dat = NULL,
  adjacency_mat = NULL,
  area_r = rep(1, input_list$data$n_regions),
  diffusion_formula = NULL,
  preference_formula = NULL,
  ctmc_diffusion_bounds = 0,
  ...
)
```

Arguments

`input_list`	List containing data, parameter, and map lists for the model. This object is updated in-place and returned by the function.
`move_type`	Integer indicating the movement model type: • `0` – unstructured Markov (parameters transformed via a multinomial logit), • `1` – Continuous Time Markov Chain (CTMC) formulation. Default is `0`.
`do_recruits_move`	Integer flag (0 or 1) indicating whether recruits move. Default is 0 (recruits do not move).
`use_fixed_movement`	Integer flag (0 or 1) indicating whether to use a fixed movement matrix (1) or estimate movement parameters (0). Default is 0.
`Fixed_Movement`	Numeric array specifying a fixed movement rate/matrix. It must be dimensioned as `[n_regions, n_regions, n_years, n_ages, n_sexes]`. If `NA` (the default), a neutral array of ones will be created internally.
`Use_Movement_Prior`	Integer flag (0 or 1) indicating whether to use movement priors. Default is 0 (priors not used).
`Movement_prior`	Optional data.frame providing informative priors for movement. Required columns are `region_from`, `age`, `sex`, and `alpha`, where `alpha` is a list-column and each element is a numeric vector of length `n_regions` containing prior concentration parameters for movement from the specified region. If `NULL` (default), no movement prior is used.
`Movement_ageblk_spec`	Only applicable for move_type = 0. Either: • Character string `"constant"` for age-invariant movement (default), or • A `list` of integer vectors specifying age blocks that share parameters. Example: `list(c(1:6), c(7:10), c(11:n_ages))` makes ages 1–6, 7–10, and 11–n_ages share parameters. To indicate full age invariance, use `"constant"` or `list(c(1:n_ages))`.
`Movement_yearblk_spec`	Only applicable for move_type = 0. Either: • Character string `"constant"` for time-invariant movement (default), or • A `list` of integer vectors specifying year blocks that share movement parameters.
`Movement_sexblk_spec`	Only applicable for move_type = 0. Either: • Character string `"constant"` for sex-invariant movement (default), or • A `list` of integer vectors specifying sex blocks that share movement parameters.
`cont_vary_movement`	Character string specifying continuous varying movement type. Available options:

	• `"none"` (no continuous variation) • `"iid_y"` (iid deviations by year) • `"iid_a"` (iid deviations by age) • `"iid_y_a"` (iid deviations by year and age) • `"iid_y_s"` (iid deviations by year and sex) • `"iid_a_s"` (iid deviations by age and sex) • `"iid_y_a_s"` (iid deviations by year, age, and sex) Default is `"none"`.
`Movement_cont_pe_pars_spec`	Character string specifying how process-error parameters for continuous-varying movement are shared or estimated. Available options: • `"est_shared_r"` – estimate shared across regions, • `"est_shared_a"` – estimate shared across ages, • `"est_shared_s"` – estimate shared across sexes, • `"est_shared_r_a"`, `"est_shared_a_s"`, `"est_shared_r_s"`, • `"est_shared_r_a_s"` – combinations of shared structure, • `"est_all"` – estimate all process-error parameters independently, • `"fix"` – treat process-error parameters as fixed (not estimated), • `"none"` – no process-error parameters (default). Default is `"none"`.
`ctmc_move_dat`	Data.frame with CTMC covariates used to build design matrices for diffusion and preference. Required columns (when `move_type == 1`) include `regions`, `years`, `ages`, and `sexes`, plus any covariates referenced in `diffusion_formula` and `preference_formula`. Can include projection years (years > n_yrs) with projected covariate values. Year effects in formulas (e.g., splines) are automatically capped at `n_yrs`
`adjacency_mat`	Square adjacency matrix (`n_regions` x `n_regions`) for CTMC movement (used when `move_type == 1`). Non-zero entries indicate allowed transitions between region pairs; zero entries indicate transitions that are not allowed.
`area_r`	Numeric vector of region areas of length `n_regions`. Required for CTMC movement to convert rates to per-area or per-distance values. Defaults to `rep(1, n_regions)`.
`diffusion_formula`	An R formula describing the linear predictor for diffusion rates in the CTMC model (required when `move_type == 1`). Variables used in the formula must be present as columns in `ctmc_move_dat`.
`preference_formula`	An R formula describing the linear predictor for preference in the CTMC model (required when `move_type == 1`). Variables used in the formula must be present as columns in `ctmc_move_dat`.
`ctmc_diffusion_bounds`	Numeric indicating whether diffusion bounds are placed to ensure that the generator matrix is Metzler. 0 == no bounds (default), 1 == bounds enforced, representing slipstream diffusion (i.e., residual preference gradients get added to diffusion to ensure Metzler matrix).

`...`	Additional named starting values that may be supplied. Typical names: `move_pars`, `move_devs`, `move_pe_pars`, `log_move_diffusion_pars`, `move_preference_pars`, etc. If not supplied, sensible defaults are created.

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps(` `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Rec` *Setup model objects for specifying recruitment module and associated processes*

Description

Setup model objects for specifying recruitment module and associated processes

Usage

```
Setup_Mod_Rec(
  input_list,
  rec_model,
  rec_dd = NULL,
  rec_lag = 1,
  Use_h_prior = 0,
  h_prior = NULL,
  Use_Rec_prop_Prior = 0,
  Rec_prop_prior = NULL,
  do_rec_bias_ramp = 0,
  bias_year = NA,
  max_bias_ramp_fct = 1,
  sigmaR_switch = 1,
  dont_est_recdev_last = 0,
  init_age_strc = 2,
  equil_init_age_strc = 1,
  init_F_prop = 0,
  sigmaR_spec = NULL,
  InitDevs_spec = NULL,
  RecDevs_spec = NULL,
  h_spec = NULL,
  t_spawn = 0,
  sexratio_spec = "fix",
  sexratio_blocks = c(paste("none_Region_", c(1:input_list$data$n_regions), sep = "")),
  ...
)
```

Arguments

`input_list`	List containing data, parameters, and map lists used by the model.
`rec_model`	Character string specifying the recruitment model. Options are: • `"mean_rec"`: Recruitment is a fixed mean value. • `"bh_rec"`: Beverton-Holt recruitment with steepness parameter.
`rec_dd`	Character string specifying recruitment density dependence, options: `"local"`, `"global"`, or `NULL`.
`rec_lag`	Integer specifying the recruitment lag duration relative to spawning stock biomass (SSB).
`Use_h_prior`	Integer flag (0 or 1) indicating whether to apply a prior on steepness `h`.
`h_prior`	Data frame specifying beta prior distributions for the ‘h_trans‘ parameters. Must include the following columns: - ‘region‘: Integer region index corresponding to the element in ‘h_trans‘ being penalized. - ‘mu‘: Mean of the prior in normal space (used to calculate the corresponding beta distribution). - ‘sd‘: Standard deviation of the prior in normal space. For each row, a beta distribution is scaled to the interval [0.2, 1], and the corresponding element of ‘h_trans‘ is transformed to that scale and penalized using the log-density from the beta distribution.
`Use_Rec_prop_Prior`	Integer flag (0 or 1) indicating whether to apply a prior on recruitment proportions.
`Rec_prop_prior`	Scalar or array specifying prior values for recruitment proportion parameters. If scalar, a constant uniform prior is applied across all dimensions.
`do_rec_bias_ramp`	Integer flag (0 or 1) indicating whether to apply a recruitment bias correction ramp.
`bias_year`	Numeric vector of length 4 defining the recruitment bias ramp periods: • Element 1: End year of no bias correction period. • Element 2: End year of ascending bias ramp period. • Element 3: End year of full bias correction period. • Element 4: Start year of final no bias correction period. For example, with 65 years total, `c(21, 31, 60, 64)` means: • Years 1–21: No bias correction. • Years 22–31: Ascending bias correction. • Years 32–60: Full bias correction. • Years 61–63: Descending bias ramp. • Years 64–65: No bias correction.
`max_bias_ramp_fct`	Numeric specifying the maximum bias correction to apply to the recruitment bias ramp (should be between 0 and 1)
`sigmaR_switch`	Integer year indicating when `sigmaR` switches from early to late values (0 disables switching).

`dont_est_recdev_last`	Integer specifying how many of the most recent recruitment deviations to not estimate. Default is 0.
`init_age_strc`	Integer flag specifying initialization of initial age structure: • `0`: Initialize by iteration. • `1`: Initialize using a scalar geometric series w/o any movement in all groups (does not account for movement in all group). • `2`: Initialize using a matrix geometric series (accounts for movement; default). • `3`: Initialize using a scalar geometric series w/o any movement in only plus groups (does not account for movement in plus group).
`equil_init_age_strc`	Integer flag specifying how initial age structure deviations should be initialized. Default is stochastic for all ages except the recruitment age and the plus group. • `0`: Equilibrium initial age structure. • `1`: Stochastic initial age structure for all ages, except for the plus group, which follows equilibrium calculations (geometric series) • `2`: Stochastic initial age structure for all ages
`init_F_prop`	Numeric value specifying the initial fishing mortality proportion relative to mean fishing mortality for initializing age structure.
`sigmaR_spec`	Character string specifying estimation of recruitment variability (`sigmaR`): • `NULL` or `"est_all"`: Estimate separate `sigmaR` for early and late periods. • `"est_shared"`: Estimate one `sigmaR` shared across periods. • `"fix"`: Fix both `sigmaR` values. • `"fix_early_est_late"`: Fix early `sigmaR`, estimate late `sigmaR`.
`InitDevs_spec`	Character string specifying estimation of initial age deviations: • `NULL`: Estimate deviations for all ages and regions. • `"est_shared_r"`: Estimate deviations shared across regions. • `"fix"`: Fix all deviations.
`RecDevs_spec`	Character string specifying recruitment deviation estimation: • `NULL`: Estimate deviations for all regions and years. • `"est_shared_r"`: Estimate deviations shared across regions (global recruitment deviations). • `"fix"`: Fix all recruitment deviations.
`h_spec`	Character string specifying steepness estimation: • `NULL`: Estimate steepness for all regions if `rec_model == "bh_rec"`. • `"est_shared_r"`: Estimate steepness shared across regions. • `"fix"`: Fix steepness values. If `rec_model == "mean_rec"`, steepness is fixed.
`t_spawn`	Numeric fraction specifying spawning timing within the year.
`sexratio_spec`	Character string specifying sex ratio estimation scheme:

- `"est_all"` estimates sex ratio for all blocks and regions independently
- `"est_shared_r"` estimates sex ratio shared across regions but varying by block
- `"fix"` fixes all sex ratio (no estimation)

`sexratio_blocks`

Character vector specifying blocks of years and regions for sex ratio. Format examples:

- `"Block_1_Year_1-15_Region_1"`
- `"Block_2_Year_16-terminal_Region_2"`
- `"none_Region_3"` (means no block, constant for that region; this is the default option)

`...` Additional arguments specifying starting values for recruitment parameters such as `ln_global_R0`, `Rec_prop`, `h`, `ln_InitDevs`, `ln_RecDevs`, and `ln_sigmaR`.

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_SrvIdx_and_Co` `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_SrvIdx_and_Comps`

Setup observed survey indices and composition data (age and length comps)

Description

Setup observed survey indices and composition data (age and length comps)

Usage

