Package: ss3sim (via r-universe)

October 25, 2024

Type Package

Title Fisheries Stock Assessment Simulation Testing with Stock Synthesis

Version 1.21.0

Description A framework for fisheries stock assessment simulation testing with Stock Synthesis (SS3) as described in Anderson et al. (2014) <doi:10.1371/journal.pone.0092725>.

License MIT + file LICENSE

URL https://github.com/ss3sim/ss3sim

BugReports https://github.com/ss3sim/ss3sim/issues

Depends R (>= 4.0)

Imports cli, dplyr, foreach, ggplot2, graphics, grDevices, gtools, lifecycle, magrittr, purrr, r4ss (>= 1.46.1), rlang, stats, stringi, tibble, tidyr, utils

Suggests doParallel, knitr, pak, rmarkdown, testthat

VignetteBuilder knitr

Remotes github::r4ss/r4ss@main

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://ss3sim.r-universe.dev

RemoteUrl https://github.com/ss3sim/ss3sim

RemoteRef HEAD

RemoteSha 54a41f5b8581c9555ffca5f46efac071c4c26c68

Contents

add_colnames *Create matching column names across a list of data frames*

Description

Add missing columns to each data frame in the list allowing for the use rbind() to create a single data frame. The code is based on rbind.fill from the plyr package.

Usage

```
add_colnames(dfs, bind = FALSE, fillwith = NA)
```

Arguments

dfs	A list of data frames, where the length can be one.
bind	A logical value specifying if the data frame(s) should be returned as a single data frame. The default is FALSE, which returns a list of data frames same as what was provided in dfs.
fillwith	A single value that will be used to populate all of the missing columns.

Value

Depending on the input to bind you can either return the same structure, i.e., a list of data frames, or a data frame with all rows from each original data frame. Missing values will be filled with the entry in fillwith.

Author(s)

Kelli F. Johnson

Examples

```
x <- data.frame("a" = 1:10, "b" = 21:30)
y <- data.frame("a" = 11:15, "y" = letters[1:5])
alist <- ss3sim:::add_colnames(list(x, y), bind = FALSE)
adataframe <- ss3sim:::add_colnames(list(x, y), bind = TRUE)
# clean up
rm(x, y, alist, adataframe)
```

add_tv_parlines *Add short time varying parameter lines. At time of writing, this method will work for MG, selectivity, and catchability time varying, but not for SR*

Description

Add short time varying parameter lines. At time of writing, this method will work for MG, selectivity, and catchability time varying, but not for SR

Usage

```
add_tv_parlines(string, tab, ctl_string, ss3.ctl)
```

Arguments

`string`	The code representing the section the parameter is from.
`tab`	As created in `change_tv()`.
`ctl_string`	The code as called in the .ss_new comment for time varying.
`ss3.ctl`	A Stock Synthesis control file that has been read in using `readLines()`.

Value

A modified version of ss3.ctl (a vector of strings), containing the new parameter line

calculate_bias *Calculate bias adjustment for recruitment deviations*

Description

Bias adjustment is performed to ensure that the only the most informative data available are used when estimating recruitment deviations. This process involves running the estimation method with the hessian both before this function and then running the estimation method again with the new values. Estimation files in the original folder will be deleted to ensure that convergence afterwards is not based on the input run. These files are permanently archived in the `bias_[0-9]{2}` folder within the directory. Only the three middle steps listed below are performed using `calculate_bias`, and the rest of the steps must be performed externally.

Usage

```
calculate_bias(dir, ctl_file_in)
```

Arguments

`dir`	A character string specifying the path to the folder with the results from a stock assessment model run. Paths are passed without a terminal slash.
`ctl_file_in`	A string providing the path to the input Stock Synthesis `.ctl` file.

Details

- Estimate recruitment and the standard error about those estimates.
- Correct the estimates given their estimated uncertainty using a ramp.
- Save a new control file.
- Move original estimation files.
- Estimate model parameters.

Value

A list of bias adjustment parameters.

Author(s)

Kelli F. Johnson

`calculate_data_units` *Calculate fleets, years, and data types needed given sampling parameters*

Description

Given the sampling arguments that are specified in `..._params`, e.g., `index_params`, calculate the super set of fleets, years, and data types that will be needed in the data file of expected values that is generated by the OM.

Usage

```
calculate_data_units(
  index_params = NULL,
  lcomp_params = NULL,
  agecomp_params = NULL,
  calcomp_params = NULL,
  mlacomp_params = NULL,
  wtatage_params = NULL
)
```

Arguments

`index_params`	Named lists containing the arguments for `sample_index()`.
`lcomp_params`	Named lists containing the arguments for `sample_lcomp()`.
`agecomp_params`	Named lists containing the arguments for `sample_agecomp()`.
`calcomp_params`	Named lists containing the arguments for `sample_calcomp()`.
`mlacomp_params`	Named lists containing the arguments for `sample_mlacomp()`.
`wtatage_params`	Named lists containing the arguments for `sample_wtatage()`.

Value

A list with the following three elements:

- fleets,
- years, and
- types.

Note

A superset by nature is larger than the individual sets used to create it (unless all sampling arguments are identical), so that the returned list will created some unnecessary combinations. This was done intentionally for simplicity but may be changed later.

Author(s)

Cole Monnahan

See Also

See further examples in clean_data and change_data

Examples

```
## Only one fleet
calculate_data_units(lcomp_params = list(fleets = 1, years = c(3, 4, 6)))
## Add new fleet
morefleets <- calculate_data_units(
  lcomp_params = list(fleets = 1, years = c(3, 4, 6)),
  agecomp_params = list(fleets = 2, years = 5)
)

## Add length or age if missing and conditional-age-at-length is included
test <- mapply(calculate_data_units,
  SIMPLIFY = FALSE,
  lcomp_params = list(NULL, list(fleets = 1, years = 1:10)),
  agecomp_params = list(NULL, NULL),
  MoreArgs = list(calcomp_params = list(fleets = 1, years = 1:10))
)

rm(test)
```

calculate_re *Calculate relative error*

Description

Calculate the relative error (RE; $[EM - OM]/OM$) of parameters and derived quantities stored in a scalar or time series data frame generated by get_results_all().

Usage

```
calculate_re(dat, add = TRUE, EM = "em")
```

Arguments

dat	An input data frame. Should be either a scalar or time series data frame as returned from get_results_all() or a related get_results_*(). Specifically, the data frame needs to have columns with _em and _om as names. If the data is provided in long rather than wide format, then convert_to_wide() will be used internally before calculating RE and a wide data frame will be returned.
add	Logical: should the relative error columns be added to dat or should the original EM and OM columns be dropped? If FALSE, then the returned data frame will have only the identifying columns and the new relative error columns. You could then merge selected columns back into dat if you wished. The default is to return all columns.
EM	A character value specifying the name of the EM to calculate the RE of when the results are provided in long format and there is the potential for multiple EMs. See the column model_run for options.

Value

The default is to return a data frame structured the same as the input data frame, i.e., dat, but with additional columns, where '_re' is appended to the base string of the column name. All NAN and Inf values are returned as NA values, typically because you cannot divide by zero. Irrelevant columns, i.e., columns of entirely zero of NA are removed prior to returning the data frame.

Author(s)

Sean Anderson and Cole Monnahan

See Also

get_results_all(), get_results_scenario()

Examples

```
# Example with built in package data:
data("ts_dat", package = "ss3sim")
data("scalar_dat", package = "ss3sim")
head(calculate_re(ts_dat))
head(calculate_re(ts_dat, add = FALSE))
head(calculate_re(scalar_dat, add = FALSE))
rm("ts_dat", "scalar_dat")
```

change_catch *Change catch in the Stock Synthesis data list*

Description

Change catch in the data so at least all combinations of fleet, season, and year, needed for catch are available. Equilibrium years are generated if there are equilibrium parameters in the control list.

Usage

```
change_catch(dat_list, ctl_list)
```

Arguments

dat_list	A data file read in using r4ss::SS_readdat().
ctl_list	A control file read in using r4ss::SS_readctl(). The start and end year of the resulting data list will be based on years with positive fishing mortality values, and equilibrium catches will be non-zero only if there is a equilibrium fishing mortality parameter for that fleet and season combination.

Value

A modified Stock Synthesis data file as a list in R.

Author(s)

Kathryn L. Doering

See Also

change_f() changes the fishing mortality, F, parameters using the control file, but these F values will only be implemented for years with corresponding entries in the Stock Synthesis data file. Thus, this function must be implemented after change_f().

change_comp — *Change composition data to dummy data for running the operating model*

Description

Change the composition data in a Stock Synthesis data list object or file to include rows of data that are desired. Typically, this will be an operating model (OM) because only dummy-data observations are used here, i.e., all compositions are set to a value of one. Creating these dummy observations is helpful before running your OM because it will facilitate the creation of observed values for each desired combination.

Usage

```
change_comp(dat_list, type = c("len", "age", "cal"), paramlist, nsex = 1, bins)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section` = 1 reads in the input values used to run the model and `section` = 2 reads in the expected values generated given all the input to the OM. `section` = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`type` The sample type you want. See the function call for available types, e.g., `formals(change_comp)$type`; the first value will be used as the default if user input is not provided.

`paramlist` A list of parameter values derived from the data frame used to set up your simila-tion. For example, `setup_scenarios(setup_scenarios_defaults())[[1]]` will give you defaults that you can extract from. Typically, `mylist[[c("agecomp_params", "lcomp_params")]]` are passed. Make sure that you only pass the portion of the list that pertains to the data you want.

`nsex` An integer value between one and two specifying the number of sexes in the model, where 1 is based on females only for spawning stock biomass and two-sex models allow for sex-specific parameters.

`bins` A vector of bins for the composition data. The bins do not need to be named because they will be renamed with their value and a leading character based on what type of data they are.

Examples

```
# todo: remove this example when testing is complete
## Not run:
change_comp(
  dat_list = dat, type = "len",
  paramlist = scenariol[[1]][[c("lcomp_params", "agecomp_params")]]
)

## End(Not run)
```

change_data *Change the data that is available from a list object*

Description

Alter the structure of data that is available from a Stock Synthesis operating model (OM), which in turn leads to changes in the output and ability to sample data after running the model.

Usage

```
change_data(
  dat_list,
  outfile = NULL,
  fleets,
  years,
  types = c("len", "age", "cal", "mla", "mwa"),
  age_bins = NULL,
  len_bins = NULL,
  pop_binwidth = NULL,
  pop_minimum_size = NULL,
  pop_maximum_size = NULL,
  lcomp_constant = NULL,
  tail_compression = NULL,
  nsex = 1
)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

`fleets` A numeric vector of fleets.

`years` A numeric vector of years.

`types` A vector that can take combinations of the following entries: `"len"`, `"age"`, `"cal"`, `"mla"`. `types` controls what data structures the function acts on, with `"len"` augmenting the length-composition data, `"age"` augmenting the age-composition data, `"cal"` augmenting the conditional age-at-length data, and `"mla"` augmenting the mean length-at-age data.

`age_bins` A numeric vector of age bins to use. If left as `NULL`, the age bin structure will be taken from the OM.

`len_bins` A numeric vector of length bins to use. If left as `NULL`, the length bin structure will be taken from the OM. For conditional age-at-length data, the last value provided to `len_bins` will be used for Lbin_lo and -1 will be used for Lbin_hi for the largest length bin category, i.e., row of conditional age-at-length data.

`pop_binwidth` Population length bin width. Note that this value must be smaller than the bin width specified in length-composition data `len_bins` or Stock Synthesis will fail, see notes in the Stock Synthesis manual.

`pop_minimum_size`
Population minimum length bin value.

`pop_maximum_size`
Population maximum length bin value.

`lcomp_constant` The robustification constant for length-composition data. Must be a numeric value, as a proportion. For example, 0.1 means 10 percent. See the notes in the Stock Synthesis manual. A `NULL` value indicates no action resulting in using the current value, and a value of 0 will throw an error because zero leads to an error when zeroes exist in the data. Instead use a very small value like 1e-07.

`tail_compression`
Tail compression value to be used in Stock Synthesis. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the notes in the Stock Synthesis manual. A `NULL` value indicates no action, a negative value turns the feature off in Stock Synthesis.

`nsex` An integer value of 1 or 2 specifying the number of sexes in the model. If 1, then females are the only included sex. This information can be found in the data file for a given model and dictates how the composition data are structured.

Details

`change_data()` is called internally within ss3sim, but it can be used to manipulate data or to prepare a new OM for use in a simulation. Original data is removed and dummy data is added to the Stock Synthesis `.dat` object. The dummy data expands the data structure to provide information for all years and fleets, potentially adding many rows of data.

Currently, `.dat` files with multiple sexes cannot be manipulated with `change_data()`.

The robustification constant is added to both the observed and expected proportions of length composition data, before being normalized internally. It is designed to help stabilize the model, but is unclear how and when to use it for optimal effect. The same value is used for all length data.

Value

An invisible data list, and a file is written to the disk if an entry other than the default of `NULL` is provided for `outfile`.

Author(s)

Cole Monnahan, Ian G. Taylor, Sean Anderson, Kelli F. Johnson

See Also

See `clean_data()` for a counter function.

Other change functions: `change_e()`, `change_em_binning()`, `change_f()`, `change_o()`, `change_retro()`, `change_tv()`

`change_dat_bin` *Change the bins for a composition object*

Description

Change the bins of a data frame object from a `dat_list`, filling in the columns with ones.

Usage

```
change_dat_bin(object, bins)
```

Arguments

`object` A data frame from a list object read in by `r4ss::SS_readdat()`. The leading columns, typically `Yr`, `Seas`, ... can be anything, but there must be columns with names that start with a lower-case letter, followed by integers.

`bins` A vector of characters or whatever you want the names of the new bins to be. Typically, this will be output from `setup_bins()`.

Value

A modified data frame where columns holding old composition data are removed in their entirety and new columns of ones are filled for each value in `bins`.

`change_e` *Methods to alter the parameters estimated in a Stock Synthesis model*

Description

Takes Stock Synthesis `.ctl` and `forecast.ss` files, along with a list structure which houses the data file as read in by `r4ss::SS_readdat()` and changes which parameters are estimated, how natural mortality is estimated, and if forecasts are performed. The function can be called by itself or within `run_ss3sim()` to alter an estimation model `.ctl` file.

Usage

```
change_e(
  ctl_file_in = "em.ctl",
  ctl_file_out = "em.ctl",
  dat_list = NULL,
  for_file_in = "forecasts.ss",
  par_name = NULL,
  par_int = "NA",
  par_phase = "NA",
  forecast_num = 0,
  verbose = FALSE
)
```

Arguments

`ctl_file_in` A string providing the path to the input Stock Synthesis `.ctl` file.

`ctl_file_out` A string providing the path to the output Stock Synthesis control file. If the value is `NULL`, the file will not be written to the disk.

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`for_file_in` A string providing the path to the input SS `forecast.ss` file.

`par_name` A vector of character values corresponding to parameter names that you wish to initialize at different values or change the phase in which they are estimated. Entries are searched for in `ctl_file_in`, and therefore, it is best to use full parameter names as they are specified in the file.

`par_int` A vector of initial values, one for each entry in `par_name`. Values of `NA` leave the INIT value for that parameter at the value found in the `.ctl` file.

`par_phase` A vector of phase values, one for each parameter in `par_name`. Values can be `NA` if you do not wish to change the phase for a given parameter. Negative values will fix the parameter at the INIT value.

`forecast_num` Number of years to perform forecasts. For those years, the data will be removed from the `dat_list`, enabling Stock Synthesis to generate forecasts rather than use the data to fit the model.

`verbose` When `TRUE` messages will be returned from the function. Often useful for debugging. The default is `FALSE`.

Value

Altered versions of Stock Synthesis `.ctl` and `forecast.ss` files are written to the disk and the altered `dat_list` is returned invisibly.

Author(s)

Kelli F. Johnson

See Also

Other change functions: change_data(), change_em_binning(), change_f(), change_o(), change_retro(), change_tv()

Examples

```
d <- system.file("extdata", "models", "cod-om", package = "ss3sim")
change_e(
  ctl_file_in = file.path(d, "codOM.ctl"),
  ctl_file_out = file.path(tempdir(), "change_e.ctl"),
  dat_list = codomdat,
  for_file_in = file.path(d, "forecast.ss"),
  par_name = c("_steep", "Size_DblN_peak_Fishery(1)"),
  par_int = c(0.3, 40), par_phase = c(3, 2),
  forecast_num = 0
)
# clean up the temporary files
file.remove(file.path(tempdir(), "change_e.ctl"))
```

change_em_binning *Change population and observed length-composition bins*

Description

change_em_binning() alters the bin structure for the population and length-composition data in a Stock Synthesis estimation model (EM). The original length-composition data from the EM .dat is changed according to the user's specification. If the data file also contains conditional age-at-length data, then these data will be re-binned as well.

