Package 'ss3sim'

April 19, 2017

Type Package

Title Fisheries Stock Assessment Simulation Testing with Stock Synthesis

Version 0.9.5

Date 2017-04-18

Description Develops a framework for fisheries stock assessment simulation testing with Stock Synthesis 3 (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

LazyData true

Suggests knitr, doParallel, reshape2, rmarkdown

VignetteBuilder knitr

Depends R (>= 3.3.3)

Imports foreach, plyr, r4ss, gtools, lubridate, ggplot2, bbmle, dplyr, magrittr, grDevices, graphics, stats, utils

RoxygenNote 6.0.1

NeedsCompilation no

Author Sean Anderson [aut, cre],
Cole Monnahan [aut],
Kelli Johnson [aut],
Kotaro Ono [aut],
Juan Valero [aut],
Curry Cunningham [aut],
Allan Hicks [aut],
Felipe Hurtado-Ferro [aut],
Peter Kuriyama [aut],
Roberto Licandeo [aut],
Carey McGilliard [aut],
Melissa Muradian [ctb],

Merrill Rudd [aut],
Christine Stawitz [aut],
Cody Szuwalski [aut],
Ian Taylor [aut],
Katyana Vert-pre [aut],
Athol Whitten [aut]

Maintainer Sean Anderson <sean@seananderson.ca>

Repository CRAN

Date/Publication 2017-04-19 03:55:46 UTC

R topics documented:

add_nulls *Add NULL values to non-existent list elements*

Description

Add NULL values to non-existent list elements

Usage

```
add_nulls(param_list, desired_params)
```

Arguments

param_list	A list in which the names correspond to parameter names and the values correspond to the values to be passed.
desired_params	A character vector of desired list elements.

Value

A list with the desired elements as described by the desired_params argument. Any values that were missing in param_list will be returned with values of NULL.

Author(s)

Sean C. Anderson

bias_ss3 *Perform a single bias adjustment run*

Description

This function is run within run_bias_ss3 and for a single run it:

- uses **r4ss** function SS_output to read in the output from a single bias adjustment run
- uses **r4ss** function SS_fitbiasramp to calculate the bias adjustment parameters for that run
- Writes the bias adjustment parameters to the file AdjustBias.DAT within the dir folder, overwriting the file if iter = 1 (the first run) and appending the file otherwise

Usage

```
bias_ss3(iter, dir)
```

Arguments

iter	Replicate number. Used to identify this iteration if there are multiple adjustment runs.
dir	Passes dir from the function run_bias_ss3 to bias_ss3. In run_bias_ss3 this is run within an sapply function for each of the bias adjustment runs.

Value

A plain text file containing the bias adjustment variables is created at dir/AdjustBias.DAT. A PDF figure is created in dir/biasramp-N.pdf, where N represents the iteration number.

Author(s)

Carey McGilliard

References

Methot, R. D. and Taylor, I. G. (2011). Adjusting for bias due to variability of estimated recruitments in fishery assessment models. Can. J. Fish. Aquat. Sci., 68(10):1744-1760.

See Also

run_bias_ss3, run_ss3sim, ss3sim_base

---

calculate_data_units *Given sampling arguments, calculate super set of fleets, years, and data types.*

Description

Given sampling arguments, calculate super set of fleets, years, and data types.

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

An invisible list of 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. See the vignette for further information.

See further examples in `change_data`.

Author(s)

Cole Monnahan

See Also

clean_data, change_data

Examples

```
## Should throw error since nothing passed
# calculate_data_units()
## Only one fleet
calculate_data_units(lcomp_params=list(fleets=1, years=c(3,4,6)))
## Add new fleet
calculate_data_units(lcomp_params=list(fleets=1, years=c(3,4,6)),
                     agecomp_params=list(fleets=2, years=5))
## If CAL data called, need other types even if not specified
calculate_data_units(calcomp_params=list(fleets=1, years=c(3,4,6)))
```

`calculate_re` *Calculate relative error*

Description

Takes a scalar or time series data frame from an **ss3sim** run and calculates relative error (em - om) / em.

Usage

```
calculate_re(dat, add = TRUE)
```

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 function. Specifically, the data frame needs to have columns with _em and _om as names.
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.

Author(s)

Sean Anderson and Cole Monnahan

See Also

get_results_all, link{get_results_scenario}

Examples

```
# Example with built in package data:
d1 <- system.file("extdata", "output", "ss3sim_ts.csv",
  package = "ss3sim")
d2 <- system.file("extdata", "output", "ss3sim_scalar.csv",
  package = "ss3sim")
ss3sim_ts <- read.csv(d1)
ss3sim_scalar <- read.csv(d2)

head(calculate_re(ss3sim_ts))
head(calculate_re(ss3sim_ts, add = FALSE))
head(calculate_re(ss3sim_scalar, add = FALSE))

## Not run:
# Full example:
d <- system.file("extdata", package = "ss3sim")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
case_folder <- paste0(d, "/eg-cases")

run_ss3sim(iterations = 1, scenarios = "D0-F0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em, ss_mode = "optimized")

get_results_all(user_scenarios = "D0-F0-cod")
ss3sim_ts <- read.csv("ss3sim_ts.csv")
ss3sim_scalar <- read.csv("ss3sim_scalar.csv")

head(calculate_re(ss3sim_ts))
head(calculate_re(ss3sim_scalar, add = FALSE))

# clean up:
unlink("D0-F0-cod", recursive = TRUE)
unlink("ss3sim_ts.csv", recursive = TRUE)
```

```
unlink("ss3sim_scalar.csv", recursive = TRUE)

## End(Not run)
```

`calculate_runtime` *Calculate run time*

Description

Internal function used by `get_results_scenario` to calculate the runtime (in minutes) from a `Report.sso` file.

Usage

```
calculate_runtime(start_time, end_time)
```

Arguments

`start_time`	Vector of characters as read in from the r4ss report file
`end_time`	Vector of characters as read in from the r4ss report file

Details

This runtime includes the overhead for reading and writing the file up to the Report.sso time stamp.

Value

A numeric value representing the number of minutes of total runtime as reported by SS.

Author(s)

Cole Monnahan

`case_comp` *Write a case file for length- or age-composition data*

Description

Use R code to write arguments to the disk, which will later be used in a **ss3sim** simulation.

Usage

```
case_comp(fleets = 1, Nsamp = NULL, years = NULL, cpar = 2, type, case,
  spp)
```

Arguments

fleets	Vector of fleet numbers, where the order of fleets will dictate the order of all remaining arguments.
Nsamp	A list of length length(fleets), where each element of the list contains a vector of sample sizes for each year for that given fleet.
years	A list of length length(fleets), where each element of the list contains a vector of years for the given fleet.
cpar	A vector of cpar for each fleet.
type	A character value of "agecomp" or "lcomp", to write age- or length-composition specifications, respectively. Argument can be a vector (e.g., c("agecomp", "lcomp")) if you want the case files to be the same for length and age compositions.
case	The casenumber you want to write to. If case = 1 and type = "agecomp", then the result will be 'agecomp1'.
spp	A vector of character values argument specifying the species.

Examples

```
case_comp(fleets = 1:2, case = 30, spp = "cod",
  Nsamp = list(rep(10, 40), rep(10, 25)),
  years = list(61:100, 76:100), cpar = 2:1, type = "agecomp")
done <- file.remove("agecomp30-cod.txt")
```

---

case_deparse *Turn an argument describing an object into a character.*

Description

Turn an argument describing an object into a character.

Usage

```
case_deparse(x)
```

Arguments

x	The argument you would like to deparse. "M1-F1-D1-R1"

Details

Includes checks to make sure multiple lines will not be created.

Value

A single character value.

case_fishing *Write a case file for fishing data to the disk.*

Description

Use R code to write arguments to the disk, which will later be used in a **ss3sim** simulation.