```
Setup_Mod_SrvIdx_and_Comps(
  input_list,
  ObsSrvIdx,
  ObsSrvIdx_SE,
  UseSrvIdx,
  srv_idx_type,
  ObsSrvAgeComps,
  UseSrvAgeComps,
  ObsSrvLenComps,
  UseSrvLenComps,
  ISS_SrvAgeComps,
  ISS_SrvLenComps,
  SrvAgeComps_LikeType,
  SrvLenComps_LikeType,
  SrvAgeComps_Type,
```

```
  SrvLenComps_Type,
  ...
)
```

Arguments

`input_list`	List containing a data list, parameter list, and map list
`ObsSrvIdx`	Observed survey index data as a numeric array with dimensions `[n_regions, n_years, n_srv_fleets]`.
`ObsSrvIdx_SE`	Standard errors associated with `ObsSrvIdx`, also dimensioned `[n_regions, n_years, n_srv_fleets]`.
`UseSrvIdx`	Logical or binary indicator array (`[n_regions, n_years, n_srv_fleets]`) specifying whether to include a survey index in the likelihood (`1`) or ignore it (`0`).
`srv_idx_type`	Character vector of length `n_srv_fleets` specifying the type of index data. Options are `"abd"` for abundance, `"biom"` for biomass, and `"none"` if no index is available.
`ObsSrvAgeComps`	Observed survey age composition data as a numeric array with dimensions `[n_regions, n_years, n_ages, n_sexes, n_srv_fleets]`. Values should reflect counts or proportions (not required to sum to 1, but should be on a comparable scale).
`UseSrvAgeComps`	Indicator array (`[n_regions, n_years, n_srv_fleets]`) specifying whether to fit survey age composition data (`1`) or ignore it (`0`).
`ObsSrvLenComps`	Observed survey length composition data as a numeric array with dimensions `[n_regions, n_years, n_lens, n_sexes, n_srv_fleets]`. Values should reflect counts or proportions.
`UseSrvLenComps`	Indicator array (`[n_regions, n_years, n_srv_fleets]`) specifying whether to fit survey length composition data (`1`) or ignore it (`0`).
`ISS_SrvAgeComps`	Input sample size for age compositions, array dimensioned `[n_regions, n_years, n_sexes, n_srv_fleets]`. Required if observed age comps are normalized (i.e., sum to 1), to correctly scale the contribution to the likelihood.
`ISS_SrvLenComps`	Same as `ISS_SrvAgeComps`, but for length compositions.
`SrvAgeComps_LikeType`	Character vector of length `n_srv_fleets` specifying the likelihood type used for survey age composition data. Options include `"Multinomial"`, `"Dirichlet-Multinomial"`, and `"iid-Logistic-Normal"`. Use `"none"` to omit the likelihood.
`SrvLenComps_LikeType`	Same as `SrvAgeComps_LikeType`, but for survey length composition data.
`SrvAgeComps_Type`	Character vector specifying how age compositions are structured by fleet and year range. Options include: • `"agg"`: Aggregated across regions and sexes. • `"spltRspltS"`: Split by region and by sex (compositions sum to 1 within region-sex group). • `"spltRjntS"`: Split by region but summed jointly across sexes.

- `"none"`: No composition data used.

Format each element as `"<type>_Year_<start>-<end>_Fleet_<fleet number>"` (e.g., `"agg_Year_1-10_Fleet_1"`).

`SrvLenComps_Type`	Same as `SrvAgeComps_Type`, but for length compositions.
`...`	Additional arguments specifying starting values for overdispersion parameters (e.g., `ln_SrvAge_theta`, `ln_SrvLen_theta`, `ln_SrvAge_theta_agg`, `ln_SrvLen_theta_agg`).

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Srvsel_and_Q`

Setup survey selectivity and catchability specifications

Description

Setup survey selectivity and catchability specifications

Usage