Usage

```
change_em_binning(
  dat_list,
  outfile = NULL,
  bin_vector,
  lbin_method = NULL,
  pop_binwidth = NULL,
  pop_minimum_size = NULL,
  pop_maximum_size = NULL
)
```

Arguments

dat_list
: A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and section = 2 reads in the expected values generated given all the input to the OM. section = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

outfile
: A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.

bin_vector
: A numeric vector of new length bins to substitute into the *.dat file.

lbin_method
: A numeric value of either NULL, 1, 2, 3 to change the lbin_method for the population bin. NULL means to not re-bin.

pop_binwidth
: Population length bin width. Only necessary for lbin_method = 2. Note that this value must be smaller than the bin width specified in length-composition data len_bins or Stock Synthesis will fail (see notes in the Stock Synthesis manual).

pop_minimum_size
: Population minimum length bin value. Only necessary for lbin_method = 2.

pop_maximum_size
: Population maximum length bin value. Only necessary for lbin_method = 2.

Author(s)

Kotaro Ono (length-composition rebinning), Sean Anderson (conditional age-at-length rebinning)

See Also

Other change functions: change_data(), change_e(), change_f(), change_o(), change_retro(), change_tv()

Examples

```
# Note that typically this function is used with estimation models in ss3sim,
# but it is used with an operating model data file in the following examples.
f <- system.file("extdata", "models", "cod-om", "codOM.dat", package = "ss3sim")
d <- r4ss::SS_readdat(f, verbose = FALSE)

# An example with lbin_method = 1
l1 <- change_em_binning(d,
  outfile = NULL, lbin_method = 1,
  bin_vector = seq(20, 152, by = 4)
)
l1$lbin_vector
head(l1$lencomp)
```

```
# An example with lbin_method = 2
new_bin_vec <- seq(min(d$lbin_vector), max(d$lbin_vector), by = 4)
# add the max value if necessary.
if (new_bin_vec[length(new_bin_vec)] != d$lbin_vector[length(d$lbin_vector)]) {
  new_bin_vec <- c(
    new_bin_vec,
    d$lbin_vector[length(d$lbin_vector)]
  )
}
pop_bin_input <- 5
pop_min_size_input <- min(d$lbin_vector_pop) - 1
pop_max_size_input <- max(d$lbin_vector_pop) + 5
lbin_vec_pop <- seq(pop_min_size_input,
  pop_max_size_input,
  length.out = (pop_max_size_input - pop_min_size_input) /
    pop_bin_input + 1
)
l2 <- change_em_binning(
  dat_list = d,
  bin_vector = new_bin_vec,
  lbin_method = 2,
  # Note: need more inputs with lbin_method = 2
  pop_binwidth = pop_bin_input,
  pop_minimum_size = pop_min_size_input,
  pop_maximum_size = pop_max_size_input
)
l2$lbin_method
# note bin width is now the same as the input
pop_bin_input
l2$binwidth
# note the minimum size has changed based on the input:
pop_min_size_input
l2$minimum_size
# so has max
l2$maximum_size
l2$lbin_vector
# other modified components:
l2$lbin_vector_pop
head(l2$lencomp)
```

change_f *Specify fishing mortality, F, using the Stock Synthesis control file*

Description

Replace or input a time series of fishing mortality, F, values into a Stock Synthesis control file. In Stock Synthesis, inserting F values in this manner, relies on the assumption that F operates continuously throughout the year and and the process operates jointly with natural mortality (Baranov 1918; Branch 2009). The documentation for Stock Synthesis also describes this process as F method == 2, where F is continuous and modeled using full parameters.

Usage

```
change_f(years, fleets, fvals, seasons = 1, ses = 0.005, ctl_list)
```

Arguments

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`fleets` *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.

`fvals` A list of the same length as `fleets` with one entry per fishing mortality level, F, entry in `years`. A single value will be repeated for every value in `years`. If more than one fleet is present, then the single value will be used for all fleets, i.e., there is no way to map a single value to each year specific to the given fleet. Instead you would need to provide a list of vectors of repeated values.

`seasons` A list of seasons to be entered into the Stock Synthesis control file for each fleet. The structure is the same as `fvals`, i.e., a list or a scalar. The default is 1, which will be applied to all fleets in all years.

`ses` A list of fishing level standard errors (ses) to be entered into the Stock Synthesis control file for each fleet. The structure is the same as `fvals`, i.e., a list or a scalar. The default is 0.005, which will be applied to all fleets in all years.

`ctl_list` A control file read in by `SS_readctl`.

Details

The argument `years` is the only argument that must be a vector or a list of vectors. Other arguments can be specified using a single scalar value that will be repeated for all fisheries in all years. If the input argument needs to be different for any year or fishery, the argument must be a list with vectors for each fishery, where each vector is the same length as the vectors within the `years` argument. Although, both the `years` and other input arguments can be specified using a single vector if the length of `fleets` is just one or a vector of values is specified for fleets because all of these vectors will just be combined into a single data frame. Where it gets complicated is when there are multiple fleet and year combinations, then it is best to just use the list structure common to other functions within **ss3sim**.

`change_f()` overrides any F values that are in the supplied control file with the newly specified values, i.e., `fvals`. Users do not need to specify values for years in which there will be zero fishing because Stock Synthesis will automatically set them to zero when running the operating model. Using the control file rather than the par file to manipulate the operating model requires a few other files within the operating model folder to be set up in a particular manner. That is, (1) the starter file must be set up to read parameters from the control file rather than the par file and (2) the data file must have a dummy catch entry for every year, fishery combination that will be specified in the control file. If a year, fishery combination is specified in the control file and not present in the data file, then the entry in the control file will be ignored. ss3sim automatically corrects for this using `ss3sim_base()` by specifying a row for every year and fleet using `change_catch()`.

Value

Modified Stock Synthesis control file list.

Author(s)

Kelli F. Johnson

See Also

See r4ss::SS_readctl() and r4ss::SS_writectl() for how to supply ctl_list and how to write the file back to the disk once you are done manipulating the list object.

Other change functions: change_data(), change_e(), change_em_binning(), change_o(), change_retro(), change_tv()

Examples

```
dat <- r4ss::SS_readdat(
  system.file("extdata", "models", "cod-om", "codOM.dat", package = "ss3sim"),
  verbose = FALSE
)
ctl <- r4ss::SS_readctl(
  system.file("extdata", "models", "cod-om", "codOM.ctl", package = "ss3sim"),
  verbose = FALSE, use_datlist = TRUE, datlist = dat
)
# Using original vector-style inputs
newctl <- change_f(years = 1:50, fleets = 1, fvals = 0.2, ctl_list = ctl)
# Using list-style inputs for when there are multiple fisheries
newctl <- change_f(
  years = list(1:5, 1:10), fleets = 3:4,
  fvals = list(rep(0.1, 5), rep(0.2, 10)), ctl_list = ctl
)
rm(dat, ctl, newctl)
```

change_lcomp_constant *Set the robustification constant for length-composition data*

Description

This function replaces the robustification value for length-composition data in a .dat file that was read in using r4ss::SS_readdat() with those specified in lcomp_constant. It then writes a new file with name outfile into the working directory.

Usage

```
change_lcomp_constant(lcomp_constant, dat_list, outfile = NULL)
```

Arguments

lcomp_constant	The new value to be used. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the Stock Synthesis manual for further information. A NULL value indicates no action resulting in using the current value, and a value of 0 will throw an error since that leads to an error when zeroes exist in the data. Instead use a very small value like 1e-07.

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

Details

The robustification constant is added to both the observed and expected proportions of length composition data, before being normalized internally. It is designed to help stabilize the model, but is unclear how and when to use it for optimal effect. The same value is used for all length data.

Value

A modified Stock Synthesis `.dat` file, and that file returned invisibly (for testing) as a vector of character lines.

Author(s)

Cole Monnahan

change_o *Methods to include parameters in a Stock Synthesis operating model*

Description

`change_o` takes a Stock Synthesis `.ctl` file and implements parameter value changes that are NOT time varying. `change_o` is specifically set up to work with an operating model `.ctl` file.

Usage

```
change_o(
  change_o_list,
  ctl_file_in = "control.ss_new",
  ctl_file_out = "om.ctl",
  par_name = NULL,
  par_int = NULL,
  verbose = FALSE
)
```

Arguments

`change_o_list`	A list of named vectors. Names correspond to parameters in the operating model and the vectors correspond to deviations. Alternatively, `par_name` and `par_init` can be passed to this function.
`ctl_file_in`	A string providing the path to the input Stock Synthesis `.ctl` file.
`ctl_file_out`	A string providing the path to the output Stock Synthesis control file. If the value is `NULL`, the file will not be written to the disk.
`par_name`	A vector of character values corresponding to parameter names that you wish to initialize at different values or change the phase in which they are estimated. Entries are searched for in `ctl_file_in`, and therefore, it is best to use full parameter names as they are specified in the file.
`par_int`	A vector of initial values, one for each entry in `par_name`. Values of `NA` leave the INIT value for that parameter at the value found in the `.ctl` file.
`verbose`	When `TRUE` messages will be returned from the function. Often useful for debugging. The default is `FALSE`.

Value

The function creates modified versions of the `.ctl` files. The function also returns `change_o_list` invisibly.

Specifying the `change_o_list`

Parameters initial values will change according to the values passed to `change_o_list`. Each parameter should have a single value specified.

Parameter names must be unique and match the full parameter name in the `.ctl` file.

Passing arguments to `change_o()` through `run_ss3sim()`

(1) add a column called `co.par_name` to the `simdf` that specifies which parameters you want to change in the OM, each element of this vector needs to be wrapped in quotations to be later evaluated, e.g., `'c("SR_BH_steep","SR_sigmaR")'` represents a single entry; and (2) add an additional column called `co.par_int` to the `simdf` that specifies INIT values for each parameter in the previous column, e.g., `"c(0.6, 1.0)"`, if there is more than one value, the vector needs to be wrapped in quotations to be evaluated later.

Author(s)

Kathryn L. Doering

See Also

Other change functions: `change_data()`, `change_e()`, `change_em_binning()`, `change_f()`, `change_retro()`, `change_tv()`

change_pop_bin *Set up population length bin structure*

Description

The population length bins in Stock Synthesis structure size data and empirical weight-at-age data. change_pop_bin changes the data file to contain specifications to create a vector (length-bin method of 2) rather than the actual bins from the length data (length-bin method of 1) or an actual vector (length-bin method of 3).

Usage

```
change_pop_bin(
  dat_list,
  binwidth = NULL,
  minimum_size = NULL,
  maximum_size = NULL,
  maximum_age = NULL
)
```

Arguments

dat_list	A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and section = 2 reads in the expected values generated given all the input to the OM. section = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.
binwidth	A numeric value specifying the width of the size bins.
minimum_size	The smallest size bin.
maximum_size	The largest size bin.
maximum_age	The highest age. Used to structure the maximum age of the population and the ageing-error matrix, which will be assumed to have no bias and maximum precision for any added ages.

Details

The only required argument is dat_list and the remaining arguments default to a value of NULL, which leads to the data file not being changed.

Value

A modified Stock Synthesis data file in list form. The list is only returned if it is assigned to an object.

change_q *Adds or removes catchability parameters from a control file*

Description

Manipulates the control list to simultaneously add and remove elements related to time series data on trends.

Usage

```
change_q(
  string_add = NULL,
  string_remove = NULL,
  ctl_list,
  dat_list = lifecycle::deprecated(),
  ctl_file_in = lifecycle::deprecated(),
  dat_file_in = lifecycle::deprecated(),
  ctl_file_out = lifecycle::deprecated(),
  overwrite = lifecycle::deprecated(),
  verbose = lifecycle::deprecated()
)
```

Arguments

`string_add`	A vector of fleet names and/or integers representing fleets that need q parameters added to the control file.
`string_remove`	A vector of fleet names and/or integers representing fleets that need q parameters removed from the control file.
`ctl_list`	A control file read in by `SS_readctl`.
`dat_list`	Deprecated with ss3sim version 1.19.1 because users can obtain fleet information from `ctl_list`.
`ctl_file_in`	Deprecated with ss3sim version 1.19.1 because users can pass list as read in by `r4ss::SS_readctl()` rather than specifying the file name to be read in.
`dat_file_in`	Deprecated with ss3sim version 1.19.1 because users can pass list as read in by `r4ss::SS_readdat()` rather than specifying the file name to be read in.
`ctl_file_out`	Deprecated with ss3sim version 1.19.1 because ss3sim uses the returned list internally rather than the saved control file.
`overwrite`	Deprecated with ss3sim version 1.19.1 because the file is no longer being saved to the disk. So, there is nothing to overwrite.
`verbose`	Deprecated with ss3sim version 1.19.1 because all messages were removed.

Details

Catchability, q, represents the proportionality constant between data on trends and estimated population abundance. Thus a survey thought to encapsulate the entire population, e.g., an acoustic survey of the entire area, will have $q = 1$. In Stock Synthesis, environmental time series are modelled similarly to a survey or catch-per-unit-effort time series and thus will also have a catchability term. Readers interested in the complete range of functionality should see the catchability section of the Stock Synthesis user manual. `change_q()` has limited functionality relative to what is available in Stock Synthesis. For example, `change_q()` cannot add parameters for additional variance. Though it will remove additional variance parameters for fleets that no longer have survey data. Additionally, the float term is not used within ss3sim and is set to zero.

Value

A modified Stock Synthesis control list.

Author(s)

Kelli F. Johnson

See Also

- `check_q()` determines which fleets should removed or added.
- `r4ss::SS_readctl()` reads in the control file passed to `ctl_list`.
- `find_position()` allows `string_*` to use strings or integers.

Examples

```
removedfleet1 <- change_q(string_remove = 1, ctl_list = codomctl)
removedfleet2 <- change_q(string_remove = 2, ctl_list = codomctl)
removedfleets <- change_q(
  string_remove = c("Fishery", 2),
  ctl_list = codomctl
)
testthat::expect_null(removedfleets[["Q_options"]])
newctl <- codomctl
newctl[["fleetnames"]] <- c(newctl[["fleetnames"]], "testfleet")
newctl[["Nfleets"]] <- length(newctl[["fleetnames"]])
newctl <- change_q(string_add = "testfleet", ctl_list = newctl)
testthat::expect_equal(newctl[["Q_options"]][, "fleet"], 1:3)
```

change_recyear *Change start year main recruitment deviations in control file*

Description

Change start year main recruitment deviations in control file

Usage

```
change_recyear(ctl_list, main)
```

Arguments

`ctl_list`	A control file read in by `SS_readctl`.
`main`	An integer specifying the year to start the main period of recruitment.

Value

A `r4ss::SS_readctl()` list with an augmented start year of the recruitment deviations in the main period.

Author(s)

Kelli F. Johnson

`change_rec_devs` *Replace recruitment deviations*

Description

This function replaces the recruitment deviations in the control file of a Stock Synthesis model with those specified in the argument `recdevs`. The new control file is then written to the disk if `ctl_file_out` is specified. It is imperative that the path provided in `ctl_file_in` be to a `ss_new` file so `change_rec_devs` can properly determine where to place the recruitment deviations in the control file.

Usage

```
change_rec_devs(recdevs, ctl_file_in, ctl_file_out = "control_recruitment.ss")
```

Arguments

`recdevs`	A vector of recruitment deviations to be entered into the Stock Synthesis control file. The vector must be the same length as the vector of recruitment deviations that are commented out in the `ss_new` control file. This vector can be found by searching for `# all recruitment deviations` within the file. If a single value is provided instead of a vector, the value will be repeated for every recruitment deviation in the model. Alternatively, users can supply a named vector with each name being a year of the model. Missing years will be filled in with values of zero.
`ctl_file_in`	A string providing the path to the input Stock Synthesis `.ctl` file.
`ctl_file_out`	A string providing the path to the output Stock Synthesis control file. If the value is `NULL`, the file will not be written to the disk.

Value

A modified Stock Synthesis control file.

Author(s)

Kelli F. Johnson

Examples

```
d <- system.file(file.path("extdata", "models"), package = "ss3sim")
change_rec_devs(
  recdevs = rlnorm(101),
  ctl_file_in = file.path(d, "cod-om", "codOM.ctl"),
  ctl_file_out = file.path(tempdir(), "control_recdevs.ss")
)
# Change the recruitment deviations in years 2:11
change_rec_devs(
  recdevs = stats::setNames(rlnorm(10), 2:11),
  ctl_file_in = file.path(d, "cod-om", "codOM.ctl"),
  ctl_file_out = file.path(tempdir(), "control_recdevsInitial.ss")
)
lapply(
  X = dir(tempdir(), pattern = "control_.+ss", full.names = TRUE),
  FUN = unlink
)
```

change_retro *Alter a starter file for a retrospective analysis*

Description

A retrospective analysis tests the effect of peeling back the number of operating model years observable to the estimation model. This function alters the Stock Synthesis starter file to run a retrospective analysis.

Usage

```
change_retro(
  str_file_in = "starter.ss",
  str_file_out = "starter.ss",
  retro_yr = 0
)
```

Arguments

`str_file_in` A string providing the path to the input Stock Synthesis `starter.ss` file.