Usage

```
case_fishing(years = 1, years_alter = NULL, fvals = 2, case, spp)
```

Arguments

years	Vector of years for which F values are specified, if there is more than one fleet or season the catches must be ordered by season:year:fishey (e.g., season1year1fishery1, season2year1fishery1, season1year2fishery1). The actual vector does not have to correspond to true years but must be the correct length (e.g., instead of `2000:2004` you can use `1:5`). Use this argument to create an index to old values. `years_alter` will use values in this vector. For example, with two seasons and one fishery that operates for 4 years you could use the following: `1:8`.
years_alter	Vector of years for the which F values will be altered. If there is more than one fishery or season, use the mapping system created in `years` because actual year values cannot be recycled. For example, to change the second season of the second year in the example above, use: `4`.
fvals	Vector of F values to be entered into `ss3.par` file, where `length(fvals) == length(years_alter)` must be true.
case	The case number you want to write to. If `case = 1`, then the result will be `'F1'`.
spp	A vector of character values argument specifying the species.

Examples

```
case_fishing(1:100, 1:100, seq(0, 0.4, length.out = 100), 2, "cod")
done <- file.remove("F2-cod.txt")
```

case_index *Write a case file for index data to the disk.*

Description

Use R code to write arguments to the disk, which will later be used in a **ss3sim** simulation.

Usage

```
case_index(fleets = 1, years = NULL, sd = 2, case, spp)
```

Arguments

fleets	Vector of fleet numbers, where the order of fleets will dictate the order of all remaining arguments.
years	A list of length length(fleets), where each element of the list contains a vector of years for the given fleet.
sd	A list of standard deviations for each fleet.
case	The case number you want to write to. If case = 1, then the result will be 'index1'.
spp	A vector of character values argument specifying the species.

Examples

```
case_index(fleets = 2, case = 1, spp = "cod", years = list(7:10), sd = 0.1)
done <- file.remove("index1-cod.txt")
```

---

case_tv *Write time varying casefiles to the disk*

Description

Use R code to write arguments to the disk, which will later be used in a **ss3sim** simulation.

Usage

```
case_tv(species, parameter, perc_change, outfile, dir_out = "cases",
  dir_models = system.file("models", package = "ss3models"), nyears = 100,
  verbose = FALSE)
```

Arguments

species	A vector of species, for which a unique case file will be generated.
parameter	A character value specifying the parameter to add deviates to. The argument must match the parameter name exactly.
perc_change	A vector of percents, which will be used to add deviates to the parameter specified in parameter. A percentage must be supplied for every year in the model.
outfile	A character value specifying the case letter and number used to save the file.
dir_out	A character value specifying the directory to save the outfile to.
dir_models	The path where the models are stored, such that file.path(dir_models, species, "om", "ss3.ctl") leads to valid ss3.ctl operating model files.
nyears	The length time-series included in the model. The length of perc_change must equal nyears.
verbose	Useful for debugging to print output to screen. Default is FALSE.

Author(s)

Peter Kuriyama

Examples

```
temp_path <- file.path(tempdir(), "cod")
dir.create(temp_path, showWarnings = FALSE)

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

om <- file.path(d, "models", "cod-om")
ig <- file.copy(om, temp_path, recursive = TRUE)
ig <- file.rename(file.path(temp_path, "cod-om"), file.path(temp_path, "om"))
verify_input(file.path(temp_path, "om"), type = "om")
ig <- file.rename(file.path(temp_path, "om", "om.ctl"),
  file.path(temp_path, "om", "ss3.ctl"))
case_tv(species = "cod", parameter = "NatM_p_1_Fem_GP_1",
perc_change = rep(0.5, 100), outfile = "G1",
dir_out = temp_path, dir_models = gsub("/cod", "", temp_path),
nyears = 100, verbose = TRUE)
unlink(temp_path, recursive = TRUE)
```

---

change_agecomp *(Depreciated) Sample age compositions from expected values*

Description

change_agecomp is a depreciated function. Please use sample_agecomp instead. change_agecomp will be removed in the next major version.

Usage

```
change_agecomp(...)
```

Arguments

...	Arguments that get passed to sample_agecomp.

change_data *Change the data that is available as output from an SS operating model.*

Description

change_data alters the data structure for a data list as read in by SS_readdat, for use in preparing the data file for an SS operating model. Original data is removed and dummy data is added, as specified, to the SS .dat file. This causes SS to produce expected values (OM "truth") when the operating model is run, from which data can be sampled. For each data type altered, change_data will add data for the fleets and years given; potentially adding many rows of redundant data. Currently, .dat files with multiple genders cannot be manipulated with change_data. calculate_data_units is used internally in ss3sim_base to create a superset of fleets and years from sample arguments, and clean_data to strip out unused data after change_data is called (see examples below). change_data is called internally automatically, but can also be used by an **ss3sim** user to manipulate data as a case, or to prepare a new OM for use in a simulation. See the vignette for more details.

Usage

```
change_data(dat_list, outfile, fleets, years, types, age_bins = NULL,
  len_bins = NULL, pop_binwidth = NULL, pop_minimum_size = NULL,
  pop_maximum_size = NULL, lcomp_constant = NULL, tail_compression = NULL,
  write_file = TRUE)
```

Arguments

dat_list	An SS data list object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.
outfile	A character value giving the location of an SS .dat file to output, if write_file=TRUE.
fleets	A numeric vector of fleets
years	A numeric vector of years
types	A vector that can take combinations of the following entries: "index", "len", "age", "cal", "mla". types controls what data structures the function acts on, with "index" changing indices/CPUE, "len" augmenting the length composition data, "age" augmenting the age composition, "cal" augmenting the conditional age at length, and "mla" augmenting the mean length at age data.
age_bins	*A numeric vector of age bins to use. If left as NULL then the age bin structure will be taken from the OM.
len_bins	*A numeric vector of length bins to use. If left as NULL then the length bin structure will be taken from the OM.
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 SS3 will fail (see notes in the SS3 manual).
pop_minimum_size	*Population minimum length bin value.

`pop_maximum_size`
: *Population maximum length bin value.

`lcomp_constant`
: *A new robustification constant for length composition data to be used. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the SS3 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`.

`tail_compression`
: *A new tail compression value to be used in SS3. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the SS3 manual for further information. A `NULL` value indicates no action, a negative value indicates to SS3 to ignore it (not use that feature).

`write_file`
: Should the `.dat` file be written? The new `.dat` file will always be returned invisibly by the function. Setting `write_file = FALSE` can be useful for testing. Note that you must supply a value to the argument `outfile`, but this argument can be set to any arbitrary value (such as `NULL`) if `write_file = FALSE`.

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

An invisible data list, and a file is written if `write_file = TRUE`.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to `run_ss3sim`. If one of these arguments is not specified in a case file, then a value of `NULL` will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan, Ian Taylor, Sean Anderson, Kelli Johnson

See Also

`sample_lcomp`, `sample_agecomp`

Other change functions: `change_em_binning`, `change_e`, `change_f`, `change_maturity`, `change_retro`, `change_tv`

Examples

```
d <- system.file("extdata", package = "ss3sim")
fleets <- 1:2
years <- c(5, 10, 15)
```

```
types <- c("len", "age")
file_in <- r4ss::SS_readdat(paste0(d, "/models/cod-om/codOM.dat"))
file_in <- change_fltname(file_in)

# Basic test with just length data, default bins:
out <- change_data(file_in, outfile = "ignore.dat", types = "len",
  years = years, fleets = fleets, write_file = FALSE)
print(out$lbin_vector)
print(out$lencomp)

# Change the length bins:
out <- change_data(file_in, "ignore.dat", types = "len",
  years = years, fleets = fleets, len_bins = 3:6, write_file = FALSE)
out$lbin_vector
out$lencomp

# Change the population length bins:
out <- change_data(file_in, "ignore.dat", types = "len",
  years = years, fleets = fleets, pop_binwidth = 1, pop_minimum_size = 5,
  pop_maximum_size = 210, write_file = FALSE)
out$binwidth
out$maximum_size
out$minimum_size

# Sample from index, length composition, age composition, catch at length,
# mean length at age data: (normally this is all done from within run_ss3sim).

index_params = list(fleets = c(1, 2), years = list(c(1, 2),
  c(10, 11)), sds_obs = c(0.1, 0.2))
lcomp_params = list(Nsamp = list(12345), fleets = 1, years = list(c(1, 5)))
agecomp_params = list(Nsamp = list(12345), fleets = c(1, 2),
  years = list(2, c(15, 16)))
calcomp_params = list(Nsamp = list(1), fleets = c(1), years = 98)
mlacomp_params = list(fleets = c(2), Nsamp = 54, years = list(c(1, 15)))
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/models/cod-om/codOM.dat")
dat_list <- r4ss::SS_readdat(f_in, verbose = FALSE)
dat_list <- change_fltname(dat_list)
data_units <- calculate_data_units(index_params = index_params,
  lcomp_params = lcomp_params, agecomp_params = agecomp_params,
  calcomp_params = calcomp_params, mlacomp_params = mlacomp_params)
data_units
dat2 <- with(data_units, change_data(dat_list = dat_list, fleets = fleets,
  years = years, types = types, write_file = FALSE))
dat3 <- clean_data(dat_list = dat2, lcomp_params = lcomp_params,
  index_params = index_params, agecomp_params = agecomp_params,
  calcomp_params = calcomp_params, mlacomp_params = mlacomp_params,
  verbose = TRUE)
```