```
Setup_Mod_Srvsel_and_Q(
  input_list,
  cont_tv_srv_sel = paste("none_Fleet_", 1:input_list$data$n_srv_fleets, sep = ""),
  srv_sel_blocks = paste("none_Fleet_", 1:input_list$data$n_srv_fleets, sep = ""),
  srv_sel_model,
  Use_srv_q_prior = 0,
  srv_q_prior = NA,
  srv_q_blocks = paste("none_Fleet_", 1:input_list$data$n_srv_fleets, sep = ""),
  srvsel_pe_pars_spec = NULL,
  srv_fixed_sel_pars_spec,
  srv_q_spec = NULL,
  srv_sel_devs_spec = NULL,
  corr_opt_semipar = NULL,
  srv_q_formula = NULL,
  srv_q_cov_dat = NULL,
  Use_srv_selex_prior = 0,
  srv_selex_prior = NULL,
  t_srv = array(0.5, dim = c(input_list$data$n_regions, input_list$data$n_srv_fleets)),
  cont_tv_srv_sel_penalty = TRUE,
  srvsel_devs_shared_ages = NULL,
  ...
)
```

Arguments

`input_list` List containing a data list, parameter list, and map list

`cont_tv_srv_sel`

Character vector specifying the form of continuous time-varying selectivity for each survey fleet. The vector must be length `n_srv_fleets`, and each element must follow the structure: `"<time variation type>_Fleet_<fleet number>"`.

Valid time variation types include:

- `"none"`: No continuous time variation. (default)
- `"iid"`: Independent and identically distributed deviations across years.
- `"rw"`: Random walk in time.
- `"3dmarg"`: 3D marginal time-varying selectivity.
- `"3dcond"`: 3D conditional time-varying selectivity.
- `"2dar1"`: Two-dimensional AR1 process.

For example:

- `"iid_Fleet_1"` applies an iid time-varying structure to Fleet 1.
- `"none_Fleet_2"` means no time variation is used for Fleet 2.

`srv_sel_blocks` Character vector specifying the survey selectivity blocks for each region and fleet.

Each element must follow one of the following structures:

- ‘"Block_<block number>_Year_<start>-<end>_Fleet_<fleet number>"‘
- ‘"Block_<block number>_Year_<start>-terminal_Fleet_<fleet number>"‘
- ‘"none_Fleet_<fleet number>"‘

This argument defines how survey selectivity varies over time for each fleet:

- `"Block_..."` entries specify discrete time blocks during which selectivity parameters are assumed constant.
- `"none_..."` entries indicate that selectivity is constant across all years for the specified fleet.

If time-block-based selectivity is specified for a fleet (via `srv_sel_blocks`), its corresponding continuous selectivity option (in `cont_tv_srv_sel`) must be set to `"none_Fleet_<fleet number>"`. The two approaches—blocked and continuous time-varying selectivity—are mutually exclusive. The default for each fleet is `"none_Fleet_x"` (i.e., no selectivity blocks).

`srv_sel_model` Character vector specifying the survey selectivity functional form for each fleet, and optionally by time block.

Each element must follow one of the following structures:

- `"<selectivity model>_Fleet_<fleet number>"`
- `"<selectivity model>_Fleet_<fleet number>_Block_<block number>"`

The first form applies a single selectivity model across all years for the specified fleet. The second form allows the user to assign a distinct selectivity model to a specific time block, as defined in `srv_sel_blocks`.

Available selectivity model types include:

- `"logist1"` — Logistic function with parameters `a50` and `k`.

- `"logist2"` — Logistic function with parameters `a50` and `a95`.
- `"gamma"` — Dome-shaped gamma function with parameters `amax` and `delta`.
- `"exponential"` — Exponential function with a power parameter.
- `"dbnrml"` — Double-normal function with six parameters.

If multiple selectivity time blocks are specified for a fleet (using `srv_sel_blocks`), then the corresponding selectivity model for each block must be explicitly defined using the `"<model>_Block_<block>_Fleet_<fleet>"` format. If blocks are not defined for a fleet, use the `"<model>_Fleet_<fleet number>"` format only. For mathematical definitions and implementation details of each selectivity form, refer to the model equations vignette.

`Use_srv_q_prior`

Integer (0 or 1). Flag to enable/disable survey catchability priors. When set to 1, applies log-normal priors to survey selectivity parameters as specified in `srv_q_prior`. When set to 0, no priors are applied.

`srv_q_prior`

Data frame containing prior specifications for survey catchability parameters. Must include columns: `region` (region index), `fleet` (fleet index), `block` (time block index), `mu` (prior mean on natural scale), and `sd` (prior standard deviation on log scale). Each row specifies a log-normal prior N(log(mu), sd) for a given catchability parameter. Only parameters with rows in this data frame will have priors applied.

`srv_q_blocks`

Character vector specifying survey catchability (q) blocks for each fleet. Each element must follow the structure: `"Block_<block number>_Year_<start>-<end>_Fleet_<fleet number>"` or `"none_Fleet_<fleet number>"`.

This allows users to define time-varying catchability blocks independently of selectivity blocks. The blocks must be non-overlapping and sequential in time within each fleet.

For example:

- `"Block_1_Year_1-35_Fleet_1"` assigns block 1 to Fleet 1 for years 1–35.
- `"Block_2_Year_36-56_Fleet_1"` continues with block 2 for years 36–56.
- `"Block_3_Year_57-terminal_Fleet_1"` assigns block 3 from year 57 to the terminal year for Fleet 1.
- `"none_Fleet_2"` indicates no catchability blocks are used for Fleet 2.

Internally, these specifications are converted to a `[n_regions, n_years, n_srv_fleets]` array, where each block is mapped to the appropriate years and fleets.

`srvsel_pe_pars_spec`

Character string specifying how process error parameters for survey selectivity are estimated across regions and sexes. This is only relevant if `cont_tv_srv_sel` is not set to `"none"`; otherwise, all process error parameters are treated as fixed.

Available options include:

- `"est_all"`: Estimates separate process error parameters for each region and sex.
- `"est_shared_r"`: Shares process error parameters across regions (sex-specific parameters are still estimated).
- `"est_shared_s"`: Shares process error parameters across sexes (region-specific parameters are still estimated).

- `"est_shared_r_s"`: Shares process error parameters across both regions and sexes, estimating a single set of parameters.
- `"est_shared_f_x"`: Shares process error parameters with another fleet, where `x` is the fleet number to share with. This option forces multiple fleets to have identical process error variance and correlation structures for their time-varying selectivity. For example, `"est_shared_f_2"` means the current fleet will use the same process error parameters as fleet 2. The reference fleet (fleet x) must use one of the other sharing options and cannot itself be sharing with another fleet.
- `"fix"` or `"none"`: Does not estimate process error parameters; all are treated as fixed.

`srv_fixed_sel_pars_spec`

Character string specifying the structure for estimating fixed-effect parameters of the survey selectivity model (e.g., a50, k, amax). This controls whether selectivity parameters are estimated separately or shared across regions and sexes.

Available options include:

- `"est_all"`: Estimates separate fixed-effect selectivity parameters for each region and sex.
- `"est_shared_r"`: Shares parameters across regions (sex-specific parameters are still estimated).
- `"est_shared_s"`: Shares parameters across sexes (region-specific parameters are still estimated).
- `"est_shared_r_s"`: Shares parameters across both regions and sexes, estimating a single set of fixed-effect parameters.
- `"est_shared_f_x"`: Shares fixed-effect selectivity parameters with another fleet, where `x` is the fleet number to share with. This option forces multiple fleets to have identical selectivity curves by using the same underlying parameters (e.g., same a50, k, amax values). For example, `"est_shared_f_2"` means the current fleet will use the same fixed-effect selectivity parameters as fleet 2. The reference fleet (fleet x) must use one of the other sharing options and cannot itself be sharing with another fleet.
- `"fix"`: Fixes all selectivity parameters to their initial values (no estimation).
- `"none"`: No selectivity parameters are estimated (equivalent to `"fix"`).

`srv_q_spec`

Character string specifying the structure of survey catchability (`q`) estimation across regions. This controls whether separate or shared parameters are used.

Available options include:

- `"est_all"`: Estimates separate catchability parameters for each region.
- `"est_shared_r"`: Estimates a single catchability parameter shared across all regions.

`srv_sel_devs_spec`

Character string specifying the structure of process error deviations in time-varying survey selectivity dimensioned by the number of survey fleets. This determines how deviations are estimated across regions and sexes.

Available options include:

- `"est_all"`: Estimates a separate deviation time series for each region and sex.
- `"est_shared_r"`: Shares deviations across regions (sex-specific deviations are still estimated).
- `"est_shared_s"`: Shares deviations across sexes (region-specific deviations are still estimated).
- `"est_shared_r_s"`: Shares deviations across both regions and sexes, estimating a single deviation time series.
- `"est_shared_f_x"`: Shares deviations with another fleet, where `x` is the fleet number to share with. This option allows multiple fleets to use identical deviation parameters, reducing the number of parameters to estimate. For example, `"est_shared_f_2"` means the current fleet will use the same deviation parameters as fleet 2. The reference fleet (fleet x) must use one of the other sharing options (`"est_all"`, `"est_shared_r"`, `"est_shared_s"`, or `"est_shared_r_s"`) and cannot itself be sharing with another fleet.
- `"fix"`: Fixes all deviation parameters to zero (no time-variation).
- `"none"`: No deviation parameters are estimated (equivalent to `"fix"`).

This argument is only used when a continuous time-varying selectivity form is specified (e.g., via `cont_tv_srv_sel`).

`corr_opt_semipar`

Character string specifying which correlation structures to suppress when using semi-parametric time-varying selectivity models. Only used if `cont_tv_sel` is set to one of `"3dmarg"`, `"3dcond"`, or `"2dar1"`.

This option allows users to turn off estimation of specific correlation components in the time-varying selectivity model. This can improve stability or enforce assumptions about independence in the temporal or age structure.

Available options:

- `"corr_zero_y"`: Sets year (temporal) correlations to 0.
- `"corr_zero_b"`: Sets bin correlations to 0.
- `"corr_zero_y_b"`: Sets both year and bin correlations to 0.
- `"corr_zero_c"`: Sets cohort correlations to 0. Only valid for `cont_tv_sel` = `"3dmarg"` or `"3dcond"`.
- `"corr_zero_y_c"`: Sets year and cohort correlations to 0. Only valid for `cont_tv_sel` = `"3dmarg"` or `"3dcond"`.
- `"corr_zero_b_c"`: Sets bin (age) and cohort correlations to 0. Only valid for `cont_tv_sel` = `"3dmarg"` or `"3dcond"`.
- `"corr_zero_y_b_c"`: Sets all correlations (year, bin (age), and cohort) to 0. Only valid for `cont_tv_sel` = `"3dmarg"` or `"3dcond"`; equivalent to an iid structure.

These correlation-suppression flags are ignored when `cont_tv_sel` is set to any other value.

`srv_q_formula` A named list of formulas specifying environmental covariate relationships for each region and survey fleet. Each element should be named using the convention ‘"Region_<region>_Fleet_<fleet>"‘ and contain a formula object using covariate names present in ‘srv_q_cov_dat‘. The formula determines how environmental covariates influence survey catchability. If ‘NULL‘, no environmental covariate effects are included.

`srv_q_cov_dat` A named list containing time series vectors (typically by year) of environmental covariates used in the ‘srv_q_formula‘. Each entry should be a numeric vector of length equal to the number of years, and names must match the variable names used in the formulas. If ‘NULL‘, survey catchability is assumed to be time-invariant (i.e., not influenced by environmental variables).

`Use_srv_selex_prior`

Integer (0 or 1). Flag to enable/disable survey selectivity priors. When set to 1, applies log-normal priors to survey selectivity parameters as specified in `srv_selex_prior`. When set to 0, no priors are applied.

`srv_selex_prior`

Data frame containing prior specifications for survey selectivity parameters. Must include columns: `region` (region index), `fleet` (fleet index), `block` (time block index), `sex` (sex index), `par` (parameter index), `mu` (prior mean on natural scale), and `sd` (prior standard deviation on log scale). Each row specifies a log-normal prior N(log(mu), sd) for one selectivity parameter. Only parameters with rows in this data frame will have priors applied.

If both ‘srv_q_formula‘ and ‘srv_q_cov_dat‘ are non-‘NULL‘, the model constructs time-varying design matrices for each region and fleet based on the provided formulas and environmental covariates. A coefficient array (‘srv_q_coeff‘) and a mapping array (‘map_srv_q_coeff‘) are created to estimate and track the associated regression coefficients. The design matrix is stored in ‘srv_q_env‘, a 4D array indexed by `[region, year, fleet, covariate]`.

If either argument is ‘NULL‘, environmental covariate effects are excluded and survey catchability is treated as constant over time.

Important: All covariate time series in ‘srv_q_cov_dat‘ must:

- Be numeric vectors with a length equal to the number of years in the model.
- Align to the same years across all covariates.
- Contain no missing values; users must impute or interpolate missing covariate values prior to use. For years in which the index is not used, values can be set at 0.

Covariates that are defined but not used in any formula can be filled with zeros (e.g., `rep(0, n_yrs)`). This avoids issues with list structure but does not affect the design matrix or model results.

Example formulas:

- `"Region_1_Fleet_1" = ~ 0 + poly(env1_r1_f1, 3) + env2_r1_f1` uses a 3rd-degree polynomial for `env1_r1_f1` and a linear term for `env2_r1_f1`.
- `"Region_2_Fleet_1" = ~ 0 + env1_r2_f1 + env2_r2_f1` includes additive effects of two covariates.
- `"Region_3_Fleet_2" = ~ NULL` disables environmental covariates for that fleet-region.

`t_srv` Survey timing in fractions (n_regions * n_srv_fleets; default is 0.5)

`cont_tv_srv_sel_penalty`	Whether or not to apply continuous time-varying selectivity penalties (if cont_tv_srv_sel > 0)
`srvsel_devs_shared_ages`	List object for specifying which ages are shared when selectivity deviations are semi-parametric (e.g., list(1:5, 6:10, 11:30) specifies that ages 1-5, 6-10, and 11-30 have the same deviations.)
`...`	Additional arguments specifying starting values for survey selectivity and catchability parameters (srvsel_pe_pars, ln_srvsel_devs, ln_srv_fixed_sel_pars, ln_srv_q, srv_q_coeff)

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Tagging()`, `Setup_Mod_Weighting()`

`Setup_Mod_Tagging` *Setup tagging processes and parameters*

Description

Setup tagging processes and parameters

Usage