`str_file_out` A string providing the path to the output Stock Synthesis `starter.ss` file.

`retro_yr` Which retrospective year to enter into the starter file. Should be 0 (no retrospective analysis) or a negative value, which leads to the removal of data for the specified number of years. Positive values are not allowed.

Details

Note that the starter file is set up to run a single retrospective run. Therefore, if you would like to run retrospective analyses for, say, 0, 1, 2, 3, 4, and 5 years, you will need to use this function to adjust the starter file 6 separate times.

Value

A modified Stock Synthesis starter file.

Author(s)

Sean C. Anderson

See Also

Other change functions: `change_data()`, `change_e()`, `change_em_binning()`, `change_f()`, `change_o()`, `change_tv()`

Examples

```
# Create a temporary folder for the output:
temp_path <- file.path(tempdir(), "ss3sim-retro-example")
dir.create(temp_path, showWarnings = FALSE)

# Locate the package data:
starterfile <- system.file("extdata", "models", "cod-om",
  "starter.ss",
  package = "ss3sim"
)

# No retrospective analysis:
change_retro(starterfile, paste0(temp_path, "/retro-0-starter.ss"),
  retro_yr = 0
)

# A retrospective analysis of 5 years:
change_retro(starterfile, paste0(temp_path, "/retro-5-starter.ss"),
  retro_yr = -5
)
```

`change_startyear` *Change start year of the data file*

Description

Change start year of the data file

Usage

```
change_startyear(dat_list, firstyear = NULL)
```

Arguments

dat_list A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and section = 2 reads in the expected values generated given all the input to the OM. section = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

firstyear An integer specifying the year to start fitting the model. The default is NULL, which will look up the first year with non-zero catch or non-zero catch-per-unit-effort data in dat_list.

Value

A r4ss::SS_readdat() list with an augmented start year.

Author(s)

Kelli F. Johnson

---

change_tail_compression

Replace tail compression value for length composition data

Description

This function replaces the tail compression value for length-composition data in a .dat file that was read in using r4ss::SS_readdat() with those specified in tail_compression. It then writes a new file with name dat_file_out into the working directory.

Usage

```
change_tail_compression(tail_compression, dat_list, outfile = NULL)
```

Arguments

tail_compression
*The new tail_compression value to be used. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the Stock Synthesis manual for further information. A NULL value indicates no action, a negative value indicates to Stock Synthesis to ignore it (not use that feature).

dat_list A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and section = 2 reads in the expected values generated given all the input to the OM. section = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

outfile — A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.

Value

A modified Stock Synthesis .dat file is returned invisibly.

Author(s)

Cole Monnahan

change_tv *Methods to include time-varying parameters in a Stock Synthesis operating model*

Description

change_tv takes Stock Synthesis .ctl, .par, and .dat files and implements time-varying parameters using environmental variables. change_tv is specifically set up to work with an operating model .ctl file.

Usage

```
change_tv(
  change_tv_list,
  ctl_file_in = "control.ss_new",
  ctl_file_out = "om.ctl",
  dat_file_in = "ss3.dat",
  dat_file_out = "ss3.dat"
)
```

Arguments

change_tv_list — A list of named vectors. Names correspond to parameters in the operating model that currently do not use environmental deviations and the vectors correspond to deviations. See the section "Specifying the change_tv_list" for help on specifying this argument.

ctl_file_in — A string providing the path to the input Stock Synthesis .ctl file.

ctl_file_out — A string providing the path to the output Stock Synthesis control file. If the value is NULL, the file will not be written to the disk.

dat_file_in — A string providing the path to the input Stock Synthesis .dat file.

dat_file_out — A string providing the path to the output Stock Synthesis .dat file.

Details

Although there are three ways to implement time-varying parameters within Stock Synthesis, **ss3sim** and `change_tv` only use the environmental variable option. Within Stock Synthesis, time-varying parameters work on an annual time-step. Thus, for models with multiple seasons, the time-varying parameters will remain constant for the entire year.

The `ctl_file_in` argument needs to be a `.ss_new` file because the documentation in `.ss_new` files are automated and standardized. This function takes advantage of the standard documentation the `.ss_new` files to determine which lines to manipulate and where to add code in the `.ctl`, `.par`, and `.dat` files, code that is necessary to implement time-varying parameters.

ss3sim uses annual recruitment deviations and may not work with a model that ties recruitment deviations to environmental covariates. If you need to compare the environment to annual recruitment deviations, the preferred option is to transform the environmental variable into an age 0 pre-recruit survey. See page 55 of the Stock Synthesis version 3.24f manual for more information.

Value

The function creates modified versions of the `.ctl` and `.dat` files if ctl_file_out and dat_file_out are not NULL. The function also returns a list of the modified `.ctl` and `.dat` R objects invisibly.

Specifying the `change_tv_list`

Parameters will change to vary with time according to the vectors of deviations passed to `change_tv_list`. Vectors of deviations, also referred to as environmental data, must have a length equal to `endyr-startyr+1`, where `endyr` and `startyr` are specified the `.dat` file. Specify years without deviations as zero.

Parameter names must be unique and match the full parameter name in the `.ctl` file. Names for stock recruit parameters must contain "devs", "R0", or "steep", and only one stock recruit parameter can be time-varying per model.

This feature will include an *additive* functional linkage between environmental data and the parameter where the link parameter is fixed at a value of one and the par value is specified in the `.par` file: $par'[y] = par + link * env[y]$.

For catchability (q) the *additive* functional linkage is implemented on the log scale: $ln(q'[y]) = ln(q) + link * env[y]$

Author(s)

Kotaro Ono, Carey McGilliard, Kelli F. Johnson, and Kathryn L. Doering

See Also

Other change functions: `change_data()`, `change_e()`, `change_em_binning()`, `change_f()`, `change_o()`, `change_retro()`

Examples

```
## Not run:
# Create a temporary folder for the output and set the working directory:
temp_path <- file.path(tempdir(), "ss3sim-tv-example")
dir.create(temp_path, showWarnings = FALSE)
```

```
wd <- getwd()
setwd(temp_path)
on.exit(setwd(wd), add = TRUE)

d <- system.file("extdata", package = "ss3sim")
om <- file.path(d, "models", "cod-om")
dir.create("cod-om")
file.copy(om, ".", recursive = TRUE)
setwd("cod-om")

change_tv(
  change_tv_list =
    list(
      "NatM_uniform_Fem_GP_1" = c(rep(0, 20), rep(.1, 80)),
      "SR_BH_steep" = stats::rnorm(100, 0, 0.05)
    ),
  ctl_file_in = "codOM.ctl",
  ctl_file_out = "example.ctl",
  dat_file_in = "codOM.dat",
  dat_file_out = "example.dat"
)

# Clean up:
unlink("cod-om", recursive = TRUE)

## End(Not run)
```

---

change_year — *Change the years estimated*

Description

Keep all of the data in the model but change the years that are estimated in the model. First year of the model will be first year of non-zero catch. Main recruitment period starts 1/2 generation time before first year of compositional data included in the model. Late recruitment is the last year of the model by default and cannot be modified using this function, neither can early recruitment, which starts in year 1.

Usage

```
change_year(dat_list, ctl_list)
```

Arguments

dat_list — A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and section = 2 reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`ctl_list` A control file read in by `SS_readctl`.

`check_data` *Check that the Stock Synthesis data file looks correct*

Description

Check that the Stock Synthesis data file looks correct

Usage

```
check_data(x)
```

Arguments

`x` A Stock Synthesis data list object as read in by `r4ss::SS_readdat()`.

`check_data_str_range` *Check input arguments for data*

Description

Check that the param list inputs have correct structure and range given an associated data file.

Usage

```
check_data_str_range(all_params, dat_list)
```

Arguments

`all_params` A named list of the parameters containing at a minimum year and fleet values

`dat_list` A Stock Synthesis data list object as read in by `r4ss::SS_readdat()`.

`check_eqlength` *Check if input arguments have the same length*

Description

Calculate the length of all input arguments to see if they are equal. Entries that are `NULL`, and thus, have a length of zero are ignored. An optional trigger to `stop()` is provided with a tailored error message.

Usage

```
check_eqlength(..., keepgoing = FALSE)
```

Arguments

`...`	Input arguments of unknown length.
`keepgoing`	A logical value specifying if the function should continue or terminate upon finding input arguments of non-equal length. The default, `FALSE`, produces an error and terminates the function.

Value

`TRUE` or `FALSE` depending on the result of the test. Nothing is returned if the stop function is invoked.

Author(s)

Kelli F. Johnson

Examples

`check_forecast` *Check input forecast file values*

Description

Ensure that the `forecast.ss` file is configured for use in ss3sim.

Usage

```
check_forecast(for_list)
```

Arguments

`for_list`	A Stock Synthesis forecast list object as read in from `SS_readforecast` in the **r4ss** package.

Details

- fish at F_{MSY}
- use relative benchmark years (i.e., Bmark_years)
- use relative years for fishing specifications, i.e., Fcast_years

Value

A an augmented list object, as returned by r4ss::SS_readforecast(), is invisibly returned.

Author(s)

Kelli F. Johnson

check_q *Check if desired q parameters exist in control file list*

Description

Check a Stock Synthesis control file to determine if the desired fleets have q parameters set up.

Usage

```
check_q(ctl_list, Nfleets = lifecycle::deprecated(), desiredfleets)
```

Arguments

ctl_list	A control file read in by SS_readctl.
Nfleets	Deprecated with ss3sim version 1.19.1 because the number of fleets is available in ctl_list.
desiredfleets	A numeric vector specifying which fleets should have catchability parameters.

Value

A list with two vectors, add and remove, specifying which fleets to add and which to remove from the control file.

See Also

change_q() for actually adding or removing the fleets.

Examples

```
# Keep just the fishery
stopifnot(check_q(ctl_list = codomctl, desiredfleets = 1)[["remove"]] == 2)
# All elements of the returned list should be NULL
# because the model only has two \eqn{q} parameters
stopifnot(all(mapply(is.null, check_q(codomctl, desiredfleets = 1:2))))
# Fleet 3 is not present
stopifnot(check_q(codomctl, desiredfleets = 1:3)[["add"]] == 3)
stopifnot(check_q(codomctl, desiredfleets = 2:3)[["remove"]] == 1)
```

---

clean_data *Given sampling arguments, remove unneeded data from a .dat file*

Description

This prepares a .dat file to be used by an estimation method, whereas before it may have had leftover data from sampling purposes.

Usage

```
clean_data(
  dat_list,
  lcomp_params = NULL,
  agecomp_params = NULL,
  calcomp_params = NULL,
  mlacomp_params = NULL,
  verbose = FALSE
)
```

Arguments

dat_list	A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and section = 2 reads in the expected values generated given all the input to the OM. section = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.
lcomp_params	Named lists containing the arguments for sample_lcomp().
agecomp_params	Named lists containing the arguments for sample_agecomp().
calcomp_params	Named lists containing the arguments for sample_calcomp().
mlacomp_params	Named lists containing the arguments for sample_mlacomp().
verbose	When TRUE it will print a message when rows are deleted.

Value

An invisible cleaned data list as an object.

Note

This function does not write the result to file.

Author(s)

Cole Monnahan

See Also

calculate_data_units()

Other sampling functions: sample_agecomp(), sample_calcomp(), sample_catch(), sample_discard(), sample_index(), sample_lcomp(), sample_mlacomp(), sample_wtatage()

---

codemctl *Control file for the cod estimation method*

Description

A list of controls returned from r4ss::SS_readctl() for the North Sea cod operating model. The input file is stored in extdata/models.

Usage

```
codemctl
```

Format

A list with many items, some of which are highlighted below:

fleetnames a vector of names for the fleets

MG_parms natural mortality and growth parameters

SR_parms stock-recruitment relationship parameters ...

Source

North Sea cod (*Gadus morhua*; Richard D. Methot, Jr., NMFS, NOAA, pers. comm.)

See Also

- r4ss::SS_readctl()
- r4ss::SS_readdat()

Examples

```
data("codemctl", package = "ss3sim")
```

`codomctl` *Control file for the cod operating model*

Description

A list of controls returned from `r4ss::SS_readctl()` for the North Sea cod operating model. The input file is stored in `extdata/models`.

Usage

```
codomctl
```

Format

A list with many items, some of which are highlighted below:

fleetnames a vector of names for the fleets

MG_parms natural mortality and growth parameters

SR_parms stock-recruitment relationship parameters ...

Source

North Sea cod (*Gadus morhua*; Richard D. Methot, Jr., NMFS, NOAA, pers. comm.)

See Also

- `r4ss::SS_readctl()`
- `r4ss::SS_readdat()`

Examples

```
data("codomctl", package = "ss3sim")
```

`codomdat` *Data for the cod operating model*

Description

A list of data returned from `r4ss::SS_readdat()` for the North Sea cod operating model. The input file is stored in `extdata/models`.

Usage

```
codomdat
```

Format

A list with many items, some of which are highlighted below:

catch data frame of catches by year, fleet, and season

CPUE catch-per-unit-effort data

lencomp length-composition data ...

Source

North Sea cod (*Gadus morhua*; Richard D. Methot, Jr., NMFS, NOAA, pers. comm.)

See Also

`r4ss::SS_readdat()`

Examples

```
data("codomdat", package = "ss3sim")
```

`convert_to_wide` *Convert long-style ss3sim output to wide format*

Description

This function exists for back compatibility. Note that this will only work if the column model_run has only the strings"om" or "em".

Usage

```
convert_to_wide(lng)
```

Arguments

`lng`	A long dataframe produced from get_results_all().

Value

A wide dataframe (separate columns for em and om results)

Author(s)

Kathryn L. Doering

Examples

```
## Not run:
scalar <- utils::read.csv("ss3sim_scalar.csv")
scalar_wide <- convert_to_wide(scalar)

ts <- utils::read.csv("ss3sim_ts.csv")
ts_wide <- convert_to_wide(scalar)

## End(Not run)
```

`copy_ss3models` *Copy the OM or EM into a scenario directory*

Description

Copy the OM or EM into a scenario directory

Usage

```
copy_ss3models(model_dir, scenario, iteration = 1, type = c("om", "em"))
```

Arguments

`model_dir`	A directory containing an OM or EM.
`scenario`	A string giving the scenario name which will be used in the resulting directory name. If you want this directory created somewhere other than your current working directory, you can pass a full file path with the last level being the new scenario name. All intermediate directories that do not exist will be created.
`iteration`	An integer specifying the iteration of interest.
`type`	Either `"om"` or `"em"` depending on which type of model folder needs created.

Value

An invisible boolean for whether that iteration already existed.

Author(s)

Sean C. Anderson, Kelli F. Johnson

Examples

```
# Locate the package data:
om_folder <- system.file(
  "extdata", "models", "cod-om",
  package = "ss3sim"
)

# Copy the operating model:
```

```
copy_ss3models(
  model_dir = om_folder,
  scenario = "D0-F0-testing"
)
# Now look at your working directory in your file system

# Copy the EM
copy_ss3models(
  model_dir = om_folder,
  type = "em",
  scenario = "D1-F0-testing"
)
# Scenario argument affects the folder names.

# Clean up:
unlink("D0-F0-testing", recursive = TRUE)
unlink("D1-F0-testing", recursive = TRUE)
```

create_em *Create an EM from an OM*

Description

Create estimation model (EM) files from operating model (OM) files. By making small changes to the OM rather than having two sets of files, less files need to be maintained. Differences between the OM and EM are mainly related to how the OM takes input fishing mortality values rather than absolute catches.

Usage

```
create_em(
  dir_in = system.file("extdata", "models", "cod-om", package = "ss3sim"),
  dir_out = file.path(getwd(), "new-em")
)
```

Arguments

`dir_in`	A file path to a directory that contains the following files: `forecast.ss`, `starter.ss`, and a control file (e.g., `xxxOM.ctl`). The default is to get the codOM within **ss3sim**.
`dir_out`	A file path to a directory where the new files will be saved. The default is to save the files in your current working directory in a folder called `new-em`.

Value

Nothing is returned, but three files are saved to the disk in the specified folder that may also be new.

Control file

Most changes to the EM control file relate to recruitment and fishing. The phase in which recruitment deviations are estimated is checked to ensure that it is positive. Though, this might be unnecessary because the OM file can have negative or positive phases. Thus, users are encouraged to just set the phase in which recruitment is estimated in the OM at the value that they would like to use in the EM. Additional changes are made to the bias adjustment procedure based on the biology of the stock.

The F_Method is set to 3 to allow the model to estimate fishing mortality based on catches in the data file. Users might want to adjust the maximum fishing mortality based on their scenarios.

Data file

No data file is needed for the EM. The data_expval.ss file produced when executing the OM contains the expected values of the OM population dynamics. ss3sim provides three functions which carry out the random sampling process and generate .dat files to be used in the EM. See the Introduction vignette vignette("introduction", package = "ss3sim") for more details.

Forecast file

Nothing is changed in the forecast file from the OM.