---

change_e *Methods to alter which parameters are estimated in a SS3* `.ctl` *file.*

Description

Takes SS3 `.ctl` and `forecast.ss` files, along with a list structure which houses the data file as read in by `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. If used with `run_ss3sim` the case file should be named `E`. A suggested (default) case letter is `E` for estimation.

Usage

```
change_e(ctl_file_in = "em.ctl", ctl_file_out = "em.ctl", dat_list = NULL,
  for_file_in = "forecasts.ss", natM_type = "1Parm",
  natM_n_breakpoints = NULL, natM_lorenzen = NULL, natM_val = c(NA, NA),
  par_name = NULL, par_int = "NA", par_phase = "NA", forecast_num = 0,
  run_change_e_full = TRUE, verbose = FALSE)
```

Arguments

`ctl_file_in`	Input SS3 control file
`ctl_file_out`	Output SS3 control file
`dat_list`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`for_file_in`	Input SS3 forecast file
`natM_type`	*A character string corresponding to option 0:4 in SS3 (i.e. "1Parm", "n_breakpoints", "Lorenzen", "agespecific", "agespec_withseasinterpolate"). A value of `NA` will leave the configuration of natural mortality as specified in `ctl_file_in`.
`natM_n_breakpoints`	*A vector of ages at which you want breakpoints. Only used if you specify `natM_type = "n_breakpoints"`.
`natM_lorenzen`	*The reference age for the Lorenzen function. Only used if you specify `natM_type = "Lorenzen"`. Length should be one even if the `.ctl` has two genders.
`natM_val`	*A vector of numeric values. Interpretation of the values are dependent upon `natM_type`. If `natM_type = "agespecific"` or `natM_type = "agespec_withseasinterpolate"` the vector specifies the fixed natural mortality parameters for each integer age. Specify values in the following order: female area 1, female area 2, male area 1, male area, etc. Ensure that there is one entry per integer age x area x gender. If `natM_type = "1Param"`, `natM_type = "n_breakpoints"`, or `natM_type = "Lorenzen"` the vector specifies the initial and phase values for each natM parameter (i.e. `c(int, phase, int, phase,etc.)`, where the first two values could correspond to ages 0-2 natural mortality and the third and fourth value could correspond to ages 3-8 natural mortality). For any specified initial value, the parameter bounds will be altered to 50 percent above and below the specified initial value, if the initial value lies above or below the original bounds.
`par_name`	*A vector of values, separated by commas. Each value corresponds to a parameter that you wish to turn on or off in the `ctl_file_in`. The values will later be turned into character values and used to search for specific lines for each parameter in the `ctl_file_in`, therefore it is best to use full parameter names as they are specified in `ctl_file_in`.

par_int	*A vector of initial values, one for each parameter in par_name. Values can be NA if you do not wish to change the initial value for a given parameter.
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.
forecast_num	*Number of years to perform forecasts. For those years, the data will be removed from the dat_list, enabling SS3 to generate forecasts rather than use the data to fit the model.
run_change_e_full	*If FALSE change_e will only manipulate for forecasting, if TRUE (default) the full function capability will be ran.
verbose	When TRUE messages will be returned from the function. Often useful for debugging. The default is FALSE.

Details

Turning parameters on and off is the main function of change_e. change_e was not created with the capability of adding parameters to a .ctl file. The function can only add parameters for age specific natural mortality, and only for models with one growth morph. Furthermore, the function is designed to add complexity to the natural mortality type and not remove complexity. Therefore, the function will fail if natural mortality in the ctl_file_in is not specified as "1Param" and natM_type is anything other than NULL or "1Param".

Value

Altered versions of SS3 .ctl and forecast.ss files are written to the disk and the altered dat_list is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Kelli Johnson

See Also

Other change functions: change_data, change_em_binning, change_f, change_maturity, change_retro, change_tv

Examples

```
## Not run:
library(r4ss)
# 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)

d <- system.file("extdata", package = "ss3sim")
ctl_file <- paste0(d, "/models/cod-om/codOM.ctl")
data.old <- r4ss::SS_readdat(file.path(d, "models", "cod-om", "codOM.dat"))
change_e(ctl_file_in = ctl_file, ctl_file_out = "change_e.ctl",
         dat_list = data.old, for_file_in = "forecast.ss",
         natM_type = "n_breakpoints", natM_n_breakpoints = c(1, 4),
         natM_lorenzen = NULL, natM_val = c(.2, 3, 0.4, 5),
         par_name = c("_steep", "SizeSel_1P_1_Fishery"),
         par_int = c(0.3, 40), par_phase = c(3, 2),
         forecast_num = 0, run_change_e_full = TRUE )
# clean up
file.remove("change_e.ctl")
setwd(wd)

## End(Not run)
```

`change_em_binning` *Change population and observed length composition bins in an SS estimation model*

Description

`change_em_binning` alters the bin structure for the population and length composition data in an SS estimation model. It is done by taking the original length composition info from the EM `ss3.dat` then changing according to the user's specification. If the data file also contails conditional age-at-length data then these data will be re-binned as well.

Usage

```
change_em_binning(dat_list, dat_file_out, bin_vector, lbin_method = NULL,
  pop_binwidth = NULL, pop_minimum_size = NULL, pop_maximum_size = NULL,
  write_file = TRUE)
```

Arguments

`dat_list`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`dat_file_out`	A string providing the path to the output SS3 `.dat` file.
`bin_vector`	A numeric vector of new length bins to substitute into the `ss3.dat` file.
`lbin_method`	A numeric value of either `NULL, 1, 2, 3` to change the lbin_method for the population bin. Only supports either `NULL, 1,2` at the moment. `NULL` means to keep it unchanged.

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 SS3 will fail (see notes in the SS3 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

write_file Should the .dat file be written? The new .dat file will always be returned invisibly by the function. Setting write_file = FALSE can be useful for testing. Note that you must supply a value to the argument dat_file_out, but this argument can be set to any arbitrary value (such as NULL) if write_file = FALSE.

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_maturity, change_retro, change_tv

Examples

```
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/example-om/data.ss_new")
dat_list <- r4ss::SS_readdat(file = f_in, verbose = FALSE)
l <- change_em_binning(dat_list, dat_file_out = NULL, lbin_method = 1,
  bin_vector = seq(8, 30, by = 1), write_file = FALSE)
print(l$lbin_vector)
print(head(l$lencomp))

# An small example with conditional age-at-length re-binning:
f <- system.file("extdata", "models", "cod-om", "codOM.dat", package = "ss3sim")
d <- r4ss::SS_readdat(f, verbose = FALSE)

# Add catch at length data (and simplify the bin structure for this example)
olddat <- change_data(d, outfile = NULL, write_file = FALSE,
  types = c("len", "age", "cal"), fleets = 1, years = seq(2000, 2002),
  age_bins = 1:3, len_bins = 4:8)
olddat$agecomp
newdat <- change_em_binning(olddat, dat_file_out = NULL, bin_vector = c(4, 6, 8),
  lbin_method = 1, write_file = FALSE)
newdat$agecomp

# A larger conditional age-at-length re-rebinning example:
olddat <- change_data(d, outfile = NULL, write_file = FALSE,
 types = c("len", "age", "cal"), fleets = 1, years = seq(2000, 2005),
 age_bins = seq(1, 5), len_bins = round(seq(20, 30, length.out = 13), 1))

olddat$lbin_vector
head(olddat$lencomp)
```

```
head(olddat$agecomp)
newdat <- change_em_binning(olddat, dat_file_out = NULL, bin_vector = seq(20, 30, 2),
 lbin_method = 1, write_file = FALSE)
newdat$lbin_vector
head(newdat$lencomp)
newdat$agecomp
```

---

change_f *Alter the fishing mortality (*F*) values in an SS3* .par *file.*

Description

Takes an SS3 .par file and changes the *F* values for specified years. If used with run_ss3sim the case file should be named F. A suggested (default) case letter is F.