```
Setup_Mod_Tagging(
  input_list,
  UseTagging = 0,
  tag_release_indicator = NULL,
  max_tag_liberty = 0,
  Tagged_Fish = NA,
  Obs_Tag_Recap = NA,
  Tag_LikeType = NA,
  mixing_period = 1,
  t_tagging = 0,
  tag_selex = NA,
  tag_natmort = NA,
  Use_TagRep_Prior = 0,
  TagRep_Prior = NULL,
  move_age_tag_pool = as.list(1:length(input_list$data$ages)),
  move_sex_tag_pool = as.list(1:input_list$data$n_sexes),
  Init_Tag_Mort_spec = NULL,
  Tag_Shed_spec = NULL,
  TagRep_spec = "fix",
  Tag_Reporting_blocks = NULL,
  ...
)
```

Arguments

`input_list`	List containing a data list, parameter list, and map list
`UseTagging`	Numeric (0 or 1) indicating whether to use tagging data (1) or not (0)
`tag_release_indicator`	Matrix [n_tag_cohorts x 2], where columns are release region and release year
`max_tag_liberty`	Maximum number of years to track a tagged cohort
`Tagged_Fish`	Array [n_tag_cohorts x n_ages x n_sexes] describing tagged fish releases
`Obs_Tag_Recap`	Array [max_tag_liberty x n_tag_cohorts x n_regions x n_ages x n_sexes] observed tag recaptures
`Tag_LikeType`	Character string specifying tag likelihood type. One of: • `"Poisson"` • `"NegBin"` • `"Multinomial_Release"` • `"Multinomial_Recapture"` • `"Dirichlet-Multinomial_Release"` • `"Dirichlet-Multinomial_Recapture"` Example: `Tag_LikeType = "NegBin"`
`mixing_period`	Numeric indicating minimum years post-release to include in fitting
`t_tagging`	Fractional year when tagging occurs (e.g., 0.5 for mid-year)
`tag_selex`	Character string specifying tag recovery selectivity. One of: • `"Uniform_DomFleet"` • `"SexAgg_DomFleet"` • `"SexSp_DomFleet"` • `"Uniform_AllFleet"` • `"SexAgg_AllFleet"` • `"SexSp_AllFleet"` Example: `tag_selex = "SexSp_AllFleet"`
`tag_natmort`	Character string specifying tag natural mortality parameterization. One of: • `"AgeAgg_SexAgg"` • `"AgeSp_SexAgg"` • `"AgeAgg_SexSp"` • `"AgeSp_SexSp"` Example: `tag_natmort = "AgeSp_SexSp"`
`Use_TagRep_Prior`	Numeric (0 or 1) whether to use tag reporting rate prior
`TagRep_Prior`	Data frame containing prior specifications for tag reporting parameters. Must include columns: `region` (region index), `block` (time block index), `mu` (Numeric mean for tag reporting prior (normal space); `NA` if symmetric beta is used), `sd` (Numeric standard deviation for tag reporting prior (normal space)), and `type` (0 == symmetric beta, 1 == regular beta). Each row specifies a beta prior for one tag reporting parameter. Only parameters with rows in this data frame will have priors applied.

`move_age_tag_pool`	List or character specifying pooling of tagging data by age groups. Default does not pool ages. Examples: • `list(1:5, 6:11, 12:20)` pools these age groups together • `"all"` pools all ages together (internally converted to `list(1:n_ages)`) • `as.list(1:n_ages)` fits each sex separately
`move_sex_tag_pool`	List or character specifying pooling of tagging data by sex groups. Default do not pool sexes. Examples: • `list(1:2)` pools sexes together • `"all"` pools all sexes together (internally converted to `list(1:n_sexes)`) • `list(1, 2)` fits each sex separately
`Init_Tag_Mort_spec`	Character string `"fix"` or `"est"` specifying if initial tag mortality is fixed or estimated
`Tag_Shed_spec`	Character string `"fix"` or `"est"` specifying if chronic tag shedding is fixed or estimated
`TagRep_spec`	Character string specifying tag reporting rate estimation scheme: • `"est_all"` estimates rates for all blocks and regions independently • `"est_shared_r"` estimates rates shared across regions but varying by block • `"fix"` fixes all reporting rates (no estimation)
`Tag_Reporting_blocks`	Character vector specifying blocks of years and regions for tag reporting rates. Default is a single block for all regions. Format examples: • `"Block_1_Year_1-15_Region_1"` • `"Block_2_Year_16-terminal_Region_2"` • `"none_Region_3"` (means no block, constant for that region)
`...`	Additional starting values for tagging parameters such as `ln_Init_Tag_Mort`, `ln_Tag_Shed`, `ln_tag_theta`, `Tag_Reporting_Pars`

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Weighting()`

`Setup_Mod_Weighting` *Set up SPoRC model weighting*

Description

Set up SPoRC model weighting

Usage