Starter file

The names of the data and control files are specified and the maximum phase for estimation is set to 100.

Author(s)

Kelli F. Johnson

Examples

```
create_em()
# The necessary files are in the following folder
dir(file.path(getwd(), "new-em"))
# Clean up your directory
unlink(file.path(getwd(), "new-em"), recursive = TRUE)
```

create_logo *Create the ss3sim logo*

Description

Generate and save, if outfile is provided, the ss3sim logo using the built-in data.

Usage

```
create_logo(outfile = NULL)
```

Arguments

`outfile`	A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

Value

A png file or a graphics device with the logo used for the `ss3sim` project.

Author(s)

Kelli F. Johnson

Examples

```
ss3sim:::create_logo()
grDevices::dev.off()
```

`facet_form` *Helper function for building a ggplot facet*

Description

Used internally by the plotting functions to create faceting formulas.

Usage

```
facet_form(horiz = NULL, horiz2 = NULL, vert = NULL, vert2 = NULL)
```

Arguments

`horiz, horiz2`	A character string denoting which column to use as the first (`horiz`) and second (`horiz2`) level of faceting in the horizontal direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the horizontal space.
`vert, vert2`	A character string denoting which column to use as the first (`vert`) and second (`vert2`) level of faceting in the vertical direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the vertical space.

Value

A formula which can be used in `ggplot2::facet_grid()` or `NULL` if all arguments are `NULL`.

Author(s)

Cole Monnahan

fill_across *Fill in matrix across rows of weight-at-age data by interpolation*

Description

Function that fills in matrix across rows of wtatage data by interpolation Missing Rows are then backfilled

Usage

```
fill_across(mat, minYear, maxYear)
```

Arguments

mat	A matrix
minYear	Minimum year
maxYear	Maximum year

Author(s)

Peter Kuriyama and Allan Hicks

See Also

- sample_lcomp()
- sample_agecomp()
- sample_wtatage()

find_position *Find integer reference to fleet names*

Description

Find the position of each desired value, i.e., x, in a vector of strings. Builds on match() by allowing x to be a combination of strings to be matched and known positions.

Usage

```
find_position(x, table)
```

Arguments

x	A vector of strings and/or integers to be matched.
table	A vector of strings. If the character strings includes values that can be coerced to integers, they must be in a matching position in the vector. For example, table = c("3","4","1") will confuse the function because x = 1 will produce a return value of 3 rather than 1.

Value

An integer vector indicating the positions of x in table. Same as match(), if x[i] is found to be equal to table[j], then the value returned in the i-th position of the integer vector is j. The smallest value of j, i.e., the first match, is always returned.

find_position() differs from match() in three ways. First, values of x that are not found are removed. Thus, the length of the integer vector has the potential to be shorter than the length of x. Second, x can contain a mix of integer positions that are already known and strings to be found. Third, table cannot include integers that do not match their position. See the specifications for table for more details.

Author(s)

Kelli F. Johnson

See Also

See match() for a more formal version of find_position() that returns an integer vector the same length as x.

Examples

```
# Standard use
find_position(c("sad", 1), c("happy", "sad"))
# Incorrect use
find_position(c("sad", 2), c("happy", "sad", "2"))
```

get_bin *Get Stock Synthesis binary/executable location*

Description

Get Stock Synthesis binary/executable location

Usage

```
get_bin(bin_name = "ss3")
```

Arguments

bin_name	A string providing the name of the binary/executable without the extension. The default is "ss3", which is the name of the executable that is saved in ss3sim on GitHub.

Value

A string providing the full path to a Stock Synthesis binary. If using the GitHub version of ss3sim, this will be an internal binary. Otherwise, get_bin() searches for a version of the binary in your path. See the ss3sim vignette fore more information.

Author(s)

Sean C. Anderson

Examples

```
## Not run:
get_bin()

## End(Not run)
```

---

get_compfit *Get summaries of fits to composition data from report file list*

Description

Extract the summary of fits to composition data, where the sections are structured similarly for each type of data in the report file.

Usage

```
get_compfit(report.file, name)
```

Arguments

report.file	An SS_output list for a model (operating model or estimation model).
name	A character string that matches the element of report.file that you wish to extract, e.g., "Length_Comp_Fit_Summary".

---

get_model_folder *Get the folder location of an included Stock Synthesis model configuration*

Description

This function returns the location of one of the built-in model configurations.

Usage

```
get_model_folder(folder_name)
```

Arguments

folder_name	The model folder name. One of "cod-om", "cod-em", "fla-om", "fla-em", "sar-om", "sar-em" representing cod, flatfish, and sardine-like model configurations and operating (om) and estimating model (em) varieties. See the **ss3sim** paper or vignette for further details.

Value

A character object showing the location of the appropriate model configuration folder in the package `extdata` folder.

Examples

```
get_model_folder("cod-em")
```

get_nll_components *Get negative log likelihood (NLL) values from a report file list*

Description

Names of the available NLL components will depend on the version of the model. Names are native to the estimation framework and all available components are extracted.

Usage

```
get_nll_components(report.file)
```

Arguments

`report.file`	An `SS_output` list for a model (operating model or estimation model).

Value

A vector of named numeric values, where `"NLL_"` is appended to the names in the `report.file`.

Author(s)

Merrill Rudd

get_recdevs *Return a set of recruitment deviations*

Description

This function returns a set of pseudo-random recruitment deviations based on an iteration number. Given the same iteration number the function will return the same recruitment deviations. The deviations are standard normal. I.e., they have a mean of 0 and a standard deviation of 1.

Usage

```
get_recdevs(iteration, n, seed = 21)
```

Arguments

iteration	The iteration number. This is used as an ID to set the random number seed.
n	The length of the vector returned.
seed	An integer value to pass to `set.seed()`.

Value

A vector of standard normal recruitment deviations.

Examples

```
get_recdevs(1, 10)
get_recdevs(1, 10)
get_recdevs(2, 10)
```

`get_results_all` *Extract Stock Synthesis simulation output*

Description

This high level function extracts results from Stock Synthesis model runs. Give it a directory which contains directories for different "scenario" runs, within which are iterations. It writes two data.frames to file: one for single scalar values (e.g., MSY) and a second that contains output for each year of the same model (timeseries, e.g., biomass(year)). These can always be joined later.

Usage

```
get_results_all(
  directory = getwd(),
  overwrite_files = FALSE,
  user_scenarios = NULL,
  type = c("long", "wide"),
  filename_prefix = "ss3sim"
)
```

Arguments

directory	The directory which contains scenario folders with results.
overwrite_files	A switch to determine if existing files should be overwritten, useful for testing purposes or if new iterations are run.
user_scenarios	A character vector of scenarios that should be read in. Default is `NULL`, which indicates find all scenario folders in `directory`.
type	A character string specifying if you want the results to be written to the disk and returned as a long or wide data frame, where the default is `"long"`.
filename_prefix	A character string specifying a prefix to append to the filename. Defaults to "ss3sim".

Value

Returns a list of 3 dataframes: scalar, ts, and dq. Creates two .csv files in the current working directory, where the names of those files are based on `filename_prefix` and the default leads to the following: `ss3sim_ts.csv` and `ss3sim_scalar.csv`.

Author(s)

Cole Monnahan, Merrill Rudd, Kathryn L. Doering

See Also

Other get-results: `get_results_derived()`, `get_results_scalar()`, `get_results_scenario()`, `get_results_timeseries()`

`get_results_derived` *Extract time series from a model run with the associated standard deviation.*

Description

Extract time series from an `r4ss::SS_output()` list from a model run. Returns a data.frame of the results for spawning stock biomass (SSB), recruitment, forecasts, and effort by year.

Usage

```
get_results_derived(report.file)
```

Arguments

`report.file`	An `SS_output` list for a model (operating model or estimation model).

Author(s)

Kelli F. Johnson

See Also

Other get-results: `get_results_all()`, `get_results_scalar()`, `get_results_scenario()`, `get_results_timeseries(`

`get_results_iter` *Get results for 1 iteration*

Description

Get results for 1 iteration

Usage

```
get_results_iter(dir_1_iter = NULL, mod_dirs = NULL, iter_name = NULL)
```

Arguments

`dir_1_iter`	The full or relative path to the Stock Synthesis iteration folder. Assumed to contain multiple model folders that contain "om" or "em" (not case sensitive) somewhere in the model file name. If specified, mod_dirs need not be specified.
`mod_dirs`	The full or relative path to the Stock Synthesis model folders as a vector of characters. If specified, dir_1_iter need not be specified.
`iter_name`	Name of the iteration, which will be appended to the dataframes . Defaults to NULL, in which case the iter_name will be the folder name of dir_1_iter or the folder name 1 level up from the first mod_dirs specified

Value

A list of 3 data frames called scalar, timeseries, and derived (for derived quantities). These lists contain information for multiple model runs (estimation models and operating models) for 1 iteration.

Author(s)

Kathryn L. Doering

`get_results_mod` *Get results for 1 model run*

Description

Get results for 1 model run

Usage

```
get_results_mod(dir = getwd(), is_EM = NULL, is_OM = NULL)
```

Arguments

`dir`	The full or relative path to the Stock Synthesis model file folder. If not specified, uses the working directory.
`is_EM`	Is this an estimation model? Defaults to NULL, which will look for the letters "em" (lower or uppercase) to decide if this is an estimation model or operating model.
`is_OM`	Is this an operating model? Defaults to NULL, which will look for the letters "om" (lower or uppercase) to decide if this is an estimation model or operating model.

Value

A list of 3 data frames called scalar, timeseries, and derived (for derived quantities). These data frames contain results for 1 model run.

Author(s)

Kathryn L. Doering

`get_results_scalar` *Extract scalar quantities from a model run.*

Description

Extract scalar quantities from an `r4ss::SS_output()` list from a model run. Returns a data.frame of the results (a single row) which can be rbinded later.

Usage

```
get_results_scalar(report.file)
```

Arguments

`report.file`	An `SS_output` list for a model (operating model or estimation model).

Author(s)

Cole Monnahan; Merrill Rudd

See Also

Other get-results: `get_results_all()`, `get_results_derived()`, `get_results_scenario()`, `get_results_timeseries()`

get_results_scenario *Extract Stock Synthesis simulation results for one scenario*

Description

Extract results from all iterations inside a supplied scenario folder. The function writes the following .csv files to the scenario folder

1. scalar metrics with one value per iteration (e.g. R_0, h),
2. a timeseries data ('ts') which contains multiple values per iteration (e.g. SSB_y for a range of years y), and
3. residuals on the log scale from the surveys across all iterations; this functionality is currently disabled and not tested.

Usage

```
get_results_scenario(scenario, directory = getwd(), overwrite_files = FALSE)
```

Arguments

`scenario` A single character giving the scenario from which to extract results.

`directory` The directory which contains the scenario folder.

`overwrite_files` A boolean (default is `FALSE`) for whether to delete any files previously created with this function. This is intended to be used if iterations were added since the last time it was called, or any changes were made to this function.

Author(s)

Cole Monnahan and Kathryn L. Doering

See Also

`get_results_all()` loops through these .csv files and combines them together into a single "final" dataframe.

Other get-results: `get_results_all()`, `get_results_derived()`, `get_results_scalar()`, `get_results_timeseries()`

get_results_timeseries

Return the time series information from an iteration

Description

Extract and return time series from an r4ss::SS_output() list, that is read in from the estimation method of a single iteration. The main time series information is included but no information about the uncertainty of those measurements is available. See the derived quantities for uncertainty.

Usage

```
get_results_timeseries(report.file)
```

Arguments

report.file An SS_output list for a model (operating model or estimation model).

Details

Information about both season and area are included in the data frame. For values that have no associated season or area, i.e., are summary values over all areas and seasons, the values are repeated for each area/season combination within a given year. For example, the recruitment deviation is for all areas and is thus repeated in each row across areas for a given year.

Value

A data frame with the following columns:

- year
- Area
- Seas
- Bio_smry
- SpawnBio
- Recruit_0
- retainB_0-9+
- retainN_0-9+
- deadB_0-9+
- deadN_0-9+
- F_0-9+
- SPRratio
- rec_dev
- raw_rec_dev

Author(s)

Cole Monnahan

See Also

Other get-results: `get_results_all()`, `get_results_derived()`, `get_results_scalar()`, `get_results_scenario()`

`get_scenarios` *Identify scenarios in* `directory`

Description

Find scenario directories, where it is known if a directory was derived from ss3sim and contains iterations of the operating and estimating models if there are directories with numeric names that contain om and/or em directories, i.e., "1", "2", "3", ..., "100".

Usage

```
get_scenarios(directory = getwd(), full = FALSE)
```

Arguments

`directory`	The directory or vector of directories that you want to search for scenarios. The search is recursive, and thus, it is in one's best interest to provide a shorter path name rather than one high up in the call stack.
`full`	A logical entry. If `TRUE`, the full path name is returned, which can be helpful if a vector of directories is supplied to `directory`, otherwise it is impossible to know which scenarios are located where. If `FALSE`, a vector of names is returned. This is the default behavior.

Value

A character vector of names of directories that contain output from ss3sim. Full paths are only provided if `full = TRUE`.

Author(s)

Merrill Rudd

get_sigmar *Get Variability About Recruitment Deviations (σ_R)*

Description

Use the name of the operating model to open the ctl file and obtain the INIT value for sigmaR (recruitment deviations sigma)

Usage

```
get_sigmar(om)
```

Arguments

`om`	The name of the operating model, which should be the prefix of the `.ctl` file, e.g., "myOM". A full directory can be specified with the the prefix of the file name but leaving off the '.ctl' portion.

Author(s)

Kelli F. Johnson

get_success *Determine if a Stock Synthesis run was successful*

Description

Use the presence of files generated by Stock Synthesis to determine if the run was successful or not. There are two levels of success that must be determined if the run was meant to include estimating the hessian.

Usage

```
get_success(dir)
```

Arguments

`dir`	A character string specifying the path to the folder with the results from the run of the Stock Synthesis model.

Value

A named vector with two values, `ran` and `hess`, (1) a zero or one for the presence of the `Report.sso` file and (2) a zero or one for the presence of a positive definite hessian matrix.

Author(s)

Kelli F. Johnson

make_df — *Make a list of lists with dataframe components into a dataframes*

Description

Bind together list of list components with the same name

Usage

```
make_df(list_name, list_df)
```

Arguments

`list_name`	A name to subset from iter_list
`list_df`	A list of dataframes. These need not have the same column names, as this function will fill in with NAs.

Value

A dataframe

Author(s)

Kathryn L. Doering

plot_boxplot — *Plot results of a simulation as boxplots*

Description

Generate boxplots using `ggplot2::ggplot()` to visualize outliers and central tendencies.

Usage

```
plot_boxplot(
  data,
  x,
  y,
  horiz = NULL,
  horiz2 = NULL,
  vert = NULL,
  vert2 = NULL,
  relative.error = FALSE,
  axes.free = TRUE,
  print = TRUE,
  fill = NA
)
```

Arguments

`data` A valid data frame containing scalar or timeseries values from a **ss3sim** simulation. That data are generated from `get_results_all`.

`x` A character string denoting which column to use as the x variable. For time-series data, setting `x = "year"` leads to a time-series plot.

`y` A character string denoting which column to use as the y variable. Must be a numeric column.

`horiz, horiz2` A character string denoting which column to use as the first (`horiz`) and second (`horiz2`) level of faceting in the horizontal direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the horizontal space.

`vert, vert2` A character string denoting which column to use as the first (`vert`) and second (`vert2`) level of faceting in the vertical direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the vertical space.

`relative.error` Boolean for whether the y-axis scale should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1, 1)`, the y-axis label is changed automatically, and a black, dashed line at `y=0` is added. The argument can also accept a color entry if you wish the line to be something other than black. E.g., `"red"` will add a red dashed line at zero as well as fix the y-axis limits.

`axes.free` Boolean for whether the y-axis scales should be free in `facet_grid`.

`print` A logical for whether the plot is printed or not.

`fill` A character string that represents a single color that will be used to fill the boxplots. The default value of `NA` leads to unfilled boxplots.

Details

Median, hinges, and whiskers as well as outliers are displayed to summarize the data. The lower and upper hinges are the first and third quantiles (i.e., 25th and 75th percentiles). The upper and lower whiskers are 1.5*inner-quartile range, i.e., the distance between the first and third quartiles. Outliers are those points that lie beyond the whiskers. These explanations are detailed in `ggplot2::geom_boxplot()`.

Values of `NA` are removed prior to plotting such that the typical error message from `ggplot2::ggplot()` is not printed to the screen.

The **ss3sim** plotting functions are simply wrappers for **ggplot2** code, specific to the output from **ss3sim** `get_results_all()` objects. They are designed to quickly explore simulation output, rather than produce publication-level figures. The functions use arguments passed as characters that refer to columns of `data`. Scalar plots requires a value for `x`; while, for time-series plots, `x = "year"` will be necessary.