Usage

```
change_f(years, years_alter, fvals, par_file_in = "ss3.par",
  par_file_out = "ss3.par")
```

Arguments

years	*Vector of years for which *F* values are specified, if there is more than one fleet or season the catches must be ordered by season:year:fishey (e.g., season1year1fishery1, season2year1fishery1, season1year2fishery1). The actual vector does not have to correspond to true years but must be the correct length (e.g., instead of 2000:2004 you can use 1:5). Use this argument to create an index to old values. years_alter will use values in this vector. For example, with two seasons and one fishery that operates for 4 years you could use the following: 1:8.
years_alter	*Vector of years for the which *F* values will be altered. If there is more than one fishery or season, use the mapping system created in years because actual year values cannot be recycled. For example, to change the second season of the second year in the example above, use: 4.
fvals	*Vector of *F* values to be entered into ss3.par file, where length(fvals) == length(years_alter) must be true.
par_file_in	A string providing the path to the input SS3 .par file.
par_file_out	A string providing the path to the output SS3 .par file.

Value

A modified SS3 .par file.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Curry James Cunningham

See Also

Other change functions: change_data, change_em_binning, change_e, change_maturity, change_retro, change_tv

Examples

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

# Find the example .par file in the package data:
d <- system.file("extdata", package = "ss3sim")
par_file <- paste0(d, "/change_f/ss3.par")

change_f(years = 1:49, years_alter = 2, fvals = 9999, par_file_in =
par_file, par_file_out = paste0(temp_path, "/test.par"))
```

---

change_fltname *Standardize column name for FltSvy in event* **r4ss** *is not the newest version.*

Description

change_fltname alters the name for the fleet/survey column which is typically named FltSvy by SS_readdat, but was inconsistent in older versions (.e.g. Fleet was used for mean size-at-age).

Usage

```
change_fltname(dat_list)
```

Arguments

dat_list	An SS data list object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.

Value

An invisible data list.

Author(s)

Kelli Johnson

Examples

```
d <- system.file("extdata", package = "ss3sim")
file_in <- file.path(d, "Simple", "simple.dat")
# Here one should include the argument \code{section = 2}
# but this \code{.dat} file does not have multiple sections.
dat_in <- r4ss::SS_readdat(file_in, verbose = FALSE)
dat_fixed <- change_fltname(dat_in)
# Check mean size-at-age
names(dat_fixed$MeanSize_at_Age_obs)[3] == "FltSvy"
```

---

change_index *(Depreciated) Sample the biomass with observation error*

Description

change_index is a depreciated function. Please use sample_index instead. change_index will be removed in the next major version.

Usage

```
change_index(...)
```

Arguments

...	Arguments that get passed to sample_index.

---

change_lcomp *(Depreciated) Sample length compositions from expected values*

Description

change_lcomp is a depreciated function. Please use sample_lcomp instead. change_lcomp will be removed in the next major version.

Usage

```
change_lcomp(...)
```

Arguments

...	Arguments that get passed to sample_lcomp.

`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 `SS_readdat` with those specified in `lcomp_constant`. It then writes a new file with name `dat_file_out` into the working directory. If used with `run_ss3sim` the case file should be named `lcomp_constant`. A suggested case letter is `C`.

Usage

```
change_lcomp_constant(lcomp_constant, dat_list, dat_file_out,
  write_file = TRUE)
```

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 SS3 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`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`dat_file_out`	A string providing the path to the output SS3 `.dat` file.
`write_file`	Should the data file be written to disk?

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 SS3 `.dat` file, and that file returned invisibly (for testing) as a vector of character lines.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to `run_ss3sim`. If one of these arguments is not specified in a case file, then a value of `NULL` will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan

change_maturity *Alter a control file to specify the SS3 maturity option*

Description

Alter a control file to specify the maturity option in SS3. You could use this function to, for example, tell SS3 to read empirical age-fecundity and body weight-at-age data from `wtatage.ss`.

Usage

```
change_maturity(ctl_file_in = "em.ctl", ctl_file_out = "em.ctl",
  maturity_option = 1L)
```

Arguments

`ctl_file_in`	A string providing the path to the input SS3 `.ctl` file.
`ctl_file_out`	A string providing the path to the output SS3 `.ctl` file.
`maturity_option`	*An integer specifying `1` for length logistic, `2` for age logistic, `3` to read age-maturity for each female, `4` to read age-fecundity for each female growth patern, or `5` to read empirical age-fecundity and body weight-at-age from a separate file (`wtatage.ss`).

Value

A modified SS3 control file.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to `run_ss3sim`. If one of these arguments is not specified in a case file, then a value of `NULL` will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Sean C. Anderson

See Also

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

Examples

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

# Locate the package data:
ctlfile <- system.file("extdata", "models", "cod-em",
 "codEM.ctl", package = "ss3sim")

# Change the maturity option from 1 to 5:
change_maturity(ctlfile, "test.ctl", maturity_option = 5L)

unlink("test.ctl")
setwd(wd)
```

---

change_rec_devs *Replace recruitment deviations*

Description

This function replaces the recruitment deviations in the `ss3.par` file with those specified in `recdevs_new`, as well as a comment (for debugging). It then writes a new file with name `par_file_out` into the working directory.

Usage

```
change_rec_devs(recdevs_new, par_file_in = "ss3.par",
  par_file_out = "ss3.par")
```

Arguments

`recdevs_new`	A vector of new recruitment deviations.
`par_file_in`	A string providing the path to the input SS3 `.par` file.
`par_file_out`	A string providing the path to the output SS3 `.par` file.

Details

This function does not need to be specified in a case file if you are running and ss3sim simulation through case files with `run_ss3sim`.

Value

A modified SS3 `.par` file.

Author(s)

Cole Monnahan

Examples

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

par_file <- system.file("extdata", "models", "cod-om", "ss3.par",
  package = "ss3sim")
change_rec_devs(recdevs_new = rlnorm(100), par_file_in = par_file,
  par_file_out = paste0(temp_path, "/test.par"))
```

`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 SS3 starter file to run a retrospective analysis. If used with `run_ss3sim` the case file should be named `R`. A suggested (default) case letter is `R`.

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 SS3 `starter.ss` file.
`str_file_out`	A string providing the path to the output SS3 `starter.ss` file.
`retro_yr`	*Which retrospective year to enter into the starter file. Should be 0 (no retrospective analysis) or a negative value.

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 SS3 starter file.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to `run_ss3sim`. If one of these arguments is not specified in a case file, then a value of `NULL` will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Sean C. Anderson

See Also

Other change functions: change_data, change_em_binning, change_e, change_f, change_maturity, 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_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 SS_readdat with those specified in `tail_compression`. It then writes a new file with name `dat_file_out` into the working directory. If used with run_ss3sim the case file should be named `tail_compression`. A suggested case letter is `T`.

Usage

```
change_tail_compression(tail_compression, dat_list, dat_file_out,
  write_file = TRUE)
```

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 SS3 manual for further information. A NULL value indicates no action, a negative value indicates to SS3 to ignore it (not use that feature).

dat_list	An SS data list object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.
dat_file_out	A string providing the path to the output SS3 .dat file.
write_file	A logical value indicating if the altered file should be written to the disk. The default value is TRUE.

Value

A modified SS3 .dat file, and that file returned invisibly (for testing) as a vector of character lines.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan

---

change_tv *Methods to include time-varying parameters in an SS3 operating model*

Description

change_tv takes SS3 .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", par_file_in = "ss3.par",
  par_file_out = "ss3.par", str_file_in = "starter.ss",
  str_file_out = "starter.ss", rpt_file_in = "Report.sso")
```

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" below for help on specifying this argument.
ctl_file_in	A string providing the path to the input SS3 .ctl file.
ctl_file_out	A string providing the path to the output SS3 .ctl file.

dat_file_in	A string providing the path to the input SS3 .dat file.
dat_file_out	A string providing the path to the output SS3 .dat file.
par_file_in	A string providing the path to the input SS3 .par file.
par_file_out	A string providing the path to the output SS3 .par file.
str_file_in	A string providing the path to the input SS3 starter.ss file.
str_file_out	A string providing the path to the output SS3 starter.ss file.
rpt_file_in	Input SS3 report file

Details

Although there are three ways to implement time-varying parameters within SS3, **ss3sim** and change_tv only use the environmental variable option. Within SS3, 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 SS3 version 3.24f manual for more information.