```
Setup_Mod_Weighting(
  input_list,
  Wt_Catch = 1,
  Wt_FishIdx = 1,
  Wt_SrvIdx = 1,
  Wt_Rec = 1,
  Wt_F = 1,
  Wt_Tagging = 1,
  Wt_FishAgeComps,
  Wt_SrvAgeComps,
  Wt_FishLenComps,
  Wt_SrvLenComps
)
```

Arguments

`input_list`	List containing data, parameter, and map lists.
`Wt_Catch`	Either a numeric scalar (lambda) applied to the overall catch dataset or an array of lambdas (i.e., weights can change by year and fleet) dimensioned by n_regions, n_years, n_fish_fleets.
`Wt_FishIdx`	Either a numeric scalar (lambda) applied to the overall fishery index dataset or an array of lambdas (i.e., weights can change by year and fleet) dimensioned by n_regions, n_years, n_fish_fleets.
`Wt_SrvIdx`	Either a numeric scalar (lambda) applied to the overall survey index dataset or an array of lambdas (i.e., weights can change by year and fleet) dimensioned by n_regions, n_years, n_srv_fleets.
`Wt_Rec`	Numeric weight (lambda) applied to the recruitment penalty.
`Wt_F`	Numeric weight (lambda) applied to the fishing mortality penalty.
`Wt_Tagging`	Numeric weight (lambda) applied to tagging data.
`Wt_FishAgeComps`	Numeric weight (lambda) applied to fishery age composition data.
`Wt_SrvAgeComps`	Numeric weight (lambda) applied to survey age composition data.
`Wt_FishLenComps`	Numeric weight (lambda) applied to fishery length composition data.
`Wt_SrvLenComps`	Numeric weight (lambda) applied to survey length composition data.

See Also

Other Model Setup: `Setup_Mod_Biologicals()`, `Setup_Mod_Catch_and_F()`, `Setup_Mod_Dim()`, `Setup_Mod_FishIdx_and_Comps()`, `Setup_Mod_Fishsel_and_Q()`, `Setup_Mod_Movement()`, `Setup_Mod_Rec()`, `Setup_Mod_SrvIdx_and_Comps()`, `Setup_Mod_Srvsel_and_Q()`, `Setup_Mod_Tagging()`

`Setup_Sim_Biologicals` *Set up simulation containers and inputs for biological parameters*

Description

Set up simulation containers and inputs for biological parameters

Usage

```
Setup_Sim_Biologicals(
  natmort_input,
  WAA_input,
  WAA_fish_input,
  WAA_srv_input,
  MatAA_input,
  AgeingError_input = NULL,
  SizeAgeTrans_input = NULL,
  sim_list
)
```

Arguments

`natmort_input`	Natural mortality array [n_regions × n_yrs × n_ages × n_sexes × n_sims]
`WAA_input`	Spawning weight-at-age array [n_regions × n_yrs × n_ages × n_sexes × n_sims]
`WAA_fish_input`	Fishery weight-at-age array [n_regions × n_yrs × n_ages × n_sexes × n_sims]
`WAA_srv_input`	Survey weight-at-age array [n_regions × n_yrs × n_ages × n_sexes × n_sims]
`MatAA_input`	Maturity-at-age array [n_regions × n_yrs × n_ages × n_sexes × n_sims]
`AgeingError_input`	Ageing error matrix [n_regions × n_model_ages × n_obs_ages × n_sims]
`SizeAgeTrans_input`	Size-age transition matrix [n_regions × n_yrs × n_lens × n_ages × n_sexes x n_sims]
`sim_list`	Simulation list object from ‘Setup_Sim_Dim()‘

See Also

Other Simulation Setup: `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`Setup_Sim_Containers` *Setup containers for simulation and output*

Description

Setup containers for simulation and output

Usage

```
Setup_Sim_Containers(sim_list)
```

Arguments

`sim_list`	List from ‘Setup_Sim_Dim()‘ containing core simulation dimensions (‘n_regions‘, ‘n_yrs‘, ‘n_ages‘, ‘n_sexes‘, ‘n_sims‘, ‘n_fish_fleets‘, ‘n_srv_fleets‘, ‘n_obs_ages‘, ‘n_lens‘). The function appends container arrays for biological, fishery, and survey quantities.

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`Setup_Sim_Dim` *Initialize Simulation Dimension Settings*

Description

Creates and returns a list of key dimension values used to set up a simulation or management strategy evaluation (MSE). This list provides structural information such as number of simulations, years, regions, ages, fleets, and whether to include a feedback loop.

Usage

```
Setup_Sim_Dim(
  n_sims,
  n_yrs,
  n_regions,
  n_ages,
  n_lens,
  n_obs_ages = n_ages,
  n_sexes,
  n_fish_fleets,
  n_srv_fleets,
  run_feedback = FALSE,
  feedback_start_yr = NULL
)
```

Arguments

`n_sims`	Integer. Number of simulation replicates.
`n_yrs`	Integer. Number of years in the simulation.
`n_regions`	Integer. Number of modeled regions.
`n_ages`	Integer. Number of modeled age classes.
`n_lens`	Integer. Number of modeled length bins.
`n_obs_ages`	Integer. Number of observed age classes (can differ from `n_ages`, default = `n_ages`).
`n_sexes`	Integer. Number of sexes.
`n_fish_fleets`	Integer. Number of fishery fleets.
`n_srv_fleets`	Integer. Number of survey fleets.
`run_feedback`	Logical. Whether to include a feedback management loop (default = `FALSE`).
`feedback_start_yr`	Integer. First year that feedback is applied (only used if `run_feedback` = `TRUE`).

Value

A list containing the specified dimension values, with elements:

- `n_sims`, `n_yrs`, `n_regions`, `n_ages`, `n_lens`, `n_obs_ages`, `n_sexes`, `n_fish_fleets`, `n_srv_fleets`
- `init_iter` (set internally to `n_ages * 10`)
- `feedback_start_yr`, `run_feedback`

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`Setup_sim_env` *Constructs simulation objects in a new simulation environment for use in simulation functions*

Description

Constructs simulation objects in a new simulation environment for use in simulation functions

Usage

```
Setup_sim_env(sim_list)
```

Arguments

`sim_list`	Simulation list objects

Value

A new simulation environment with objects from sim_list

See Also

Other Simulation Setup: `Setup_Sim_Biologicals`(), `Setup_Sim_Containers`(), `Setup_Sim_Dim`(), `Setup_Sim_Fishing`(), `Setup_Sim_Rec`(), `Setup_Sim_Survey`(), `Setup_Sim_Tagging`(), `Simulate_Pop_Static`(), `run_annual_cycle`(), `simulation_data_to_SPoRC`(), `simulation_self_test`()

Examples

```
## Not run:
sim_env <- Setup_sim_env(sim_list)

## End(Not run)
```

`Setup_Sim_Fishing` *Setup values and dimensions of fishing processes*

Description

Setup values and dimensions of fishing processes

Usage