Note that there are some subtle differences between the functions. Boxplots cannot have a color mapped to them like points or lines, and thus, `color` is not a valid argument. The time-series point and line plots are grouped internally by 'ID', which is a combination of scenario and iteration and will be automatically added to the data set if not already present.

Output

These functions print the `ggplot` object, but also return it invisibly for saving or printing again later. For example, you could save the `ggplot` object and add a custom theme or change an axis label before printing it.

Author(s)

Cole Monnahan

Examples

```
# Plot scalar values
data("scalar_dat", package = "ss3sim")
re <- calculate_re(scalar_dat)
## Not run:
plot_boxplot(re,
  x = "E", y = "depletion_re", horiz = "D",
  relative.error = TRUE
)

## End(Not run)
rm(re)
# Merge scalar and time-series values to plot time series with color
# Be patient, the time-series boxplots take some time.
data("ts_dat", package = "ss3sim")
ts_dat[, "model_run"] <- factor(ts_dat[, "model_run"],
  levels = c("om", "em")
)
## Not run:
plot_boxplot(ts_dat,
  x = "year", y = "SpawnBio",
  horiz = "scenario", vert = "model_run"
)

## End(Not run)
```

`plot_cummean` *Plot the cumulative mean for a parameter*

Description

Plot the cumulative mean for a parameter

Usage

```
plot_cummean(
  data,
  var,
  order_var = "iteration",
  group = NULL,
  use_facet = FALSE
)
```

Arguments

`data`	A valid data frame containing scalar or time series values from a **ss3sim** simulation. That data are generated from `get_results_all()`.
`var`	The column name of the parameter in data of which to plot cumulative mean. A string.
`order_var`	A column to order the data before calculating the cumulative mean
`group`	A column in data to group the data together before calculating the cumulative mean
`use_facet`	Should the group be used to create facets? If TRUE, facets are created; If FALSE, grouping will be done by making different color lines in the same plot.

Value

A list containing the ggplot object and the data used to make it

Examples

```
data("scalar_dat", package = "ss3sim")
obj <- plot_cummean(scalar_dat[scalar_dat$model_run == "em", ],
  "VonBert_K_Fem_GP_1",
  group = "scenario",
  use_facet = TRUE
)
# obj$plot
# obj$data
rm(obj)
# group can also be left NULL if only plotting a single scenario.
# it is recommended to set use_facet FALSE in this case.
obj2 <- plot_cummean(
  scalar_dat[
    scalar_dat$scenario == unique(scalar_dat$scenario)[1] &
      scalar_dat$model_run == "em",
  ],
  var = "VonBert_K_Fem_GP_1",
  group = NULL,
  use_facet = FALSE
)
# obj2$plot
# obj2$data
rm(obj2)
```

`plot_lines` *Plot time-series values as lines*

Description

Plot time-series values as lines

Usage

```
plot_lines(
  data,
  x = "year",
  y,
  horiz = NULL,
  horiz2 = NULL,
  vert = NULL,
  vert2 = NULL,
  relative.error = FALSE,
  color = NULL,
  axes.free = TRUE,
  print = TRUE
)
```

Arguments

`data`	A valid data frame containing scalar or timeseries values from a **ss3sim** simulation. That data are generated from `get_results_all`.
`x`	A character string denoting which column to use as the x variable. For time-series data, setting `x = "year"` leads to a time-series plot.
`y`	A character string denoting which column to use as the y variable. Must be a numeric column.
`horiz`, `horiz2`	A character string denoting which column to use as the first (`horiz`) and second (`horiz2`) level of faceting in the horizontal direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the horizontal space.
`vert`, `vert2`	A character string denoting which column to use as the first (`vert`) and second (`vert2`) level of faceting in the vertical direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the vertical space.
`relative.error`	Boolean for whether the y-axis scale should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1, 1)`, the y-axis label is changed automatically, and a black, dashed line at `y=0` is added. The argument can also accept a color entry if you wish the line to be something other than black. E.g., `"red"` will add a red dashed line at zero as well as fix the y-axis limits.
`color`	A character string denoting which column to use to map color. Not valid for boxplot functions. Useful for looking at EM performance criteria against other dimensions of the EM or OM. See example below for how to merge in a metric from a scalar dataset to a ts dataset.

`axes.free` Boolean for whether the y-axis scales should be free in `facet_grid`.

`print` A logical for whether the plot is printed or not.

Details

The **ss3sim** plotting functions are simply wrappers for **ggplot2** code, specific to the output from **ss3sim** `get_results_all()` objects. They are designed to quickly explore simulation output, rather than produce publication-level figures. The functions use arguments passed as characters that refer to columns of `data`. Scalar plots requires a value for `x`; while, for time-series plots, `x = "year"` will be necessary.

Note that there are some subtle differences between the functions. Boxplots cannot have a color mapped to them like points or lines, and thus, `color` is not a valid argument. The time-series point and line plots are grouped internally by 'ID', which is a combination of scenario and iteration and will be automatically added to the data set if not already present.

Output

These functions print the `ggplot` object, but also return it invisibly for saving or printing again later. For example, you could save the `ggplot` object and add a custom theme or change an axis label before printing it.

Author(s)

Cole Monnahan

Examples

```
data("scalar_dat", "ts_dat", package = "ss3sim")
# Merge in max_grad, a performance metric, to use for color
re <- merge(
  by = "ID",
  calculate_re(ts_dat, add = FALSE),
  calculate_re(scalar_dat, add = FALSE)[, c("ID", "max_grad")]
)
## Not run:
plot_lines(re,
  y = "SpawnBio_re", horiz = "D", vert = "E",
  relative.error = TRUE, color = "max_grad"
)

## End(Not run)
```

plot_points *Plot results of a simulation as a scatterplot*

Description

Generate a scatterplot using ggplot2::ggplot to visualize the relationship between two continuous variables.

Usage

```
plot_points(
  data,
  x,
  y,
  horiz = NULL,
  horiz2 = NULL,
  vert = NULL,
  vert2 = NULL,
  jitter.height = 0,
  jitter.width = 0,
  color = NULL,
  relative.error = FALSE,
  axes.free = TRUE,
  print = TRUE
)
```

Arguments

`data`	A valid data frame containing scalar or timeseries values from a **ss3sim** simulation. That data are generated from `get_results_all`.
`x`	A character string denoting which column to use as the x variable. For time-series data, setting x = `"year"` leads to a time-series plot.
`y`	A character string denoting which column to use as the y variable. Must be a numeric column.
`horiz`, `horiz2`	A character string denoting which column to use as the first (`horiz`) and second (`horiz2`) level of faceting in the horizontal direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the horizontal space.
`vert`, `vert2`	A character string denoting which column to use as the first (`vert`) and second (`vert2`) level of faceting in the vertical direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the vertical space.
`jitter.height`, `jitter.width`	Parameters for `ggplot2::position_jitter()` that specify the vertical and horizontal spread added to points. Where, added values are both positive and negative, so the total spread is twice the value specified here. If `NULL`, the spread will be 40% of the resolution of the data; this means the jitter values will occupy

80% of the implied bins. Categorical data is aligned on the integers, so a width or height of 0.5 will spread the data so it's not possible to see the distinction between the categories. The default within ss3sim is to not jitter, i.e., a spread of 0.0.

`color` A character string denoting which column to use to map color. Not valid for boxplot functions. Useful for looking at EM performance criteria against other dimensions of the EM or OM. See example below for how to merge in a metric from a scalar dataset to a ts dataset.

`relative.error` Boolean for whether the y-axis scale should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1, 1)`, the y-axis label is changed automatically, and a black, dashed line at `y=0` is added. The argument can also accept a color entry if you wish the line to be something other than black. E.g., `"red"` will add a red dashed line at zero as well as fix the y-axis limits.

`axes.free` Boolean for whether the y-axis scales should be free in `facet_grid`.

`print` A logical for whether the plot is printed or not.

Details

Points are placed on the figure using the width setting in `ggplot2::position_jitter()` that defaults to 40% resolution of the data, meaning that the jitter values will occupy 80% of the implied bins. The previous information was found in the documentation for `ggplot2::position_jitter()`.

Values of `NA` are removed prior to plotting such that the typical error message from **ggplot2** is not printed to the screen.

The **ss3sim** plotting functions are simply wrappers for **ggplot2** code, specific to the output from **ss3sim** `get_results_all()` objects. They are designed to quickly explore simulation output, rather than produce publication-level figures. The functions use arguments passed as characters that refer to columns of `data`. Scalar plots requires a value for `x`; while, for time-series plots, `x = "year"` will be necessary.

Note that there are some subtle differences between the functions. Boxplots cannot have a color mapped to them like points or lines, and thus, `color` is not a valid argument. The time-series point and line plots are grouped internally by 'ID', which is a combination of scenario and iteration and will be automatically added to the data set if not already present.

Output

These functions print the `ggplot` object, but also return it invisibly for saving or printing again later. For example, you could save the `ggplot` object and add a custom theme or change an axis label before printing it.

Author(s)

Cole Monnahan

Examples

```
# Plot scalar values
data("scalar_dat", package = "ss3sim")
```

```
re <- calculate_re(scalar_dat)
## Not run:
plot_points(re,
  x = "E", y = "depletion_re", horiz = "D",
  color = "max_grad", relative.error = TRUE
)

## End(Not run)
rm(re)
# Merge scalar and time-series values to plot time series with color
data("ts_dat", package = "ss3sim")
re <- merge(
  by = "ID",
  calculate_re(ts_dat, add = FALSE),
  calculate_re(scalar_dat, add = FALSE)[, c("ID", "max_grad")]
)
## Not run:
plot_points(re,
  x = "year", y = "SpawnBio_re",
  horiz = "scenario", color = "max_grad", relative.error = TRUE
)

## End(Not run)
rm(re)
```

plot_ss3sim *Base plot for ss3sim data*

Description

Use ggplot2::ggplot to plot data from ss3sim simulation.

Usage

```
plot_ss3sim(
  data,
  x,
  y,
  color = NULL,
  relative.error = FALSE,
  axes.free = TRUE,
  print = TRUE,
  horiz = NULL,
  horiz2 = NULL,
  vert = NULL,
  vert2 = NULL
)
```

Arguments

`data` A valid data frame containing scalar or timeseries values from a **ss3sim** simulation. That data are generated from `get_results_all`.

`x` A character string denoting which column to use as the x variable. For time-series data, setting `x = "year"` leads to a time-series plot.

`y` A character string denoting which column to use as the y variable. Must be a numeric column.

`color` A character string denoting which column to use to map color. Not valid for boxplot functions. Useful for looking at EM performance criteria against other dimensions of the EM or OM. See example below for how to merge in a metric from a scalar dataset to a ts dataset.

`relative.error` Boolean for whether the y-axis scale should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1, 1)`, the y-axis label is changed automatically, and a black, dashed line at `y=0` is added. The argument can also accept a color entry if you wish the line to be something other than black. E.g., `"red"` will add a red dashed line at zero as well as fix the y-axis limits.

`axes.free` Boolean for whether the y-axis scales should be free in `facet_grid`.

`print` A logical for whether the plot is printed or not.

`horiz, horiz2` A character string denoting which column to use as the first (`horiz`) and second (`horiz2`) level of faceting in the horizontal direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the horizontal space.

`vert, vert2` A character string denoting which column to use as the first (`vert`) and second (`vert2`) level of faceting in the vertical direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the vertical space.

Details

The **ss3sim** plotting functions are simply wrappers for **ggplot2** code, specific to the output from **ss3sim** `get_results_all()` objects. They are designed to quickly explore simulation output, rather than produce publication-level figures. The functions use arguments passed as characters that refer to columns of `data`. Scalar plots requires a value for `x`; while, for time-series plots, `x = "year"` will be necessary.

Note that there are some subtle differences between the functions. Boxplots cannot have a color mapped to them like points or lines, and thus, `color` is not a valid argument. The time-series point and line plots are grouped internally by 'ID', which is a combination of scenario and iteration and will be automatically added to the data set if not already present.

Output

These functions print the `ggplot` object, but also return it invisibly for saving or printing again later. For example, you could save the `ggplot` object and add a custom theme or change an axis label before printing it.

Author(s)

Cole Monnahan

`profile_fmsy` *Determine Fmsy for a given operating model*

Description

Runs an operating model over a range of fishing mortality, F, levels to determine the F at maximum sustainable yield, F_{MSY}.

Usage

```
profile_fmsy(
  om_in,
  results_out,
  start = 0,
  end = 1.5,
  by_val = 0.01,
  verbose = FALSE
)
```

Arguments

`om_in`	A full or relative path to a directory that contains an **ss3sim** operating model.
`results_out`	A full or relative path to a directory where the results will be saved. The directory will be created if it does not already exist.
`start, end`	A single numerical value for each argument specifying the lowest and highest fishing levels that you want to explore for the fishing fleet in your model.
`by_val`	Interval at which fishing mortality will be incremented between `start` and `end` using `seq(start, end, by_val)`.
`verbose`	When `TRUE` messages will be returned from the function. Often useful for debugging. The default is `FALSE`.

Details

`profile_fmsy()` runs the operating model with a constant level of fishing for each year and extracts the expected catch in the terminal year. It is assumed that the model time series is long enough for the population to come to equilibrium, and thus, the catch in the terminal year is equivalent to equilibrium catch.

If the function is run with `verbose = TRUE`, which is not the default, the coefficient of variations of the catches in the terminal years of the model will be printed to the screen. Here, terminal is defined as half as many years as there are ages in the population dynamics of your model. Thus, if the population plus group starts at age twenty, the standard deviation of the last ten years of catch divided by the mean catch over that same time will be printed to the screen for each model that is ran. For the default cod model provided within the package, the CV is less than 1e-04 for all explored levels of fishing mortality.

Ensure that the argument `om_in` leads to an operating model that is configured for use within **ss3sim**. For example, the F type must allow for an input vector of Fs rather than catches, along with other specifications.

Value

A data frame of catch by fishing mortality is returned invisibly and saved to the disk along with a figure, Fmsy.pdf.

Examples

```
## Not run:
d <- system.file("extdata", "models", "cod-om", package = "ss3sim")
fmsy.val <- profile_fmsy(
  om_in = d, results_out = "fmsy",
  start = 0.1, end = 0.2, by_val = 0.05
)
# cleanup
unlink("fmsy", recursive = TRUE)

## End(Not run)
```

---

replace_x *Replace a NULL value with NA in a list*

Description

Replace items with a zero length in a list with the value supplied in the argument replacement. Useful for scenarios where NULL is sometimes a valid input from legacy code. Nested lists can behave badly when they have a NULL entry, for example when converting to a tibble::tibble(), it will be unnamed.

Usage

```
replace_x(x, replacement = NA_integer_)
```

Arguments

x	An object that potentially has a length of zero and you wish it to be an actual value.
replacement	The value you would like to use to replace items with a length of zero. For example, the default NA_integer_ will replace all NULL values with NA. Other options for this argument could be NA_character_.

Value

The object x is returned with some items replaced. If the input object was of zero length, then the replacement parameter will be returned instead.

Author(s)

Amanda from stack overflow

Examples

```
employees <- list(
  list(
    id = 1,
    dept = "IT",
    age = 29,
    sportsteam = "softball"
  ),
  list(
    id = 2,
    dept = "IT",
    age = 30,
    sportsteam = NULL
  ),
  list(
    id = 3,
    dept = "IT",
    age = 29,
    sportsteam = "hockey"
  ),
  list(
    id = 4,
    dept = NULL,
    age = 29,
    sportsteam = "softball"
  )
)
# Meat of the example here!
## Not run:
do.call(rbind, lapply(employees, rbind)) %>%
  data.frame() %>%
  purrr::modify_depth(2, replace_x)

## End(Not run)
```

run_ss3sim *Master function to run a set of simulations using* **ss3sim**

Description

This is the main high-level wrapper function used to run a set of **ss3sim** simulations. The data frame passed to `simdf` is parsed into a list and used to control `ss3sim_base()`. Alternatively, you can call `ss3sim_base()` directly with your own lists.

Usage

```
run_ss3sim(
  iterations,
  simdf = NULL,
  parallel = FALSE,
  parallel_iterations = FALSE,
  ...
)
```

Arguments

`iterations` A numeric vector specifying which iterations are desired. For example `1:100`. The same number of iterations will be ran for each scenario. If any iterations already have a folder from a previous run, they will be skipped even if they do not contain viable results.

`simdf` A data frame of instructions with one row per scenario. See `setup_scenarios_defaults()` for default values that will be used for a generic simulation to get you started. These default values will only work with the stored cod model because some of the columns in `simdf` need to have values that match the fleet structure of the operating model. If you are not using the default cod model, please remember to add `om_dir` and `em_dir` columns to `simdf` with file paths to the locations of your operating model and estimation method. Essentially, `simdf` is a way to pass scenario-specific information to the arguments of `ss3sim_base()`, whereas the `...` method will only work for things like `seed` that are universal to all scenarios in a simulation.