Value

The function creates modified versions of the .par, .starter, .ctl, and .dat files.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

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: $ln(q'[y]) = ln(q) + link * env[y]$

Passing arguments to `change_tv` through `run_ss3sim`

(1) create a case file with an arbitrary letter not used elsewhere (anything but D, E, F, or R) and (2) include the line `function_type; change_tv` in your case file. For example, you might want to use M for natural mortality, S for selectivity, or G for growth.

Author(s)

Kotaro Ono, Carey McGilliard, and Kelli Johnson

See Also

Other change functions: `change_data`, `change_em_binning`, `change_e`, `change_f`, `change_maturity`, `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)

# Find the SS3 "Simple" model in the package data:
d <- system.file("extdata", package = "ss3sim")
simple <- paste0(d, "/Simple")
dir.create("Simple")
file.copy(simple, ".", recursive = TRUE)
setwd("Simple")

# Run SS3 to create control.ss_new and Report.sso:
system("ss3_24o_safe starter.ss -noest")

change_tv(change_tv_list = list("NatM_p_1_Fem_GP_1" = c(rep(0, 20),
      rep(.1, 11)), "SR_BH_steep"=rnorm(31, 0, 0.05)), ctl_file_in =
  "control.ss_new", ctl_file_out = "example.ctl", dat_file_in =
  "simple.dat", dat_file_out = "example.dat")

# Clean up:
setwd("../")
unlink("Simple")
setwd(wd)

## End(Not run)
```

change_year *Methods to change the years in an ss3sim model*

Description

change_year takes SS3 .ctl, .dat, .par, starter.ss, and .forecast files and changes the start and end year of the model. change_year works with **ss3sim** operating or estimation models.

Usage

```
change_year(year_begin = 1, year_end = 100, burnin = 0,
  ctl_file_in = NULL, ctl_file_out = "new.ctl", dat_file_in = NULL,
  dat_file_out = "new.dat", par_file_in = NULL, par_file_out = "new.ss",
  str_file_in = NULL, str_file_out = "starter.ss", for_file_in = NULL,
  for_file_out = "forecast.ss", verbose = FALSE)
```

Arguments

year_begin	Desired start year
year_end	Desired end year
burnin	Length of burnin period. Default is zero for an OM. Use burnin for EM models, to establish a period with no fishing. Note that the value should correspond to the number of years, not a year value.
ctl_file_in	A string providing the path to the input SS3 .ctl file.
ctl_file_out	Output SS3 control file, if NULL the file will be named the same as the ctl_file_in
dat_file_in	A string providing the path to the input SS3 .dat file.
dat_file_out	Output SS3 data file, if NULL the file will be named the same as the dat_file_in
par_file_in	A string providing the path to the input SS3 .par file.
par_file_out	Output SS3 parameter file, if NULL the file will be named the same as the par_file_in
str_file_in	A string providing the path to the input SS3 starter.ss file.
str_file_out	Output SS3 starter file, if NULL the file will be named the same as the str_file_in
for_file_in	Input SS3 forecast file
for_file_out	Output SS3 forecast file, if NULL the file will be named the same as the for_file_in
verbose	Logical argument that is passed on to internal calls to r4ss functions for reading and writing SS3 files. Setting verbose to TRUE may be useful for troubleshooting.

Details

Operating models and estimation models will not have all of the same files, thus if the file does not exist change the _file_in to NULL. The code will add data for all years specified, minus the burnin period, if the data type is present in the dat_file_in file. The manipulation performed on the dat_file_in file is not complete and users will need to specify data for years which are

deleted. The function removes all composition data except for the first year. To remove data use sample_index, sample_lcomp, or sample_agecomp. For models that use the .forecast file, all references to years must be made with relative values (i.e., 0 or negative integers).

Value

The function creates modified versions of the .par, .dat, .ctl, .starter, and .forecast files.

Author(s)

Kelli Johnson

Examples

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

# Find the SS3 "Simple" model in the package data:
d <- system.file("extdata", package = "ss3sim")
simple <- paste0(d, "/Simple")
dir.create("Simple")
file.copy(simple, ".", recursive = TRUE)
setwd("Simple")

# Run SS3 to create control.ss_new and Report.sso:
system("SS3_24o_safe starter.ss -noest")

change_year(year_begin = 1, year_end = 100, burnin = 25,
 ctl_file_in = "control.ss_new", ctl_file_out = "change_year.ctl",
 dat_file_in = "simple.dat", dat_file_out = "change_year.dat",
 par_file_in = "ss3.par", par_file_out = "change_year.par",
 str_file_in = "starter.ss", str_file_out = "change_year_starter.ss",
 for_file_in = "forecast.ss", for_file_out = "change_year_forecast.ss")

# Clean up:
setwd("../")
unlink("Simple")
setwd(wd.old)

## End(Not run)
```

---

check_data *Check that the SS3 data file looks correct*

Description

Check that the SS3 data file looks correct

Usage

```
check_data(x)
```

Arguments

x	An SS3 data list object as read in by SS_readdat.

---

cleanup_ss3 *Clean up after an SS3 run*

Description

Removes all of the unwanted output files from the specified directory.

Usage

```
cleanup_ss3(dir_name, clean_vector = c("admodel.*", "ss3.eva", "fmin.log",
  "*.rpt", "variance", "ss3.b0*", "ss3.p0*", "ss3.r0*", "ss3.bar", "ss3.cor",
  "ss3.log", "ss3.rep", "checkup.sso", "cumreport.sso",
  "derived_posteriors.sso ", "echoinput.sso", "parmtrace.sso",
  "posterior_vectors.sso", "posteriors.sso", "rebuild.sso", "sis_table.sso",
  "data.ss_new", "forecast.ss_new", "control.ss_new", "starter.ss_new",
  "wtatage.ss_new"))
```

Arguments

dir_name	The directory of interest, the function ignores case (i.e. names can be specified as lower or upper case)
clean_vector	A vector of characters specifying the unwanted files to be removed. The function allows the use of wildcards (i.e. "*").

Author(s)

Kelli Johnson

clean_data *Given sampling arguments remove ("clean") all data in a .dat file that is not specified*

Description

This prepares a `.dat` file to be used by an EM, whereas before it may have had leftover data from sampling purposes. See examples in `change_data`.

Usage

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

Arguments

`dat_list`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`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`.
`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, change_data

Other sampling functions: `sample_agecomp`, `sample_calcomp`, `sample_index`, `sample_lcomp`, `sample_mlacomp`, `sample_wtatage`

copy_ss3models *Copy the operating and estimation models and create a folder structure*

Description

Copy the operating and estimation models and create a folder structure

Usage

```
copy_ss3models(model_dir, scenarios, iterations = 1:100, type = c("om",
  "em"))
```

Arguments

model_dir	A directory containing the operating or estimation model. Each folder should be named according to a scenario ID. (See the vignette `vignette("ss3sim-vignette")` or `get_caseargs` for details on the scenario ID format.)
scenarios	Which scenarios to copy to. Supply a vector of character elements.
iterations	A numeric vector of the iterations to copy to. The function will create the folders as needed.
type	Are you copying operating or estimation models? This affects whether the model folder gets named "om" or "em"

Value

An invisble boolean for whether that iteration already existed. A set of nested folders starting with the scenario ID, then the iterations, then "om" or "em", and then the SS model files.