```
Setup_Sim_Fishing(
  sim_list,
  ln_sigmaC = array(log(0.02), dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_fish_fleets)),
  catch_units = array(1, dim = c(sim_list$n_regions, sim_list$n_fish_fleets)),
  init_F_val = 0,
  Fmort_input = array(0.1, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_fish_fleets, sim_list$n_sims)),
  fish_sel_input,
  fish_q_input = array(1, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_fish_fleets, sim_list$n_sims)),
  ObsFishIdx_SE = array(0.2, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_fish_fleets)),
 fish_idx_type = array(1, dim = c(sim_list$n_regions, sim_list$n_fish_fleets)),
  comp_fishage_like = rep(0, sim_list$n_fish_fleets),
  ISS_FishAgeComps = array(100, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_sexes, sim_list$n_fish_fleets, sim_list$n_sims)),
 ln_FishAge_theta = array(log(1), dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_fish_fleets)),
  ln_FishAge_theta_agg = rep(log(1), sim_list$n_fish_fleets),
  FishAge_corr_pars_agg = rep(0.01, sim_list$n_fish_fleets),
  FishAge_corr_pars = array(0.01, dim = c(sim_list$n_regions, sim_list$n_sexes,
```

```
    sim_list$n_fish_fleets, 2)),
  FishAgeComps_Type = array(2, dim = c(sim_list$n_yrs, sim_list$n_fish_fleets)),
  comp_fishlen_like = rep(0, sim_list$n_fish_fleets),
  ISS_FishLenComps = array(100, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_sexes, sim_list$n_fish_fleets, sim_list$n_sims)),
  ln_FishLen_theta = array(log(1), dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_fish_fleets)),
  ln_FishLen_theta_agg = rep(log(1), sim_list$n_fish_fleets),
  FishLen_corr_pars_agg = rep(0.01, sim_list$n_fish_fleets),
  FishLen_corr_pars = array(0.01, dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_fish_fleets, 2)),
  FishLenComps_Type = array(2, dim = c(sim_list$n_yrs, sim_list$n_fish_fleets))
)
```

Arguments

`sim_list`	Simulation list object from ‘Setup_Sim_Dim()‘
`ln_sigmaC`	Observation error for catch [n_regions × n_yrs × n_fish_fleets] (default: ‘log(0.02)‘)
`catch_units`	Units of catch - Array [n_regions × n_fish_fleets] • `0`: Abundance • `1`: Biomass (default)
`init_F_val`	Initial fishing mortality value (default: ‘0‘)
`Fmort_input`	Fishing mortality input array [n_regions × n_yrs × n_fish_fleets × n_sims] (default: ‘0.1‘)
`fish_sel_input`	Fishery selectivity array [n_regions × n_yrs × n_ages × n_sexes × n_fish_fleets × n_sims] (no default, must be provided)
`fish_q_input`	Fishery catchability array [n_regions × n_yrs × n_fish_fleets × n_sims] (default: ‘1‘)
`ObsFishIdx_SE`	Observation error of fishery index [n_regions × n_yrs × n_fish_fleets] (default: ‘0.2‘)
`fish_idx_type`	Array of index types [n_regions x n_fish_fleets] (default: all ‘1‘ = biomass index) • `0`: Abundance index • `1`: Biomass index
`comp_fishage_like`	Vector [n_fish_fleets] specifying likelihood for simulating age comps (default: all ‘0‘ = multinomial) • `0`: Multinomial • `1`: Dirichlet-Multinomial • `2`: Logistic Normal iid • `3`: Logistic Normal 1dar1 • `4`: Logistic Normal 2d correlation (constant by sex, 1dar1 by age)
`ISS_FishAgeComps`	Input sample sizes [n_regions × n_yrs × n_sexes × n_fish_fleets × n_sims] (default: ‘100‘)

`ln_FishAge_theta`
Overdispersion parameters [n_regions × n_sexes × n_fish_fleets] (default: ‘log(1)‘)

`ln_FishAge_theta_agg`
Overdispersion parameters for aggregated comps [n_fish_fleets] (default: ‘log(1)‘)

`FishAge_corr_pars_agg`
Correlation parameters (agg.) for options 3–4 [n_fish_fleets] (default: ‘0.01‘)

`FishAge_corr_pars`
Correlation parameters [n_regions × n_sexes × n_fish_fleets x 2] (default: ‘0.01‘)

`FishAgeComps_Type`
Array [n_yrs × n_fish_fleets] (default: ‘2‘ = joint by sex, split by region)

- `0`: Aggregated
- `1`: Split by sex and region
- `2`: Joint by sex, split by region
- `999`: Not simulated

`comp_fishlen_like`
Vector [n_fish_fleets] specifying likelihood for simulating length comps (default: all ‘0‘ = multinomial)

- `0`: Multinomial
- `1`: Dirichlet-Multinomial
- `2`: Logistic Normal iid
- `3`: Logistic Normal 1dar1
- `4`: Logistic Normal 2d correlation (constant by sex, 1dar1 by length)

`ISS_FishLenComps`
Input sample sizes [n_regions × n_yrs × n_sexes × n_fish_fleets × n_sims] (default: ‘100‘)

`ln_FishLen_theta`
Overdispersion parameters [n_regions × n_sexes × n_fish_fleets x 2] (default: ‘log(1)‘)

`ln_FishLen_theta_agg`
Overdispersion parameters for aggregated comps [n_fish_fleets] (default: ‘log(1)‘)

`FishLen_corr_pars_agg`
Correlation parameters (agg.) for options 3–4 [n_fish_fleets] (default: ‘0.01‘)

`FishLen_corr_pars`
Correlation parameters [n_regions × n_sexes × n_fish_fleets] (default: ‘0.01‘)

`FishLenComps_Type`
Array [n_yrs × n_fish_fleets] (default: ‘2‘ = joint by sex, split by region)

- `0`: Aggregated
- `1`: Split by sex and region
- `2`: Joint by sex, split by region
- `999`: Not simulated

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`Setup_Sim_Rec` *Set up recruitment dynamics for simulation*

Description

Set up recruitment dynamics for simulation

Usage

```
Setup_Sim_Rec(
  do_recruits_move = 0,
  sexratio_input = array(if (sim_list$n_sexes == 1) 1 else 0.5, dim =
    c(sim_list$n_regions, sim_list$n_yrs, sim_list$n_sexes, sim_list$n_sims)),
 R0_input = array(10, dim = c(sim_list$n_regions, sim_list$n_yrs, sim_list$n_sims)),
 h_input = array(0.8, dim = c(sim_list$n_regions, sim_list$n_yrs, sim_list$n_sims)),
  ln_sigmaR = log(c(1, 1)),
  recruitment_opt = "bh_rec",
  rec_dd = "global",
  init_dd = "global",
  sim_list,
  init_age_strc = 2,
  t_spawn = 0,
  rec_lag = 1,
  Rec_input = NULL,
  ln_InitDevs_input = NULL
)
```

Arguments

`do_recruits_move`	Indicator for whether recruits move (default = 0): • `0`: No movement • `1`: Move
`sexratio_input`	Sex ratio array [n_regions × n_yrs × n_sexes × n_sims] (default = 1 if one sex, else 0.5 for each sex)
`R0_input`	Unfished recruitment (R0) array [n_regions × n_yrs × n_sims] (default = 10)
`h_input`	Steepness array [n_regions × n_yrs × n_sims] (default = 0.8)
`ln_sigmaR`	Logarithmic standard deviation of recruitment [2]: 1st = sigma for initial devs, 2nd = sigma for latter devs (default = log(c(1, 1)))
`recruitment_opt`	Recruitment type (default = "bh_rec"): • `"mean_rec"`: Mean recruitment • `"bh_rec"`: Beverton-Holt recruitment

	• `"resample_from_input"`: Resampling recruitment years from ‘Rec_input‘ and preserves covariance of recruitment among regions if spatially-explicit values are provided
`rec_dd`	Recruitment density dependence (default = "global"): • `"global"`: Shared across regions • `"local"`: Region-specific
`init_dd`	Initial age density dependence (default = "global"): • `"global"`: Shared across regions • `"local"`: Region-specific
`sim_list`	Simulation list object from ‘Setup_Sim_Dim()‘
`init_age_strc`	Integer specifying the initialization method for the age structure: - 0: Iterative solution to equilibrium - 1: Scalar geometric series solution w/o movement in any groups (no movement in all groups) - 2: Matrix geometric series solution (generalizes scalar solution with movement) - 3: Scalar geometric series solution w/o movement only in plus group (no movement in plus groups)
`t_spawn`	Spawn timing fraction of the year (scalar, default = 0)
`rec_lag`	Recruitment lag (default = 1)
`Rec_input`	Recruitment array [n_regions × n_yrs × n_sims] (default = NULL)
`ln_InitDevs_input`	Initial deviations [n_regions × (n_ages-1) × n_sims] (default = NULL)

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`Setup_Sim_Survey` *Setup values for survey parameterization*

Description

Setup values for survey parameterization

Usage