`parallel` A logical argument that controls whether the scenarios are run in parallel. You will need to register multiple cores first with a package such as **doParallel** and have the **foreach** package installed. For example, the following code will register two cores and must be called before running `run_ss3sim()`:

```
library(doParallel)
cl <- makeCluster(2)
registerDoParallel(cl)
```

`parallel_iterations`
A logical argument specifying if you wish to run iterations in parallel. If you set `parallel = TRUE` and `parallel_iterations = TRUE` then iterations for a given scenario will be sent to multiple processors. All iterations for a given scenario must finish before the next scenario is started. This will be useful if you want to run one scenario fast or if you want to be able to look at the results for each scenario as they finish in another instance of R. The argument will be ignored if `parallel = FALSE`.

`...` Anything else to pass to `ss3sim_base()`. This could include `bias_adjust`. Also, you can pass additional options to the executable through the argument `extras`.

Details

The operating model folder, which is passed as a file path using `simdf[["om_dir"]]`, should contain the following files:

- `forecast.ss`,

- yourmodel.ctl,
- yourmodel.dat,
- ss.par, and
- starter.ss. The files should be the formatted versions that are returned from Stock Synthesis after the model is optimized, i.e., .ss_new files. It is important to use these formatted files because many functions in **ss3sim** and **r4ss** depend on the location of keywords present in the comments and other standardized formatting. Once you have these files from a successfully optimized model, rename the .ss_new files to match the names listed above, though you can change yourmodel to whatever name is listed for the control and data files in starter.ss. The estimation model folder should also contain these files, except ss.par and yourmodel.dat files, which are unnecessary. See the vignette titled modifying-models for details on modifying an existing Stock Synthesis model to run with **ss3sim**. Alternatively, consider modifying the built-in model configuration based on north sea cod.

Note that due to the way that Stock Synthesis is being used as an OM, you may see the following error from ADMB in the console: Error – base = 0 in function prevariable& pow(const prevariable& v1, CGNU_DOUBLE u) However, this is not a problem because ADMB is not used to optimize the OM, and thus, the error can safely be ignored.

Value

The output will appear in your current R working directory. Folders will be named based on the "scenario" column of simdf or based on the date-time stamp (i.e., mmddhhmmss) generated automatically at the start of the simulation. The resulting folders will look like the following if you run your simulation at noon on January 01:

- 0101120000/1/om
- 0101120000/1/em
- 0101120000/2/om
- ...

Author(s)

Sean C. Anderson

See Also

ss3sim_base() can be called directly by passing lists to each individual argument rather than using the data-frame approach of run_ss3sim(simdf =). The lists correspond to each function called by ss3sim_base(). Each element is itself a list of arguments for the given function. Either way allows users to pass arguments to each of the change_*() or sample_*() functions. Note that if you do not include an argument, then ss3sim_base() will assume the value of that argument is NULL.

Examples

```
## Not run:
# A run with deterministic process error for model checking
# by passing user_recdevs to ss3sim_base through run_ss3sim:
recdevs_det <- matrix(0, nrow = 101, ncol = 2)
```

```
df <- data.frame(setup_scenarios_defaults(),
  "scenarios" = "determinate"
)
run_ss3sim(
  iterations = 1:2, simdf = df,
  bias_adjust = FALSE, user_recdevs = recdevs_det
)
get_results_all(user_scenarios = "determinate", overwrite = TRUE)
ts <- utils::read.csv("ss3sim_ts.csv")
expect_equivalent(
  unlist(ts$rec_dev[ts$year %in% 1:10 & ts$iteration == 2]),
  recdevs_det[1:10, 2]
)

## End(Not run)
```

sample_agecomp *Sample age compositions from a Stock Synthesis data file*

Description

Extract age-composition data from a .ss_new data file and sample the data. It is assumed that the composition data will be expected values as written by Stock Synthesis in the second section of the data file, but one can also sample input data. The resulting age-composition data are assumed to represent observed age composition and will overwrite the age data in dat_list, which is returned invisibly. The data file can also be written to the disk, if a file path is provided to outfile, and used as simulated data by an estimation model.

Usage

```
sample_agecomp(
  dat_list,
  outfile = NULL,
  fleets,
  Nsamp,
  years,
  cpar = 1,
  ESS = NULL,
  keep_conditional = TRUE,
  ...
)
```

Arguments

dat_list A Stock Synthesis data list object as read in from SS_readdat. Be sure to correctly specify which section of the data file you want to work with when reading it in using the section argument. Where, section = 1 reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

`fleets` *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.

`Nsamp` *A numeric list of the same length as `fleets`. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to `Nsamp=NULL`.

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`cpar` A numeric value or vector the same length as `fleets` controlling the variance of the Dirichlet distribution used for sampling. A value of `1` leads to the same standard deviation as a multinomial of the given `Nsamp`, `2` indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion. `NULL` or `NA` for a given fleet will lead to no dispersion.

`ESS` The final effective sample size (ESS) associated with the simulated data. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, `NULL`, leads to the true (internally calculated) #' ESS being used, which is `Nsamp` for the multinomial case or given by the formula under `cpar` for the Dirichlet case. At least one value must be provided for each fleet or a vector of year-specific values can be used for any given fleet. The argument accepts a list with entries, either a single integer or a vector of integers, for each fleet.

`keep_conditional`

A logical if conditional age-at-length data should be kept or removed entirely from the data file. `sample_agecomp` only works on the age-composition data and not on the conditional age-at-length data. To sample the conditional data, set `keep_conditional` to `TRUE` and use `sample_calcomp()`.

`...` Any argument you want to be a column in the new data frame of composition data. All extra arguments should be named columns in `data`. Each argument needs to be a list of length `length(fleets)`. Or, you can use a single value that will be repeated for each combination of fleet, year, ... in your data.

Value

A modified `.dat` file if `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Author(s)

Cole Monnahan and Kotaro Ono

See Also

Other sampling functions: clean_data(), sample_calcomp(), sample_catch(), sample_discard(), sample_index(), sample_lcomp(), sample_mlacomp(), sample_wtatage()

Examples

```
d <- system.file("extdata", package = "ss3sim")
f_in <- file.path(d, "models", "cod-om", "codOM.dat")
dat_list <- r4ss::SS_readdat(f_in, verbose = FALSE)

## Turn off age comps by specifying fleets=NULL
test <- sample_agecomp(dat_list = dat_list, fleets = NULL)

## Generate with a smaller number of fleet taking samples
ex1 <- sample_agecomp(
  dat_list = dat_list, outfile = NULL,
  fleets = 2, Nsamp = list(c(10, 50)), years = list(c(26, 27))
)
NROW(ex1$agecomp) == 2

## Generate with varying Nsamp by year for first fleet
ex2 <- sample_agecomp(
  dat_list = dat_list, outfile = NULL,
  fleets = c(1, 2),
  Nsamp = list(c(rep(50, 5), rep(100, 5)), 50),
  years = list(seq(26, 44, 2), c(26:100))
)

## Run three  cases showing Multinomial, Dirichlet(1), and over-dispersed
## Dirichlet for different levels of sample sizes
op <- graphics::par(mfrow = c(1, 3))
set.seed(1)
true <- prop.table(dat_list$agecomp[
  dat_list$agecomp$FltSvy == 1 & dat_list$agecomp$Yr == 50, -(1:9)
])
cpars <- c(NA, 1, 4)
for (samplesize in c(30, 100, 1000)) {
  if (samplesize > 30) graphics::par(mar = c(5.1, 1, 4.1, 2.1))
  graphics::plot(dat_list$agebin_vector, true,
    type = "b", ylim = c(0, 1),
    col = 4, lwd = 2, xlab = "Age",
    ylab = ifelse(samplesize == 30, "Proportion", ""),
    main = paste("Sample size =", samplesize)
  )
  if (samplesize == 30) {
    graphics::legend("topright",
      lty = 1, col = 1:4, bty = "n",
      legend = c("Multinomial", "Dirichlet(1)", "Dirichlet(4)", "Truth")
    )
  }
  for (i in seq_along(cpars)) {
    ex <- sample_agecomp(
      dat_list = dat_list, outfile = NULL, fleets = 1,
      Nsamp = list(samplesize), years = list(50), cpar = cpars[i]
    )$agecomp
    lines(dat_list$agebin_vector, prop.table(ex[1, -(1:9)]),
      col = i, type = "b"
    )
  }
}
graphics::par(op)
```

sample_calcomp *Sample conditional age-at-length data*

Description

Sample conditional age-at-length (CAAL) data from expected values of length proportions and expected values of age proportions (conditional on length) from the operating model (OM) and writes the samples to file for use by the estimation model (EM).

Usage

```
sample_calcomp(
  dat_list,
  exp_vals_list,
  outfile = NULL,
  fleets,
  years,
  Nsamp_lengths,
  Nsamp_ages,
  method = "simple_random",
  ESS_lengths = NULL,
  ESS_ages = NULL,
  lcomps_sampled = FALSE,
  ...
)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section` = 1 reads in the input values used to run the model and `section` = 2 reads in the expected values generated given all the input to the OM. `section` = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`exp_vals_list` This is a data list containing all expected values. It should not be modified by previous sampling functions to contain sampled data.

outfile A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.

fleets *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of fleets=NULL leads to zero samples for any fleet.

years *A list the same length as fleets giving the years as numeric vectors. If no fleet collected samples, keep the value to years=NULL.

Nsamp_lengths A numeric list of the same length as fleets. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, specify Nsamp_lengths = NULL. Specifically, for sample_calcomp, Nsamp_lengths denotes the total number of length samples for a given year and fleet across all length bins that can be used to then sample the conditional age at length samples. Nsamp_lengths must be greater than or equal to Nsamp_ages.

Nsamp_ages A numeric list of the same length as fleets. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, specify Nsamp_ages = NULL. Specifically, for sample_calcomp, Nsamp_ages denotes the total number of conditional age at length samples for a given year and fleet across all length bins. Nsamp_ages must be less than Nsamp_lengths.

method The method used to sample ages from the lengths. Options are "simple_random" and "length_stratified". In "simple_random" (the default option), the fish aged are randomly sampled from the age bins, so the number sampled in each age bin is not equal. In "length_stratified", an equal number of fish are aged from each length bin.

ESS_lengths The final effective sample size (ESS) associated with the simulated length data generated for conditional age at length samples. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, NULL, leads to the true (internally calculated) effective sample size being used, which is Nsamp_lengths for the multinomial case. ESS_lengths should be a numeric list of the same length as fleets. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. Note that the dimensions of ESS_lengths must be compatible with the dimensions of Nsample_lengths.

ESS_ages The final effective sample size (ESS) associated with the simulated conditional age at length data. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, NULL, leads to the true (internally calculated) effective sample size being used, which is Nsamp_ages for the multinomial case. ESS_ages should be a numeric list of the same length as fleets. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. Note that the dimensions of ESS_lengths must be compatible with the dimensions of Nsample_ages. The input value will be apportioned among the conditional age at length bins as the `Nsamp_ages` is and therefore can be a fractional value.

`lcomps_sampled` Have marginal length comps already been sampled and are included in `dat_list[["lencomp"]]`? If `FALSE`, expected values are in present in `datlist[["lencomp"]]`.

`...` Any argument you want to be a column in the new data frame of composition data. All extra arguments should be named columns in `data`. Each argument needs to be a list of length `length(fleets)`. Or, you can use a single value that will be repeated for each combination of fleet, year, ... in your data.

Details

There are many steps needed to sample CAAL data because ages are not independent from lengths. The data is located in the `.dat` file alongside age compositions. CAAL have the added complexity of one line per length bin. Thus, each row represents the observed age distribution for a length bin conditioned on the fish lengths that were observed in the length compositions. The age distribution will be truncated for older or younger fish. Often, many rows will be empty because no fish of that length bin were observed. These empty rows are not needed in the .dat file.

The sampling process includes the following steps:

1. Lengths are sampled based on the desired number of lengths, $N$. $N$ is the maximum amount that could be aged.
2. Those lengths are binned to create a length distribution, i.e., numbers of fish in each length bin.
3. Ages are sampled from fish that contributed to the length distribution. Several strategies are possible for sampling ages from those fish
4. age all fish,
5. *take random subset of fish independent of length bin*, or
6. take a fixed number of fish from each length bin.

ss3sim can currently only handle randomly sampling ages from lengthed fish. Future versions could include the last option; please contact the developers if you are interested in helping facilitate this.

Note that the overall total sample size for all CAAL bins is specified by the user for the given fleet and year in `Nsamp_ages`. These sample sizes and the expected values of age proportions (conditional on length) are used to sample for realistic age proportions. If all fish are aged, then no resampling is performed. If no fish are aged for a row of age proportions in conditional age at length data, then that row is discarded. If all fish are not aged, then a new sample size must be drawn. This new sample size must be less than or equal to the number of fish that were sampled for their length. This new sample size is used to draw ages randomly from the expected values. If we consider all rows for a fleet and year (one for each length bin), then the sum of those will be the sample size for the CAAL data. However, if the CAAL sample size is less than the length sample size, We accomplish this in the code by doing sampling without replacement for vectors of length bins equal to the number of fish in them. This ensures realistic sampling. If the option (3) above were implemented, a different strategy would need to be implemented. For instance, if the user wants 10 fish from each length bin but only 5 fish were observed, what to do? A value of NULL for fleets indicates to delete the CAAL data but not the marginal age data.

When Dirichlet sampling is used for length compositions, the number of fish observed will be real-valued and not whole fish. One cannot simply multiply by the length composition sample size to get whole numbers because they are real and rounding or truncating would be unsatisfactory. Currently, the function simply draws a multinomial sample from the length compositions of specified size (`Nsamp`). However, this does not guarantee that fewer fish are aged than lengthed. If you are specifying a small number of fish to age relative to length, then this might be alright. However, *we discourage the use of Dirichlet length samples when using CAAL data* as currently implemented.

Note that this function cannot handle all types of CAAL sampling. This function requires that there be a row of CAAL data for each length data bin (for each year and fleet that sampling is specified to be performed), where Lbin_lo and Lbin_hi are the same value. Note also that this sampling procedure represents simple random sampling for CAAL, where (1) lengths are sampled randomly, (2) fish are lengthed and placed into bins, and (3) a subset of lengthed fish are aged, where a constant proportion from each length bin are selected for aging. This does not represent length stratified sampling where a subset of lengthed fish are aged, and a constant number from each length bin is selected for aging, although these data could also be put into a Stock Synthesis model as CAAL.

Value

A modified `.dat` file if `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Author(s)

Cole Monnahan, Kotaro Ono

See Also

Other sampling functions: `clean_data()`, `sample_agecomp()`, `sample_catch()`, `sample_discard()`, `sample_index()`, `sample_lcomp()`, `sample_mlacomp()`, `sample_wtatage()`

`sample_catch` *Sample the catches with observation error*

Description

This function creates a matrix of sampled catches from the expected available catches for all fleets with catches. The input value used for `catch_se` will be used to resample the catches. There is a bit of a disconnect here because catches are defined by input `F` values not absolute catches. Let D_y be the discard from the operating model for year y. Then the sampled value is calculated as: $D_y * exp(stats::rnorm(1, 0, sds_obs) - sds_obs^2/2)$. The second term adjusts the random samples so that their expected value is D_y, i.e., the log-normal bias correction.

Usage

```
sample_catch(dat_list, outfile = NULL)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

Value

A modified `.dat` file if `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Author(s)

Kelli F. Johnson

See Also

Other sampling functions: `clean_data()`, `sample_agecomp()`, `sample_calcomp()`, `sample_discard()`, `sample_index()`, `sample_lcomp()`, `sample_mlacomp()`, `sample_wtatage()`

`sample_comp` *Sample composition data from expected values*

Description

Apply the multinomial or Dirichlet distribution to sample composition data, creating a data frame that mimics observed composition data.

Usage

```
sample_comp(data, Nsamp, fleets, years, ESS = NULL, cpar = 1, ...)
```

Arguments

`data` A data frame with informational columns followed by columns of compositional data. The informational columns must include columns labeled 'Yr' and 'FltSvy' and end with a column labeled 'Nsamp'. Columns of compositional data should follow 'Nsamp'. Rows of compositional data do not need to sum to one.

`Nsamp` *A numeric list of the same length as `fleets`. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to `Nsamp=NULL`.

`fleets` *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`ESS` The final effective sample size (ESS) associated with the simulated data. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, `NULL`, leads to the true (internally calculated) #' ESS being used, which is `Nsamp` for the multinomial case or given by the formula under `cpar` for the Dirichlet case. At least one value must be provided for each fleet or a vector of year-specific values can be used for any given fleet. The argument accepts a list with entries, either a single integer or a vector of integers, for each fleet.

`cpar` A numeric value or vector the same length as `fleets` controlling the variance of the Dirichlet distribution used for sampling. A value of `1` leads to the same standard deviation as a multinomial of the given `Nsamp`, `2` indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion. `NULL` or `NA` for a given fleet will lead to no dispersion.

`...` Any argument you want to be a column in the new data frame of composition data. All extra arguments should be named columns in `data`. Each argument needs to be a list of length `length(fleets)`. Or, you can use a single value that will be repeated for each combination of fleet, year, ... in your data.