Author(s)

Sean Anderson, Kelli 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, type = "om", iterations =
  1:3, scenarios = "D0-F0-testing")
# Now look at your working directory in your file system

# Copy the estimation model with two scenario IDs:
copy_ss3models(model_dir = om_folder, type = "em", iterations = 1:2,
  scenarios = c("D1-F0-testing", "D2-F0-testing"))
# (Note that all the scenario argument does here is affect the
```

```
# folder names.)

# Clean up:
unlink("D0-F0-testing", recursive = TRUE)
unlink("D1-F0-testing", recursive = TRUE)
unlink("D2-F0-testing", recursive = TRUE)
```

`create_argfiles` *Create template argument input files*

Description

Creates template input files based on the argument lists for specified functions. Look in your working directory for the template files. Change the case ID number (defaults to `0`) and the species identifier to a three letter identifier. To use one of the built-in model setups, use one of `cod`, `sar`, or `fla` for cod, sardine, or flatfish. An example filename would be `M1-sar.txt` or `lcomp2-fla.txt`.

Usage

```
create_argfiles(functions = c(`lcomp0-spp` = "sample_lcomp", `agecomp0-spp` =
  "sample_agecomp", `index0-spp` = "sample_index", `F0-spp` = "change_f",
  `R0-spp` = "change_retro", `E0-spp` = "change_e", `X0-spp` = "change_tv"),
  ext = ".txt", delim = "; ", ignore = c("file", "dir", "make_plot"), ...)
```

Arguments

`functions`	A named vector. The names correspond to the filenames that will get written. The values correspond to the functions to grab the arguments from.
`ext`	The file extension to create the configuration files with. Defaults to `".txt"`.
`delim`	The delimiter. Defaults to `"; "`.
`ignore`	A vector of character object of arguments to ignore in the arguments. Found via `grep` so can be part of an argument name.
`...`	Anything else to pass to `write.table`.

Details

The first column in the text files denotes the argument to be passed to a function. The second argument denotes the value to be passed. You can use any simple R syntax. For example: `c(1, 2, 4)`, or `seq(1, 100)` or `1:100` or `matrix()`. Character objects don't need to be quoted. However, be careful not to use your delimiter (set up as a semicolon) anywhere else in the file besides to denote columns.

The function `change_tv` is a special case. To pass arguments to `change_tv` through a `run_ss3sim`: (1) create a case file with an arbitrary letter not used elsewhere (anything but D, E, F, or R) and include the line `function_type; change_tv` in your case file. For example, you might want to use M for natural mortality, S for selectivity, or G for growth.

This function (`create_argfiles`) automatically adds a line `function_type; change_tv` to the top of a case file `X0-spp.txt` as a starting point for `change_tv`.

Author(s)

Sean Anderson

Examples

```
## Not run:
create_argfiles()
# Some example input lines:
#
# year1; 1990
# years; 1990:2000
# years; c(1980, 1990, 1995)
# survey_type; fishery

## End(Not run)
```

---

expand_scenarios *Create vectors of scenario IDs*

Description

Create vectors of scenarios from inputs. For passing to `run_ss3sim`, or `get_results_all`. Default case values are the base case (`0`).

Usage

```
expand_scenarios(cases = list(D = 0, E = 0, F = 0, M = 0, R = 0),
  species = c("cod", "fla", "sar"))
```

Arguments

`cases`	A named list of cases. The names in the list are the case IDs and the values are the case values.
`species`	Vector of 3-letter character IDs designating the species/stock.

Value

A character vector of scenario IDs. The case IDs will be alphabetically sorted.

Author(s)

Cole Monnahan and Sean Anderson

See Also

`run_ss3sim`, `get_results_all`

Examples

```
expand_scenarios()
expand_scenarios(cases = list(D = 0:3, E = 0, F = 0, M = 0, R = 0),
species = "cod")
```

`extract_expected_data` *Extract the expected data values*

Description

Read in a `data.ss_new` file, move the expected values up in the file, and write it back out to a new data file.

Usage

```
extract_expected_data(data_ss_new = "data.ss_new", data_out = "ss3.dat")
```

Arguments

`data_ss_new`	The location of the `.ss_new` file that was generated from a run of SS.
`data_out`	The location of the `.ss_new` file that was generated from a run of SS.

Author(s)

Kotaro Ono

`facet_form` *A helper function for building a ggplot facet. Used internally by the plotting functions.*

Description

A helper function for building a ggplot facet. Used internally by the plotting functions.

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.
`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.

Value

A formula which can be used in `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`, `fill_across`

`get_args` *Take a csv file, read it, and turn the first column into the list names and the second column into the list values.*

Description

Take a csv file, read it, and turn the first column into the list names and the second column into the list values.

Usage

```
get_args(file)
```

Arguments

file	The file name as character

---

get_bin *Get SS3 binary/executable location in package*

Description

Get SS3 binary/executable location in package

Usage

```
get_bin(bin_name = "ss3_24o_opt")
```

Arguments

bin_name	Name of SS3 binary

Value

The path to an SS3 binary. If using the GitHub version of the package, this will be an internal binary. Otherewise, this function will search for a version of the binary in your path. See the ss3sim vignette.

Examples

```
## Not run:
get_bin()

## End(Not run)
```

---

get_bin_info *Get the parameter values for change_bin*

Description

This function organizes arguments for other functions needed by change_bin.

Usage

```
get_bin_info(dat)
```

Arguments

dat	A list of sample arguments from all sampling functions

get_caseargs *Take a scenario ID and return argument lists*

Description

This function calls a number of internal functions to go from a unique scenario identifier like "D1-E2-F3-M0-R4-cod" and read the corresponding input files (e.g. "M0-cod.txt") that have two columns: the first column contains the argument names and the second column contains the argument values. The two columns should be separated by a semicolon. The output is then returned in a named list with the intention of passing these to run_ss3sim or ss3sim_base.

Usage

```
get_caseargs(folder, scenario, ext = ".txt", case_files = list(F = "F", D =
  c("index", "lcomp", "agecomp")))
```

Arguments

folder	The folder to look for input files in.
scenario	A character object that has the cases separated by the "-" delimiter. The combination of cases and stock ID is referred to as a scenario. E.g. "D0-E0-F0-M0-R0-cod". See the Details section.
ext	The file extension of the input files. Defaults to ".txt".
case_files	A named list that relates the case IDs (e.g. "D") to the files to read the arguments from (e.g. c("index", "lcomp","agecomp")). See the Details section.

Details

Let's start with an example scenario "D0-E1-F0-M0-R0-cod". The single capital letters refer to case IDs. The numbers refer to the case numbers. The last block of text (cod) represents the stock ID (any alphanumeric string of text will work) and is to help the user identify different "stocks" (intended to represent different SS3 model setups).

The stock IDs should correspond to how the case files are named and the case IDs should correspond to the cases described by the case_files. The case file names will correspond to the list values plus the stock ID. For example list(D = c("index", "lcomp", "agecomp")) combined with the stock ID cod means that the case D1 will refer to the case files index-cod.txt, lcomp-cod.txt, agecomp-cod.txt.

The case argument plain text files should have arguments in the first column that should be passed on to functions. The names should match exactly. The second column (delimited by a semicolon) should contain the values to be passed to those arguments. Multiple words should be enclosed in quotes.

You can use any simple R syntax to declare argument values. For example: c(1, 2, 4), or seq(1, 100), or 1:100, or matrix(), or NULL. Character objects don't need to be quoted, but can be if you'd like. However, be careful not to use the delimiter (set up as a semicolon) anywhere else in the file besides to denote columns. You can add comments after any # symbol just like in R.

Internally, the functions evaluate in R any entries that have no character values (e.g. `1:100`), or have an alpha-numeric character followed by a `(`. Anything that is character only or has character mixed with numeric but doesn't have the regular expression `"[A-Za-z0-9]("` gets turned into a character argument. (`NA` and `NULL` are special cases that are also passed on directly.)

Value

A (nested) named list. The first level of the named list refers to the `case_files`. The second level of the named list refers to the argument names (the first column in the input text files). The contents of the list are the argument values themselves (the second column of the input text files).

Examples