```
Setup_Sim_Survey(
  ObsSrvIdx_SE = array(0.2, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_srv_fleets)),
  sim_list,
  srv_sel_input,
  srv_q_input = array(1, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_srv_fleets, sim_list$n_sims)),
  t_srv = array(0, dim = c(sim_list$n_regions, sim_list$n_srv_fleets)),
```

```
  srv_idx_type = array(1, dim = c(sim_list$n_regions, sim_list$n_srv_fleets)),
  comp_srvage_like = rep(0, sim_list$n_srv_fleets),
  ISS_SrvAgeComps = array(100, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_sexes, sim_list$n_srv_fleets, sim_list$n_sims)),
  ln_SrvAge_theta = array(log(1), dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_srv_fleets)),
  ln_SrvAge_theta_agg = rep(log(1), sim_list$n_srv_fleets),
  SrvAge_corr_pars_agg = rep(0.01, sim_list$n_srv_fleets),
  SrvAge_corr_pars = array(0.01, dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_srv_fleets, 2)),
  SrvAgeComps_Type = array(2, dim = c(sim_list$n_yrs, sim_list$n_srv_fleets)),
  comp_srvlen_like = rep(0, sim_list$n_srv_fleets),
  ISS_SrvLenComps = array(100, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_sexes, sim_list$n_srv_fleets, sim_list$n_sims)),
  ln_SrvLen_theta = array(log(1), dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_srv_fleets)),
  ln_SrvLen_theta_agg = rep(log(1), sim_list$n_srv_fleets),
  SrvLen_corr_pars_agg = rep(0.01, sim_list$n_srv_fleets),
  SrvLen_corr_pars = array(0.01, dim = c(sim_list$n_regions, sim_list$n_sexes,
    sim_list$n_srv_fleets, 2)),
  SrvLenComps_Type = array(2, dim = c(sim_list$n_yrs, sim_list$n_srv_fleets))
)
```

Arguments

`ObsSrvIdx_SE`	Survey index observation error [n_regions × n_yrs × n_srv_fleets] (default: ‘0.2‘)
`sim_list`	Simulation list object from ‘Setup_Sim_Dim()‘
`srv_sel_input`	Survey selectivity array [n_regions × n_yrs × n_ages × n_sexes × n_srv_fleets × n_sims] (no default, must be provided)
`srv_q_input`	Survey catchability array [n_regions × n_yrs × n_srv_fleets × n_sims] (default: ‘1‘)
`t_srv`	Survey timing fraction [n_regions × n_srv_fleets] (default: ‘0‘)
`srv_idx_type`	Array of index types [n_regions x n_srv_fleets] (default: all ‘1‘ = biomass index) • `0`: Abundance index • `1`: Biomass index
`comp_srvage_like`	Vector [n_srv_fleets] specifying likelihood for simulating age comps (default: all ‘0‘ = multinomial) • `0`: Multinomial • `1`: Dirichlet-Multinomial • `2`: Logistic Normal iid • `3`: Logistic Normal 1dar1 • `4`: Logistic Normal 2d correlation (constant by sex, 1dar1 by age)
`ISS_SrvAgeComps`	Input sample sizes [n_regions × n_yrs × n_sexes × n_srv_fleets × n_sims] (default: ‘100‘)

`ln_SrvAge_theta`

Overdispersion parameters [n_regions × n_sexes × n_srv_fleets] (default: ‘log(1)‘)

`ln_SrvAge_theta_agg`

Overdispersion parameters for aggregated comps [n_srv_fleets] (default: ‘log(1)‘)

`SrvAge_corr_pars_agg`

Correlation parameters (agg.) for options 3–4 [n_srv_fleets] (default: ‘0.01‘)

`SrvAge_corr_pars`

Correlation parameters [n_regions × n_sexes × n_srv_fleets x 2] (default: ‘0.01‘)

`SrvAgeComps_Type`

Array [n_yrs × n_srv_fleets] (default: ‘2‘ = joint by sex, split by region)

- `0`: Aggregated
- `1`: Split by sex and region
- `2`: Joint by sex, split by region
- `999`: Not simulated

`comp_srvlen_like`

Vector [n_srv_fleets] specifying likelihood for simulating length comps (default: all ‘0‘ = multinomial)

- `0`: Multinomial
- `1`: Dirichlet-Multinomial
- `2`: Logistic Normal iid
- `3`: Logistic Normal 1dar1
- `4`: Logistic Normal 2d correlation (constant by sex, 1dar1 by length)

`ISS_SrvLenComps`

Input sample sizes [n_regions × n_yrs × n_sexes × n_srv_fleets × n_sims] (default: ‘100‘)

`ln_SrvLen_theta`

Overdispersion parameters [n_regions × n_sexes × n_srv_fleets] (default: ‘log(1)‘)

`ln_SrvLen_theta_agg`

Overdispersion parameters for aggregated comps [n_srv_fleets] (default: ‘log(1)‘)

`SrvLen_corr_pars_agg`

Correlation parameters (agg.) for options 3–4 [n_srv_fleets] (default: ‘0.01‘)

`SrvLen_corr_pars`

Correlation parameters [n_regions × n_sexes × n_srv_fleets x 2] (default: ‘0.01‘)

`SrvLenComps_Type`

Array [n_yrs × n_srv_fleets] (default: ‘2‘ = joint by sex, split by region)

- `0`: Aggregated
- `1`: Split by sex and region
- `2`: Joint by sex, split by region
- `999`: Not simulated

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`Setup_Sim_Tagging` *Set up simulated tagging dynamics*

Description

Set up simulated tagging dynamics

Usage

```
Setup_Sim_Tagging(
  n_tags = NULL,
  n_tags_rel_input = NULL,
  UseTagging = 0,
  max_liberty = sim_list$n_ages/2,
  tag_release_indicator = expand.grid(regions = 1:sim_list$n_regions, tag_years =
    1:sim_list$n_yrs),
  t_tagging = 0,
  ln_Init_Tag_Mort = log(1e-05),
  ln_Tag_Shed = log(1e-05),
  Tag_Reporting_input = array(0.5, dim = c(sim_list$n_regions, sim_list$n_yrs,
    sim_list$n_sims)),
  tag_selex = 5,
  tag_natmort = 3,
  tag_like = 0,
  ln_tag_theta = log(1),
  sim_list
)
```

Arguments

`n_tags`	Number of tags to release in a given year (scalar, default = NULL)
`n_tags_rel_input`	Number of tag releases by tag cohort length (default = NULL)
`UseTagging`	Boolean to use tagging (default = 0): • 0: Do not simulate tagging • 1: Simulate tagging
`max_liberty`	Maximum liberty to track cohorts (default = sim_list$n_ages / 2)
`tag_release_indicator`	Tag release indicator [regions × tag_years] (default = all combinations of regions × years via ‘expand.grid‘)
`t_tagging`	Time of tagging (e.g., start year == 0, mid year == 0.5; default = 0)
`ln_Init_Tag_Mort`	Log initial tag-induced mortality (default = log(1e-5))
`ln_Tag_Shed`	Log chronic tag shedding rate (default = log(1e-5))

`Tag_Reporting_input`	Tag reporting input [n_regions × n_yrs × n_sims] (default = 0.5)
`tag_selex`	Tag selectivity type (integer, default = 5): • 0: Uniform_DomFleet • 1: SexAgg_DomFleet • 2: SexSp_DomFleet • 3: Uniform_AllFleet • 4: SexAgg_AllFleet • 5: SexSp_AllFleet
`tag_natmort`	Tag natural mortality type (integer, default = 3): • 0: AgeAgg_SexAgg • 1: AgeSp_SexAgg • 2: AgeAgg_SexSp • 3: AgeSp_SexSp
`tag_like`	Tag likelihood type (integer, default = 0): • 0: Poisson • 1: NegBin • 2: Multinomial_Release • 3: Multinomial_Recapture • 4: Dirichlet-Multinomial_Release • 5: Dirichlet-Multinomial_Recapture
`ln_tag_theta`	Scalar in log space describing tag likelihood overdispersion (default = log(1))
`sim_list`	Simulation list (required)

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`sgl_rg_dusky_data` *Dusky data for single region assessment case study*

Description

A data list containing inputs for the 2024 GOA Dusky Rockfish Assessment

Usage