Details

Sample size, i.e., 'Nsamp', is used as a measure of precision, where higher sample sizes lead to simulated samples that more accurately represent the truth provided in `data`.

Value

A data frame of observed composition data.

Author(s)

Kelli F. Johnson

`sample_discard` *Sample the discard with observation error*

Description

This function creates an index of discards sampled from the expected available discards for specified fleets in specified years. Let D_y be the discard from the operating model for year y. Then the sampled value is calculated as: $D_y * exp(stats :: rnorm(1, 0, sds_obs) - sds_obs^2/2)$. The second term adjusts the random samples so that their expected value is D_y, i.e., the log-normal bias correction.

Usage

```
sample_discard(
  dat_list,
  outfile = NULL,
  fleets,
  years,
  sds_obs,
  seas = list(1)
)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

`fleets` *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`sds_obs` A list the same length as `fleets`. The list should contain either single values or numeric vectors of the same length as the number of years which represent the standard deviation of the observation error. Single values are repeated for all years.

`seas` A list the same length as `fleets`. The list should contain either single numeric values or numeric vectors, where the length of each vector matches the length of each vector present in `years`. Single values are repeated for all years. This parameter operates the same as `sds_obs` but it specifies the season rather than the standard deviation of the observations.

Value

A modified `.dat` file if `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Author(s)

Kelli F. Johnson

See Also

Other sampling functions: `clean_data()`, `sample_agecomp()`, `sample_calcomp()`, `sample_catch()`, `sample_index()`, `sample_lcomp()`, `sample_mlacomp()`, `sample_wtatage()`

`sample_dm` *Sample with a Dirichlet-Multinomial distribution*

Description

Sample with a Dirichlet-Multinomial distribution

Usage

```
sample_dm(data, n, par)
```

Arguments

`data`	A data frame with one row.
`n`	The desired sample size.
`par`	The cpar value to define overdispersion.

Value

A data frame with one row because right now the input data should only be a single row of data.

sample_index *Sample the CPUE/index data with observation error*

Description

Create new catch-per-unit-effort (CPUE)/indices of abundance that are based on the numbers in a data file. Typically the data file will be filled with expected values rather than observed data but it does not have to be. Sampling can only occur on fleets, years, and seasons that have current observations. If rows of information are not sampled from, then they are removed. So, you can take away rows of data but you cannot add them with this function.

Usage

```
sample_index(
  dat_list,
  outfile = lifecycle::deprecated(),
  fleets,
  years,
  sds_obs = list(0.01),
  sds_out,
  seas = list(1)
)
```

Arguments

`dat_list` A Stock Synthesis data list returned from `r4ss::SS_readdat()`. Typically, this will be read from a file that contains expected values rather than input values but any Stock Synthesis data file is fine.

`outfile` A deprecated argument.

`fleets` An integer vector specifying which fleets to sample from. The order of the fleets matters here because you must retain the ordering for all of the remaining input arguments. For example, both `fleets = c(1, 2)` and `fleets = c(2, 1)` will work but `years` will expect the years you want sampled for fleet 2 to be in the second position in the list in the former and the first position in the latter case. So, if you change the order of your input, you will also have to modify all of the remaining arguments. An entry of `fleets = NULL` will lead to no CPUE samples in the returned object.

`years` A list the same length as `fleets` specifying the years you want samples from. There must be an integer vector in the list for every fleet specified in `fleets`. The function assumes that the information for the first fleet specified in `fleets` will be the first object in the list and so on so order matters here.

`sds_obs`, `sds_out`, `seas`
A list the same length as `fleets` specifying the standard deviation of the observation error used for the sampling; the standard deviation of the observation error you would like listed in the returned output, which might not always equal what was actually used for sampling; and the seasons you want to sample from.

Each list element should contain a single numeric value or a vector, where vectors need to match the structure of `years` for the relevant fleet. If single values are passed, then, internally, they will be repeated for each year. If you want to repeat a single value for every year and fleet combination, then just pass it as a list with one entry, e.g., `seas = list(1)` will sample from season one for all fleets and years — this is the default for season. The default for `sds_obs` is 0.01 and if `sds_out` is missing, then `sds_obs` will be used for the output as well as the input.

Details

Limitations to the functionality of this function are as follows:

- you can only generate observations from rows of data that are present, e.g., you cannot make a new observation for a year that is not present in the passed data file;
- no warning will be given if some of the desired year, seas, fleet combinations are available but not all, instead just the combinations that are available will be returned in the data list object; and
- sampling uses a log-normal distribution when the log-normal distribution is specified in `CPUEinfo[["Errtype"]]` and a normal distribution for all other error types, see below for details on the log-normal sampling.

Samples are generated using the following equation when the log-normal distribution is specified:

$$B_y * exp(stats :: rnorm(1, 0, sds_obs) - sds_obs^2/2)$$

, where B_y is the expected biomass in year y and sds_obs is the standard deviation of the normally distributed biomass or the standard error of the $log_e(B_y)$. For the error term, this is the same parameterization that is used in Stock Synthesis. More details can be found in the section on indices in the Stock Synthesis manual The second term in the equation adjusts the random samples so their expected value is B_y, i.e., the log-normal bias correction.

If you only know the coefficient of variation (CV), then the input error can be approximated using $\sqrt{log_e(1 + CV^2)}$. Where, CV is assumed to be constant with mean changes in biomass. The log-normal distribution can be approximated by a proportional distribution or normal distribution only when the variance is low, i.e., $CV < 0.50$ or log standard deviation of 0.22.

Value

A Stock Synthesis data file list object is returned. The object will be a modified version of `dat_list`.

Author(s)

Cole Monnahan, Kotaro Ono

See Also

Other sampling functions: `clean_data()`, `sample_agecomp()`, `sample_calcomp()`, `sample_catch()`, `sample_discard()`, `sample_lcomp()`, `sample_mlacomp()`, `sample_wtatage()`

Examples

```
# Add a list from [r4ss::SS_readdat()] to your workspace, this is example
# data that is saved in the ss3sim package.
# Index data are saved in `dat_list[["CPUE"]]`
dat_list <- r4ss::SS_readdat(
  file = file.path(
    system.file("extdata", "example-om", package = "ss3sim"),
    "ss3_expected_values.dat"
  ),
  verbose = FALSE
)


# Sample from each available year from fleet 2 with an increasing trend in
# the observation error, i.e., the most recent year has the highest
# likelihood to be the furthest from the input data
ex1 <- sample_index(
  dat_list,
  outfile = NULL,
  fleets = 2,
  seas = list(
    dat_list[["CPUE"]][dat_list[["CPUE"]][, "index"] == 2, "seas"]
  ),
  years = list(dat_list[["CPUE"]][["year"]]),
  sds_obs = list(
    seq(0.001, 0.1, length.out = length(dat_list[["CPUE"]][["year"]]))
  )
)


## Not run:
# Sample from less years, note that sampling from more years than what is
# present in the data will not work
ex2 <- sample_index(dat_list,
  outfile = NULL,
  fleets = 2,
  seas = list(unique(
    dat_list[["CPUE"]][dat_list[["CPUE"]][, "index"] == 2, "seas"]
  )),
  years = list(dat_list[["CPUE"]][["year"]][-c(1:2)]),
  sds_obs = list(0.001)
)

# sd in the returned file can be different than what is used to sample, this
# is helpful when you want to test what would happen if the estimation method
# was improperly specified
ex3 <- sample_index(
  dat_list = dat_list,
  fleets = 2,
  seas = list(unique(
    dat_list[["CPUE"]][dat_list[["CPUE"]][, "index"] == 2, "seas"]
  )),
  years = list(dat_list[["CPUE"]][["year"]]),
  sds_obs = list(0.01),
  sds_out = list(0.20)
)
ex3[["CPUE"]][["se_log"]]

## End(Not run)
# Sample from two fleets after adding fake CPUE data for fleet 1
dat_list2 <- dat_list
dat_list2[["CPUE"]] <- rbind(
  dat_list[["CPUE"]],
  dat_list[["CPUE"]] |>
    dplyr::mutate(index = 1, seas = 1)
)
dat_list2[["N_cpue"]] <- NROW(dat_list2[["CPUE"]])
ex4 <- sample_index(
  dat_list = dat_list2,
  fleets = 1:2,
  seas = list(1, 7),
  # Subset two years from each fleet
  years = list(c(76, 78), c(80, 82)),
  # Use the same sd values for both fleets
  sds_obs = list(0.01),
  sds_out = list(0.20)
)
```

sample_lcomp *Sample length compositions from a Stock Synthesis data file*

Description

Extract length-composition data from a `.ss_new` data file and sample the data. It is assumed that the composition data will be expected values as written by Stock Synthesis in the second section of the data file, but one can also sample input data. The resulting length-composition data are assumed to represent observed length composition and will overwrite the length data in `dat_list`, which is returned invisibly. The data file can also be written to the disk, if a file path is provided to `outfile`, and used as simulated data by an estimation model.

Usage

```
sample_lcomp(
  dat_list,
  outfile = NULL,
  fleets,
  Nsamp,
  years,
  cpar = 1,
  ESS = NULL,
  ...
)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

`fleets` *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.

`Nsamp` *A numeric list of the same length as `fleets`. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to `Nsamp=NULL`.

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`cpar` A numeric value or vector the same length as `fleets` controlling the variance of the Dirichlet distribution used for sampling. A value of `1` leads to the same standard deviation as a multinomial of the given `Nsamp`, `2` indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion. `NULL` or `NA` for a given fleet will lead to no dispersion.

`ESS` The final effective sample size (ESS) associated with the simulated data. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, `NULL`, leads to the true (internally calculated) #' ESS being used, which is `Nsamp` for the multinomial case or given by the formula under `cpar` for the Dirichlet case. At least one value must be provided for each fleet or a vector of year-specific values can be used for any given fleet. The argument accepts a list with entries, either a single integer or a vector of integers, for each fleet.

`...` Any argument you want to be a column in the new data frame of composition data. All extra arguments should be named columns in `data`. Each argument needs to be a list of length `length(fleets)`. Or, you can use a single value that will be repeated for each combination of fleet, year, ... in your data.

Value

A modified `.dat` file if `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Author(s)

Cole Monnahan and Kotaro Ono

See Also

sample_agecomp() for more examples.

Other sampling functions: clean_data(), sample_agecomp(), sample_calcomp(), sample_catch(), sample_discard(), sample_index(), sample_mlacomp(), sample_wtatage()

Examples

```
dat_list <- r4ss::SS_readdat(
  verbose = FALSE,
  file = system.file(file.path("extdata", "models", "cod-om", "codOM.dat"),
    package = "ss3sim"
  )
)
## Generate with constant sample size across years
ex1 <- sample_lcomp(
  dat_list = dat_list, outfile = NULL,
  fleets = 1:2, Nsamp = list(100, 50),
  years = list(seq(26, 100, by = 2), 80:100)
)
```

---

sample_lognormal *Sample observations using log-normal error corrected for bias*

Description

Sample a standard normal in log-space and apply the error to observations.

Usage

```
sample_lognormal(obs, sd)
```

Arguments

obs	A vector of observed values you wish to sample with log-normal error.
sd	A vector of standard deviations to use in stats::rnorm().

Details

Newly sampled values are calculated $obs * exp(stats :: rnorm(1, 0, sd) - sd^2/2)$. The second term adjusts the random samples so that their expected value is obs, i.e., the log-normal bias correction.

Author(s)

Cole Monnahan

sample_mlacomp — **BETA VERSION** *Sample mean length (size-)-at-age data and write to file for use by the EM*

Description

BETA VERSION Sample mean length (size-)-at-age data and write to file for use by the EM

Usage

```
sample_mlacomp(
  dat_list,
  outfile,
  ctl_file_in,
  fleets = 1,
  Nsamp,
  years,
  mean_outfile = NULL,
  verbose = TRUE
)
```

Arguments

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section = 1` reads in the input values used to run the model and `section = 2` reads in the expected values generated given all the input to the OM. `section = 3` is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

`ctl_file_in` A path to the control file, output from an OM, containing the OM parameters for growth. These values are used to determine the uncertainty about size for fish sampled in each age bin.

`fleets` *A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.

`Nsamp` *A numeric list of the same length as `fleets`. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to `Nsamp=NULL`.

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`mean_outfile`	A path to write length and age data for external estimation of parametric growth. If `NULL` no file will be written. Also, if "remove" is included in the filename, the mean length at age data will be removed from the `.dat` file and not be available to the EM.
`verbose`	Logical value whether or not diagnostic information from **r4ss** functions should be printed to the screen. Default is FALSE.

Details

This function is in beta and untested. Use with caution. Take a `data_expval.ss` file, read in by **r4ss** function `r4ss::SS_readdat()` containing observed values, and sample from the observed ages to get realistic proportions for the number of fish in each age bin, then use the mean size-at-age and CV for growth to generate random samples of size, which are then averaged to get mean length-at-age values. These values are then written to file for the EM.

Value

A modified `.dat` file if `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Author(s)

Cole Monnahan, Kelli F. Johnson

See Also

Other sampling functions: `clean_data()`, `sample_agecomp()`, `sample_calcomp()`, `sample_catch()`, `sample_discard()`, `sample_index()`, `sample_lcomp()`, `sample_wtatage()`

`sample_mn` *Sample with a multinomial distribution*

Description

Sample with a multinomial distribution

Usage

```
sample_mn(data, n)
```

Arguments

`data`	A data frame with one row.
`n`	The desired sample size.

Value

A data frame with one row because right now the input data should only be a single row of data.

sample_wtatage *Sample empirical weight-at-age data*

Description

In Stock Synthesis, empirical weight-at-age data can be used to read empirical body weight for the population from each fleet. This data removes the use of growth parameters from the EM because weights are assigned to each age internally rather than from the growth parameters, from which spawning biomass/fecundity can be determined. These values are not data in the sense they have a likelihood but are generated from samples. Sampling empirical weight-at-age data from the expected values takes many steps.

Usage

```
sample_wtatage(
  wta_file_in,
  outfile,
  dat_list,
  ctl_file_in,
  years,
  fill_fnc = fill_across,
  fleets,
  cv_wtatage = NULL
)
```

Arguments

`wta_file_in` The file to read weight-at-age from. Specifically to get the age-0 weight-at-age. This is typically `wtatage.ss_new`.

`outfile` A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of `NULL`, which leads to increased speed because writing the file takes time and computing resources.

`dat_list` A Stock Synthesis data list object as read in from `SS_readdat`. Be sure to correctly specify which section of the data file you want to work with when reading it in using the `section` argument. Where, `section` = 1 reads in the input values used to run the model and `section` = 2 reads in the expected values generated given all the input to the OM. `section` = 3 is not used within ss3sim, but this section provides bootstrapped data sets that have been sampled internally within SS.

`ctl_file_in` A path to the control file, output from an OM, containing the OM parameters for growth and weight/length relationship. These values are used to determine the uncertainty about weight for fish sampled in each age bin. Commonly `control.ss_new`

`years` *A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.

`fill_fnc`	A function to fill in missing values (ages and years). The resulting weight-at-age file will have values for all years and ages.One function is `fill_across()`.
`fleets`	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of `fleets=NULL` leads to zero samples for any fleet.
`cv_wtatage`	A user specified coefficient of variation (CV) for growth. Default is `NULL`.

Details

The steps for sampling empirical weight-at-age are as follows:

- Sample from the expected ages to get realistic proportions for the number of fish in each age bin.
- Use the mean size-at-age and coefficient of variation for growth to generate random samples of size, which are then converted to weight and averaged to get mean weight-at-age values.
- Fill in missing ages and years.
- Write the information to the appropriate files.
- Turn on weight-at-age data in Stock Synthesis by setting the maturity option to 5.

Value

A modified `.wtatage.ss` file if `!is.null(outfile)`. A list object containing the modified `.wtatage.ss` file is returned invisibly.

Author(s)

Cole Monnahan, Allan Hicks, Peter Kuriyama

See Also

- `fill_across()`
- `ss3sim_base()`

Other sampling functions: `clean_data()`, `sample_agecomp()`, `sample_calcomp()`, `sample_catch()`, `sample_discard()`, `sample_index()`, `sample_lcomp()`, `sample_mlacomp()`

`scalar_dat` *Example scalar data from the Introduction vignette*

Description

An R object read in using `utils::read.csv("ss3sim_scalar.csv")` after processing the results from the Introduction vignette using `get_results_all()`. The data set is available so users do not have to wait for the scenarios to run.

Examples

```
data("scalar_dat", package = "ss3sim")
```

`setup_bins` *Set up bin structure for composition data*

Description

Set up the bin structure needed for composition data.

Usage

```
setup_bins(bins, nsex = 1, leader = c("l", "a"))
```

Arguments

`bins`	A vector of integer values, either lengths or ages. Do not repeat them if you are using a two-sex model, the function will do that for you.
`nsex`	A single integer of one or two specifying the number of sexes in the model.
`leader`	Most users will not need to change the leader character from the default unless you are working with age data, then just use \code"a" rather than the default of `"l"`. If you have a two-sex model, i.e., `nsex = 2`, then the function will take care of the naming for you, which is forced to `"f"` and `"m"`. You can change them afterwards if you want.