```
# Find the example data folders:
case_folder <- system.file("extdata", "eg-cases", package =
  "ss3sim")

# An example using the cases defined by default:
get_caseargs(case_folder, scenario = "D0-F0-cod")

# With a custom time-varying case for selectivity, which we'll call
# the S case. Here, we'll need to define which file the case S should
# read from ("S*-cod.txt"):
get_caseargs(case_folder, scenario = "D0-E0-F0-M0-R0-S0-cod",
  case_files = list(E = "E", D = c("index", "lcomp", "agecomp"), F =
    "F", M = "M", R = "retro", S = "S"))
```

---

get_caseval *Take a scenario ID and a case type and return the case number*

Description

Take a scenario ID and a case type and return the case number

Usage

```
get_caseval(scenario, case)
```

Arguments

`scenario`	A character object with the cases. E.g. `"M1-F1-D1-R1"`
`case`	The case you want to extract. E.g. `"M"`

get_fish600_casefolder *Get the folder location of the FISH600 case files*

Description

This function is used by some developers of **ss3sim** for simulations. This function links to the "cases" folder in `extdata`.

Usage

```
get_fish600_casefolder()
```

Value

A character object showing the location of the FISH600 case files in the package `extdata` folder.

get_model_folder *Get the folder location of an included SS3 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

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

Usage

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

Arguments

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

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 SS3 simulation output*

Description

This high level function extracts results from SS3 model runs. Give it a directory which contains directories for different "scenario" runs, within which are replicates and potentially bias adjustment runs. 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, parallel = FALSE)
```

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 replicates 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`.
`parallel`	Should the function be run on multiple cores? You will need to set up parallel processing as shown in `run_ss3sim`.

Value

Creates two .csv files in the current working directory: `ss3sim_ts.csv` and `ss3sim_scalar.csv`.

Author(s)

Cole Monnahan, Merrill Rudd

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 SS_output list from a model run. Returns a data.frame of the results for 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 Johnson

See Also

Other get-results: get_results_all, get_results_scalar, get_results_scenario, get_results_timeseries

get_results_scalar *Extract scalar quantities from a model run.*

Description

Extract scalar quantities from an 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 SS3 simulation results for one scenario.*

Description

Function that extracts results from all replicates inside a supplied scenario folder. The function writes 3 .csv files to the scenario folder: (1) scalar metrics with one value per replicate (e.g. R_0, h), (2) a timeseries data ('ts') which contains multiple values per replicate (e.g. SSB_y for a range of years y), and (3) [currently disabled and not tested] residuals on the log scale from the surveys across all replicates. The function `get_results_all` loops through these .csv files and combines them together into a single "final" dataframe.

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 replicates were added since the last time it was called, or any changes were made to this function.

Author(s)

Cole Monnahan

See Also

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

Examples

```
## Not run:
d <- system.file("extdata", package = "ss3sim")
case_folder <- paste0(d, "/eg-cases")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
run_ss3sim(iterations = 1:2, scenarios =
  c("D0-F0-G0-S0-cod"),
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = FALSE)
get_results_scenario(c("D0-F0-G0-S0-cod"))

## End(Not run)
```

get_results_timeseries *Extract time series from a model run.*

Description

Extract time series from an SS_output list from a model run. Returns a data.frame of the results for SSB, recruitment and effort by year.

Usage

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

Arguments

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

Author(s)

Cole Monnahan

See Also

Other get-results: get_results_all, get_results_derived, get_results_scalar, get_results_scenario

get_sigmar *Get recruitment deviation sigma*

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, eg. "myOM".

Author(s)

Kelli Johnson

id_scenarios *Identify ss3sim scenarios within a directory*

Description

Identify ss3sim scenarios within a directory

Usage

```
id_scenarios(directory)
```

Arguments

directory	The directory which contains scenario folders with results.

Value

A character vector of folders

Author(s)

Merrill Rudd

pastef *Paste with "/" as the separator*

Description

Paste with "/" as the separator

Usage

```
pastef(...)
```

Arguments

...	Objects to paste together

`plot_scalar_boxplot` *Print scalar values as boxplots.*

Description

Print scalar values as boxplots.

Usage

```
plot_scalar_boxplot(data, x, y, horiz = NULL, horiz2 = NULL, vert = NULL,
  vert2 = 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. Column should be a factor (e.g. "F" or "species").
`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.
`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.
`relative.error`	Boolean for whether the y-axis should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1,1)`, the y axis label is changed automatically, and a red line at y=0 is added.
`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** simulation scalar and timeseries (ts) objects. They are designed to quickly explore simulation output, rather than publication-level figures. The functions use the `aes_string` function within **ggplot2** such that arguments are passed as characters that refer to columns of `data`.

Note that there are some subtle differences between the functions. Scalar plots require a value for `x`, while for ts plots `x` is invalid because it is fixed internally as 'year', since it makes no sense to use another column. Boxplots cannot have a color mapped to them like points or lines, and thus `color` is not a valid argument. The ts point and line plots are grouped internally by 'ID', which is a combination of scenario and replicate.

Output

These functions print the ggplot object, but also return it invisibly for saving or printing again later.

Author(s)

Cole Monnahan

Examples

```
scalar_dat$depletion <- with(scalar_dat,
  (depletion_om - depletion_em) / depletion_om)
plot_scalar_boxplot(scalar_dat, x = "E", y = "depletion", horiz = "D",
  relative.error = TRUE)
```

---

plot_scalar_points *Plot scalar values as points.*

Description

Plot scalar values as points.

Usage

```
plot_scalar_points(data, x, y, horiz = NULL, horiz2 = NULL, vert = NULL,
  vert2 = NULL, 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. Column should be a factor (e.g. "F" or "species").
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.
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.
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 should be interpreted as relative error. If TRUE, ylim is set to c(-1,1), the y axis label is changed automatically, and a red line at y=0 is added.
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** simulation scalar and timeseries (ts) objects. They are designed to quickly explore simulation output, rather than publication-level figures. The functions use the aes_string function within **ggplot2** such that arguments are passed as characters that refer to columns of data.

Note that there are some subtle differences between the functions. Scalar plots require a value for x, while for ts plots x is invalid because it is fixed internally as 'year', since it makes no sense to use another column. Boxplots cannot have a color mapped to them like points or lines, and thus color is not a valid argument. The ts point and line plots are grouped internally by 'ID', which is a combination of scenario and replicate.

Output

These functions print the ggplot object, but also return it invisibly for saving or printing again later.

Author(s)

Cole Monnahan

Examples

```
scalar_dat$depletion <- with(scalar_dat,
  (depletion_om - depletion_em) / depletion_om)
plot_scalar_points(scalar_dat, x = "E", y = "depletion", horiz = 'D',
  color = "max_grad", relative.error = TRUE)
```

---

plot_ts_boxplot *Plot timeseries values as boxplots.*

Description

Plot timeseries values as boxplots.

Usage

```
plot_ts_boxplot(data, y, horiz = NULL, horiz2 = NULL, vert = NULL,
  vert2 = 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.
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.
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.
relative.error	Boolean for whether the y-axis should be interpreted as relative error. If TRUE, ylim is set to c(-1,1), the y axis label is changed automatically, and a red line at y=0 is added.
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** simulation scalar and timeseries (ts) objects. They are designed to quickly explore simulation output, rather than publication-level figures. The functions use the aes_string function within **ggplot2** such that arguments are passed as characters that refer to columns of data.

Note that there are some subtle differences between the functions. Scalar plots require a value for x, while for ts plots x is invalid because it is fixed internally as 'year', since it makes no sense to use another column. Boxplots cannot have a color mapped to them like points or lines, and thus color is not a valid argument. The ts point and line plots are grouped internally by 'ID', which is a combination of scenario and replicate.

Output

These functions print the ggplot object, but also return it invisibly for saving or printing again later.

Author(s)

Cole Monnahan

Examples

```
## Not run:
ts_dat$SpawnBio <- with(ts_dat, (SpawnBio_om-SpawnBio_em)/SpawnBio_om)
# Merge in max_grad, a performance metric, to use for color
ts_dat <- merge(scalar_dat[, c("ID", "max_grad")], ts_dat)
plot_ts_boxplot(ts_dat, y = "SpawnBio", horiz = "D", vert = "E",
  relative.error = TRUE)

## End(Not run)
```

`plot_ts_lines` *Plot timeseries values as lines.*

Description

Plot timeseries values as lines.

Usage

```
plot_ts_lines(data, 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`.
`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.
`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.
`relative.error`	Boolean for whether the y-axis should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1,1)`, the y axis label is changed automatically, and a red line at y=0 is added.
`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** simulation scalar and timeseries (ts) objects. They are designed to quickly explore simulation output, rather than publication-level figures. The functions use the `aes_string` function within **ggplot2** such that arguments are passed as characters that refer to columns of `data`.