```
sgl_rg_dusky_data
```

Format

Data list for single region dusky rockfish assessment

Source

Omori, K. L., Williams, B. C., Hulson, P.-J., Ferriss, B. 2024. Assessment of the dusky rockfish stock in the Gulf of Alaska. North Pacific Fishery Management Council, Anchorage, AK.

`sgl_rg_ebswp_data` *EBS Walleye Pollock data for single region case study*

Description

A dataset containing the necessary elements for the EBS Walleye Pollock case study.

Usage

`sgl_rg_ebswp_data`

Format

A list with multiple components needed for the single region walleye pollock model

Source

2024 Federal EBS Walleye Pollock Assessment

`sgl_rg_sable_data` *Sablefish data for single region case study*

Description

A dataset containing the necessary elements for the Alaska sablefish case study.

Usage

`sgl_rg_sable_data`

Format

A list with multiple components needed for the single region sablefish model

Source

2024 Federal Alaska Sablefish Assessment

`sgl_rg_sable_rep` *Sablefish report for single region case study*

Description

A report containing results for the Alaska sablefish case study.

Usage

```
sgl_rg_sable_rep
```

Format

Report file from the single region sablefish case study

Source

2024 Federal Alaska Sablefish Assessment

`Simulate_Pop_Static` *Simulates a static spatial, sex, and age-structured population (no feedback loop)*

Description

Simulates a static spatial, sex, and age-structured population (no feedback loop)

Usage

```
Simulate_Pop_Static(sim_list, output_path = NULL)
```

Arguments

`sim_list`	Simulation list objects
`output_path`	path to output simulation objects

Value

a list object with a bunch of simulated values and outputs

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`, `simulation_self_test()`

`simulation_data_to_SPoRC`

Extract simulation data into SPoRC format

Description

This function subsets and reshapes biological, tagging, fishery, and survey data from a simulation environment for use in SPoRC analyses.

Usage

```
simulation_data_to_SPoRC(sim_env, y, sim)
```

Arguments

`sim_env`	A simulation environment / object (list or environment) containing arrays of biological quantities, tagging information, fishery data, and survey data.
`y`	Integer. Number of years to retain (subset from ‘1:y‘).
`sim`	Integer. Simulation replicate index to extract.

Value

A named list with the following elements:

WAA Weight-at-age array [region × year × age × sex].

MatAA Maturity-at-age array [region × year × age × sex].

SizeAgeTrans Size–age transition array [region × year × length × age × sex].

AgeingError Ageing error matrix [year × age × error × sim].

tag_release_indicator Tag release indicators (or ‘NULL‘ if tagging not used).

Obs_Tag_Recap Observed tag recapture array (or ‘NULL‘).

Tagged_Fish Tagged fish counts (or ‘NULL‘).

n_tag_cohorts Number of tag release cohorts (or ‘NULL‘).

ObsCatch Observed fishery catch array [region × year × fleet].

ln_sigmaC Log Fishery Catch SD [region × year × fleet].

UseCatch Binary indicator array for catch data availability.

ObsFishIdx Observed fishery index array [region × year × fleet].

ObsFishIdx_SE Standard error for fishery index array.

UseFishIdx Binary indicator array for fishery indices.

ObsFishAgeComps Observed fishery age composition array.

ObsFishLenComps Observed fishery length composition array.

ISS_FishAgeComps Implied sample sizes for fishery age compositions.

ISS_FishLenComps Implied sample sizes for fishery length compositions.

UseFishAgeComps Binary indicator array for fishery age comps.

UseFishLenComps Binary indicator array for fishery length comps.

ObsSrvIdx Observed survey index array [region × year × fleet].

ObsSrvIdx_SE Standard error for survey index array.

UseSrvIdx Binary indicator array for survey indices.

ObsSrvAgeComps Observed survey age composition array.

ObsSrvLenComps Observed survey length composition array.

ISS_SrvAgeComps Implied sample sizes for survey age compositions.

ISS_SrvLenComps Implied sample sizes for survey length compositions.

UseSrvAgeComps Binary indicator array for survey age comps.

UseSrvLenComps Binary indicator array for survey length comps.

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_self_test()`

`simulation_self_test` *Conduct a Simulation Self Test*

Description

This function runs a self test of the fitted RTMB model by simulating new datasets under the fitted parameters, refitting the model, and comparing estimated outputs to the true values used for simulation. It can be run sequentially or in parallel.

Usage

```
simulation_self_test(
  data,
  parameters,
  mapping,
  random,
  rep,
  sd_rep,
  n_sims,
  newton_loops = 3,
  do_sdrep = FALSE,
  do_par = FALSE,
  n_cores = NULL,
  output_path = NULL,
  what = c("SSB", "Rec")
)
```

Arguments

`data`	A list containing model data from an RTMB object.
`parameters`	A list of fitted parameter values from an RTMB object.
`mapping`	A list specifying parameter mappings from an RTMB object.
`random`	Character vector specifying random effects.
`rep`	A list of report values from an RTMB object (‘$rep‘).
`sd_rep`	An ‘sdreport‘ object from RTMB summarizing parameter uncertainty.
`n_sims`	Integer. Number of simulation replicates to run.
`newton_loops`	Integer. Number of Newton loops used in model fitting (default: ‘3‘).
`do_sdrep`	Logical. If ‘TRUE‘, compute ‘sdreport‘ for each fitted replicate (default: ‘FALSE‘).
`do_par`	Logical. If ‘TRUE‘, run simulations in parallel (default: ‘FALSE‘).
`n_cores`	Integer. Number of cores to use for parallelization (default: ‘NULL‘ = detect automatically).
`output_path`	Optional file path. If provided, the simulated datasets are written to this location.
`what`	Character vector. Names of report elements in ‘rep‘ to extract and store for each replicate.

Value

A list with elements corresponding to the requested ‘what‘ values, each containing an array of simulation results across replicates. If ‘do_sdrep = TRUE‘, an additional element ‘"sd_rep"‘ is included with the list of ‘sdreport‘ objects (or ‘NA‘ if a replicate fails).

See Also

Other Simulation Setup: `Setup_Sim_Biologicals()`, `Setup_Sim_Containers()`, `Setup_Sim_Dim()`, `Setup_Sim_Fishing()`, `Setup_Sim_Rec()`, `Setup_Sim_Survey()`, `Setup_Sim_Tagging()`, `Setup_sim_env()`, `Simulate_Pop_Static()`, `run_annual_cycle()`, `simulation_data_to_SPoRC()`

Examples

```
## Not run:
# Run a simple self test with 10 simulations, extracting SSB
res <- simulation_self_test(
  data = model$data,
  parameters = model$parameters,
  mapping = model$mapping,
  random = model$random,
  rep = model$rep,
  sd_rep = model$sd_rep,
  n_sims = 10,
  what = "SSB"
)

str(res$SSB) # look at simulated SSB arrays

## End(Not run)
```

`theme_sablefish` *ggplot theme for sablefish*

Description

ggplot theme for sablefish

Usage

```
theme_sablefish()
```

Value

ggplot theme

See Also

Other Plotting: `get_biological_plot()`, `get_data_fitted_plot()`, `get_selex_plot()`, `get_ts_plot()`, `plot_all_basic()`

`three_rg_sable_data` *Sablefish data for multi region (3 area) case study*

Description

A dataset containing the necessary elements for the Alaska sablefish spatial case study.

Usage

```
three_rg_sable_data
```

Format

A list with multiple components needed for the multi (3) region sablefish model

Source

Cheng et al. 2025, Fish and Fisheries; Panmictic Panacea? Demonstrating Good Practices for Developing Spatial Stock Assessments through Application to Alaska Sablefish (Anoplopoma fimbria)

`three_rg_sable_rep` *Sablefish report for 3 region case study*

Description

A report containing results for the spatial Alaska sablefish case study.

Usage

```
three_rg_sable_rep
```

Format

Report file from the 3 region sablefish case study

Source

Cheng et al. 2025, Fish and Fisheries; Panmictic Panacea? Demonstrating Good Practices for Developing Spatial Stock Assessments through Application to Alaska Sablefish (Anoplopoma fimbria)

Index