See Also

This is a helper function used to create the bins before sampling takes place, see `ss3sim_base()`.

Examples

```
ex <- setup_bins(bins = 1:10, nsex = 2, leader = "a")
test <- length(ex) == 20 & all(grep("m", ex) == 11:20)

ex <- setup_bins(bins = 1:5, nsex = 1)
test <- ex[4] == "l4"
```

`setup_parallel` *Setup parallel processing*

Description

Setup parallel processing

Usage

```
setup_parallel()
```

`setup_scenarios` *Get scenario information from a data frame of specifications*

Description

Get scenario information from a data frame of specifications

Usage

```
setup_scenarios(df = "default", returntype = c("list", "dataframe"))
```

Arguments

`df`	A data frame with scenarios in the rows and information for function arguments in the columns. See setup_scenarios_defaults for how to set up the data frame. This data frame is used by default if you do not supply anything to `df`.
`returntype`	The class of object that you want to return. ss3sim was a big fan of lists of lists until the `tidyverse` packages were included. Now, data frames of list columns are preferred. Eventually, `list` will be downgraded from the default and data frames will be the only option as a return.

Value

Either a long data frame or a list is returned. See the input argument `returntype` for more information.

Author(s)

Kelli F. Johnson

Examples

```
defaultscenarios <- setup_scenarios()
```

`setup_scenarios_defaults` *Set up a generic scenario*

Description

Create a data frame of scenario inputs for a generic simulation that will run within ss3sim. Users can add more arguments, but the scenario will run without changing the returned value.

Usage

```
setup_scenarios_defaults(nscenarios = 1)
```

Arguments

`nscenarios`	The number of rows you want returned in the data frame. This argument removes the need for users to call `base::rbind()` repeatedly on the output when you want to have more than one scenario. All rows will be identical with the default settings. The default is a single row.

Value

A data frame with the minimal information needed to run a scenario. The number of rows of the data frame depends on `nscenarios`.

Author(s)

Kelli F. Johnson

`setup_scenarios_fleet` *Set up fleet-specific information*

Description

Sometimes, users will want to pass a single input instead of fleet-specific information to make things easier to keep track of for the user. `get_fleet` copies this single object over to all fleets for a given sampling type.

Usage

```
setup_scenarios_fleet(data)
```

Arguments

`data`	A data frame of scenario information that was passed to `setup_scenarios()` and as subsequently been passed to this function as a long data frame rather than a wide data frame.

Details

In the data frame that stores scenario-specific information by row, columns are fleet-specific with the fleet denoted after the last full stop. If this terminal full stop followed by a numerical value is not supplied, then the value will be copied for all fleets. For example, `sa.Nsamp.1` specifies the sample size for age-composition data for fleet number one. Whereas, `sa.Nsamp` specifies the input sample-size for all fleets.

A todo list for future features is as follows:

- remove fleets that have NA
- allow for arguments rather than hardwiring arg and fleet
- see if sa.Nsamp and sa.Nsamp.1 can be in the same data frame and just fill in the value for fleets that aren't specified; would need to fill up and down I think within a group to make it work.
- accomodate -999 in sample function cpar arguments
- create add_args to fill in missing arguments across fleets
- implement add_args before expand fleet such that the new arg would be expanded for all fleets but I only have to specify the default one time
- fix `.data[[""]]` to pass CRAN x <- enquo(x) y <- enquo(y) ggplot(data) + geom_point(aes(!!x, !!y))

Value

An augmented data frame is returned in the same form as the input data. The new rows correspond to parsing input arguments out across all fleets that are sampled when a single input value is provided.

Author(s)

Kelli F. Johnson

`setup_scenarios_lookup` *Create a named vector to look up full names for types of arguments*

Description

Create a named vector to look up full names for types of arguments

Usage

```
setup_scenarios_lookup()
```

setup_scenarios_name *Create a name for an unnamed scenario*

Description

Create a name for an unnamed scenario based on Sys.time.

Usage

```
setup_scenarios_name(check = FALSE)
```

Arguments

check A logical that enables checking for a unique name. If check = TRUE then the function enters a loop and will generate a names until it finds one that doesn't already exist. This could be helpful when running scenarios in parallel.

Value

A single character value is returned. The object starts with the letter s and is followed by Sys.time Where, the date/time portion is %m%d%H%M%S, better known as a two-digit month, e.g., 01; a two-digit number for the day of the month; and finally a two-digit hour, then minute, then second.

ss3sim_base *High-level wrapper to run a simulation*

Description

A wrapper function that

- calls r4ss::run() to run the operating model,
- samples the output to create fishery and survey data, and
- calls r4ss::run() to run the estimation model. This function is the main workhorse of ss3sim and is typically not called by the user but called from run_ss3sim().

Usage

```
ss3sim_base(
  iterations,
  scenarios,
  f_params,
  index_params,
  discard_params = NULL,
  lcomp_params = NULL,
  agecomp_params = NULL,
  calcomp_params = NULL,
  wtatage_params = NULL,
  mlacomp_params = NULL,
  em_binning_params = NULL,
  estim_params = NULL,
  tv_params = NULL,
  operat_params = NULL,
  om_dir,
  em_dir,
  retro_params = NULL,
  data_params = NULL,
  weight_comps_params = NULL,
  user_recdevs = NULL,
  user_recdevs_warn = TRUE,
  bias_adjust = FALSE,
  sleep = 0,
  seed = 21,
  extras = " "
)
```

Arguments

`iterations` Which iterations to run. A numeric vector.

`scenarios` A name to use as the folder name for the unique combination of parameters for the OM and EM.

`f_params` A named list containing arguments for `change_f()`. A mandatory input.

`index_params` A named list containing arguments for `sample_index()`. A mandatory input.

`discard_params` A named list containing arguments for `sample_discard()`.

`lcomp_params` A named list containing arguments for `sample_lcomp()`. A mandatory input.

`agecomp_params` A named list containing arguments for `sample_agecomp()`. A mandatory input.

`calcomp_params` A named list containing arguments for `sample_calcomp()`, for conditional age-at-length data.

`wtatage_params` A named list containing arguments for `sample_wtatage()`, for empirical weight-at-age data.

`mlacomp_params` A named list containing arguments for `sample_mlacomp()`, for mean length-at-age data.

`em_binning_params` A named list containing arguments for `change_em_binning()`.

`estim_params` A named list containing arguments for `change_e()`.

`tv_params` A named list containing arguments for `change_tv()` (time-varying).

`operat_params` A named list containing arguments for `change_o()`.

`om_dir` The directory with the operating model you want to copy and use for the specified simulations.

em_dir	The directory with the estimation model you want to copy and use for the specified simulations. If NA, then no estimation method is included and ss3sim just generates data.
retro_params	A named list containing the arguments for change_retro().
data_params	A named list containing arguments for changing data.
weight_comps_params	A named list containing arguments for r4ss::tune_comps().
user_recdevs	An optional matrix of recruitment deviations to replace the recruitment deviations built into the package. The columns represent run iterations and the rows represent years. user_recdevs can be a matrix of 0s for deterministic model checking. For traditional stochastic simulations these would be independent and normally distributed deviations with a standard deviation equal to the desired sigma R. Note that these recruitment deviations will be used verbatim (after exponentiation). user_recdevs will *not* be multiplied by sigma R and they will *not* be log-normal bias corrected. If user_recdevs are specified as anything besides NULL the package will issue a warning about this. Biased recruitment deviations can lead to biased model results.
user_recdevs_warn	A logical argument allowing users to turn the warning regarding biased recruitment deviations off when user_recdevs are specified.
bias_adjust	A logical argument specifying bias adjustment is conducted. Bias adjustment helps assure that the estimated recruitment deviations, which are assumed to be log-normally distributed, are mean unbiased leading to mean-unbiased estimates of biomass Methot and Taylor, 2011. Bias adjustment should always be performed when using maximum likelihood estimation when running simulations for publication or management. The argument allows users to turn bias adjustment off because it involves running the EM multiple times with the hessian and is not needed when initially exploring your simulation structure.
sleep	A time interval (in seconds) to pause on each iteration. Useful if you want to reduce average CPU time – perhaps because you're working on a shared server.
seed	The seed value to pass to get_recdevs() when generating recruitment deviations. The generated recruitment deviations depend on the iteration value, but also on the value of seed. A given combination of iteration, number of years, and seed value will result in the same recruitment deviations.
extras	A character string that will be passed to the extras argument of r4ss::run(). The default is " " which results in the hessian being inverted if the model has positive phases, i.e., the EM. Pass "-nohess " if you do not want to estimate the hessian or "-stopph 3 -nohess" if you want to stop the model in phase 3 and you do not want to invert the hessian. The key is that the entry must be one string with spaces between the arguments that will be passed to ADMB.

Details

This function is written to be flexible. You can specify the fishing mortality, survey catch-per-unit-effort settings, length-composition data settings, etc. in the function call as list objects (see the example below). For a generic higher-level function, see run_ss3sim().

Value

The output will appear in whatever your current R working directory is. There will be folders named after your scenarios with one folder per iteration. Each iteration folder with include an operating model and an estimation method. Your directory will look like the following:

- scen-cod/1/om
- scen-cod/1/em
- scen-cod/2/om
- ...

If em_dir = NA, then the contents of the em directories will be minimal because they will only contain the simulated data and not any fits to those data.

Author(s)

Sean Anderson with contributions from many others as listed in the DESCRIPTION file.

See Also

run_ss3sim()

Examples

```
## Not run:
# Create a temporary folder for the output and set the working directory:
# Create a temporary folder for the output and set the working directory:
temp_path <- file.path(tempdir(), "ss3sim-base-example")
dir.create(temp_path, showWarnings = FALSE)
wd <- getwd()
setwd(temp_path)
on.exit(setwd(wd), add = TRUE)

# Find the data in the ss3sim package:
d <- system.file("extdata", package = "ss3sim")
om_dir <- file.path(d, "models", "cod-om")
em_dir <- file.path(d, "models", "cod-em")

# Or, create the argument lists directly in R:

F0 <- list(
  years = 1:100,
  fleets = 1,
  fvals = c(rep(0, 25), rep(0.114, 75))
)

index1 <- list(
  fleets = 2, years = list(seq(62, 100, by = 2)),
  sds_obs = list(0.1)
)

lcomp1 <- list(
  fleets = c(1, 2), Nsamp = list(50, 100),
  years = list(26:100, seq(62, 100, by = 2))
)

agecomp1 <- list(
  fleets = c(1, 2), Nsamp = list(50, 100),
  years = list(26:100, seq(62, 100, by = 2))
)

E0 <- list(
  par_name = c("LnQ_base_Fishery", "NatM_uniform_Fem_GP_1"),
  par_int = c(NA, NA), par_phase = c(-5, -1), forecast_num = 0
)

ss3sim_base(
  iterations = 1,
  scenarios = "D1-E0-F0-cod", # name as desired
  f_params = F0,
  index_params = index1,
  lcomp_params = lcomp1,
  agecomp_params = agecomp1,
  estim_params = E0,
  om_dir = om_dir,
  em_dir = em_dir
)
replist <- r4ss::SS_output(file.path("D1-E0-F0-cod", 1, "em"),
  verbose = FALSE, printstats = FALSE, covar = FALSE
)
testthat::expect_equivalent(replist[["cpue"]][, "Yr"], index1[["years"]][[1]])

test <- replist
unlink("D1-E0-F0-cod", recursive = TRUE) # clean up

# Run without an EM, where {ss3sim} is a data-generating tool
ss3sim_base(
  iterations = 1,
  scenarios = "noEM",
  f_params = F0,
  index_params = index1,
  lcomp_params = lcomp1,
  agecomp_params = agecomp1,
  estim_params = E0,
  om_dir = om_dir,
  em_dir = NA
)

## End(Not run)
```

standardize_sampling_args *Check and standardize list components of sampling functions*

Description

Check and standardize list components of sampling functions

Usage

```
standardize_sampling_args(
  fleets,
  years,
  other_input,
  return_val = "other_input",
  other_input_name = "other_input"
)
```

Arguments

`fleets` Fleet numbers as a vector.

`years` Number of years as a list. The number of list components should be one or the same length as fleets. Within the list components should be a vector of years to correspond with each fleet.

`other_input` Some other input to interpret. The number of list components should be one or the same length as fleets. Within the list components should be a vector of length 1 the same length as the vectors within years.

`return_val` If `other_input`, return the manipulated other_input value; if `years`, return the manipulated year input. If "both" return both as list components.

`other_input_name` Only necessary if `both` is used as the return value.

ts_dat *Example time-series data from the Introduction vignette*

Description

An R object read in using `read.csv("ss3sim_ts.csv")` after processing the results from the Introduction vignette using `get_results_all()`. The data set is available so users do not have to wait for the scenarios to run.

Examples

```
data("ts_dat", package = "ss3sim")
```

verify_input *Verify and standardize Stock Synthesis input files*

Description

Verify the contents of operating model (OM) and estimation model (EM) folders, i.e., check that the necessary Stock Synthesis input files are available. If the contents are correct, the `.ctl` and `.dat` files are renamed to standardized names and the `starter.ss` file is updated to reflect these names. If the contents are incorrect, then a warning is issued and the simulation is aborted.

Usage

```
verify_input(model_dir, type = c("om", "em"))
```

Arguments

`model_dir`	Directory name for model. This folder should contain the `.ctl`, `.dat`, files etc.
`type`	One of "om" or "em" for operating or estimating model.

Value

Nothing is returned from this function. Instead, file are changed and saved to the disk.

Author(s)

Curry James Cunningham; modified by Sean Anderson

Examples

```
# Create a temporary folder for the output:
temp_path <- file.path(tempdir(), "ss3sim-verify-example")
dir.create(temp_path, showWarnings = FALSE)

d <- system.file("extdata", "models", package = "ss3sim")

om <- file.path(d, "cod-om")
em <- file.path(d, "cod-em")

file.copy(om, temp_path, recursive = TRUE)
file.copy(em, temp_path, recursive = TRUE)

# Verify the correct files exist and change file names:
verify_input(model_dir = file.path(temp_path, "cod-om"), type = "om")
verify_input(model_dir = file.path(temp_path, "cod-em"), type = "em")
unlink(temp_path, recursive = TRUE)
```

verify_plot_arguments *Helper function for ensuring correct input for the plotting functions*

Description

Used internally by the plotting functions to check that the arguments are structured appropriately.

Usage

```
verify_plot_arguments(
  data,
  x,
  y,
  horiz,
  horiz2,
  vert,
  vert2,
  color,
  relative.error,
  axes.free,
  print
)
```

Arguments

data	A valid data frame containing scalar or timeseries values from a **ss3sim** simulation. That data are generated from get_results_all.
x	A character string denoting which column to use as the x variable. For time-series data, setting x = "year" leads to a time-series plot.
y	A character string denoting which column to use as the y variable. Must be a numeric column.
horiz, horiz2	A character string denoting which column to use as the first (horiz) and second (horiz2) level of faceting in the horizontal direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the horizontal space.
vert, vert2	A character string denoting which column to use as the first (vert) and second (vert2) level of faceting in the vertical direction. E.g., "M" or "species". A value of NULL (default) indicates no faceting in the vertical space.
color	A character string denoting which column to use to map color. Not valid for boxplot functions. Useful for looking at EM performance criteria against other dimensions of the EM or OM. See example below for how to merge in a metric from a scalar dataset to a ts dataset.
relative.error	Boolean for whether the y-axis scale should be interpreted as relative error. If TRUE, ylim is set to c(-1, 1), the y-axis label is changed automatically, and a black, dashed line at y=0 is added. The argument can also accept a color entry if you wish the line to be something other than black. E.g., "red" will add a red dashed line at zero as well as fix the y-axis limits.

`axes.free` Boolean for whether the y-axis scales should be free in `facet_grid`.

`print` A logical for whether the plot is printed or not.

Details

The **ss3sim** plotting functions are simply wrappers for **ggplot2** code, specific to the output from **ss3sim** `get_results_all()` objects. They are designed to quickly explore simulation output, rather than produce publication-level figures. The functions use arguments passed as characters that refer to columns of `data`. Scalar plots requires a value for `x`; while, for time-series plots, `x = "year"` will be necessary.

Note that there are some subtle differences between the functions. Boxplots cannot have a color mapped to them like points or lines, and thus, `color` is not a valid argument. The time-series point and line plots are grouped internally by 'ID', which is a combination of scenario and iteration and will be automatically added to the data set if not already present.

Value

Nothing is returned; an informative error is thrown if an argument is invalid.

Output

These functions print the `ggplot` object, but also return it invisibly for saving or printing again later. For example, you could save the `ggplot` object and add a custom theme or change an axis label before printing it.

Author(s)

Cole Monnahan

Index