Note that there are some subtle differences between the functions. Scalar plots require a value for `x`, while for ts plots `x` is invalid because it is fixed internally as 'year', since it makes no sense to use another column. Boxplots cannot have a color mapped to them like points or lines, and thus `color` is not a valid argument. The ts point and line plots are grouped internally by 'ID', which is a combination of scenario and replicate.

Output

These functions print the ggplot object, but also return it invisibly for saving or printing again later.

Author(s)

Cole Monnahan

Examples

```
ts_dat$SpawnBio <- with(ts_dat, (SpawnBio_om-SpawnBio_em)/SpawnBio_om)
# Merge in max_grad, a performance metric, to use for color
ts_dat <- merge(scalar_dat[, c("ID", "max_grad")], ts_dat)
plot_ts_lines(ts_dat, y = "SpawnBio", horiz = "D", vert = "E",
  relative.error = TRUE, color = "max_grad")
```

---

plot_ts_points *Plot timeseries values as points.*

Description

Plot timeseries values as points.

Usage

```
plot_ts_points(data, 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.
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.
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.
relative.error	Boolean for whether the y-axis should be interpreted as relative error. If TRUE, ylim is set to c(-1,1), the y axis label is changed automatically, and a red line at y=0 is added.
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** simulation scalar and timeseries (ts) objects. They are designed to quickly explore simulation output, rather than publication-level figures. The functions use the aes_string function within **ggplot2** such that arguments are passed as characters that refer to columns of data.

Note that there are some subtle differences between the functions. Scalar plots require a value for x, while for ts plots x is invalid because it is fixed internally as 'year', since it makes no sense to use another column. Boxplots cannot have a color mapped to them like points or lines, and thus color is not a valid argument. The ts point and line plots are grouped internally by 'ID', which is a combination of scenario and replicate.

Output

These functions print the ggplot object, but also return it invisibly for saving or printing again later.

Author(s)

Cole Monnahan

Examples

```
ts_dat$SpawnBio <- with(ts_dat, (SpawnBio_om-SpawnBio_em)/SpawnBio_om)
# Merge in max_grad, a performance metric, to use for color
ts_dat <- merge(scalar_dat[, c("ID", "max_grad")], ts_dat)
plot_ts_points(ts_dat, y = "SpawnBio", horiz = "D", vert = "E",
  relative.error = TRUE, color = "max_grad")
```

---

profile_fmsy *Determine Fmsy for a given operating model*

Description

Runs an operating model over a range of fishing mortality levels to determine the profile of F values from which Fmsy can be determined.

Usage

```
profile_fmsy(om_in, results_out, start = 0, end = 1.5, by_val = 0.01,
  ss_mode = c("safe", "optimized"))
```

Arguments

om_in	A directory for an **ss3sim** operating model.
results_out	A directory to place the results
start	Lower fishing mortality level
end	Upper fishing mortality level
by_val	Interval in which you wish to increment the fishing mortality level
ss_mode	SS3 binary option. One of "safe" for the safe version of SS3 or "optimized" for the optimized version of SS3. The relevant binary needs to be in your system's path. See the vignette vignette("ss3sim-vignette", package = "ss3sim") for details and links to the binary files. Defaults to safe mode.

Details

This function extracts the number of years from the model dat file and then runs at a constant level of fishing for each year, extracting the catch in the last year. This assumes the length of the model is long enough to reach an equilibrium catch. The user is responsible for ensuring this fact.

Value

Creates a plot and a table with catches and F values (see the results_out folder). Also invisibly returns the Fmsy table as a data frame.

Examples

```
## Not run:
d <- system.file("extdata", package = "ss3sim")
omfolder <- paste0(d, "/models/cod-om")


fmsy.val <- profile_fmsy(om_in = omfolder, results_out = "fmsy",
  start = 0.1, end = 0.2, by_val = 0.05)

## End(Not run)
```

---

rename_ss3_files *Rename SS3-version-specific files*

Description

Rename SS3-version-specific files

Usage

```
rename_ss3_files(path, ss_bin, extensions)
```

Arguments

path	The path to the folder with the files.
ss_bin	A character value giving the SS binary name
extensions	A character vector of file extensions to rename without periods preceding the values.

Author(s)

Sean C. Anderson

---

run_bias_ss3 *Determine level of bias adjustment for SS3 runs*

Description

Determine level of bias adjustment from multiple SS3 runs. IMPORTANT: The Hessian must be calculated for the SS3 runs that this function uses.

Usage

```
run_bias_ss3(dir, outdir, nsim, conv_crit = 0.2)
```

Arguments

dir	Folder for all of the bias adjustment runs (e.g. "F1-D1-cod/bias" which must contain numbered folders for the nsim runs, e.g. "F1-D1-cod/bias/1/", "F1-D1-cod/bias/2/", ..., "F1-D1-cod/bias/10/" if there are nsim = 10 bias adjustment runs)
outdir	Folder containing the run folders for a given scenario (e.g. "F1-D1-cod" that contains "F1-D1-cod/1/" "F1-D1-cod/2/", etc.)
nsim	number of bias adjustment runs conducted for a particular scenario (e.g. 10)
conv_crit	The maximum percentage of bias iterations that can produce a non-invertible Hessian before a warning will be produced. If this percentage is exceeded then a file WARNINGS.txt will be produced. Currently, the simulations will continue to run.

Details

This function:

- uses the **r4ss** package to read in output from n SS3 runs,
- uses Ian Taylor's **r4ss** function to find values for the n bias adjustment parameters for each run,
- takes the average over runs for each bias adjustment parameter
- writes out the unaveraged and averaged (AdjustBias.DAT and AvgBias.DAT, respectively) bias adjustment parameters to the dir folder

- takes a `control.ss_new` file from one of the n SS runs, changes the n bias adjustment parameters, and writes the whole updated `control.ss_new` file with new bias adjustment parameters to an `em.ctl` file

Author(s)

Carey McGilliard

References

Methot, R. D. and Taylor, I. G. (2011). Adjusting for bias due to variability of estimated recruitments in fishery assessment models. Can. J. Fish. Aquat. Sci., 68(10):1744-1760.

See Also

`run_ss3sim`, `ss3sim_base`, `run_ss3model`, `bias_ss3`

Examples

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

d <- system.file("extdata", package = "ss3sim")
case_folder <- paste0(d, "/eg-cases")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
wd <- getwd()
setwd(temp_path)
# (Note that bias_nsim should be bigger, say 10, but it is set to 2
# here so the example runs faster.)
run_ss3sim(iterations = 1:1, scenarios = "D1-F0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = TRUE, bias_nsim = 2)
setwd(wd)

## End(Not run)
```

`run_ss3model` *Run an operating or estimation model for a specified set of scenario IDs*

Description

This function takes care of calling SS3. Importantly, it parses whether the user is on Unix or Windows and calls the binary correctly. This lower-level function is meant to be called by higher level functions such as `run_ss3sim`, `ss3sim_base`, or your own custom function.

Usage

```
run_ss3model(scenarios, iterations, type = c("om", "em"), admb_options = "",
  hess = FALSE, ignore.stdout = TRUE, admb_pause = 0.05,
  ss_mode = c("safe", "optimized"), show.output.on.console = FALSE, ...)
```

Arguments

scenarios	Which scenarios to run. Controls which folder contains the model that SS3 should run on.
iterations	Which iterations to run. Controls which folder contains the model that SS3 should run on.
type	Are you running the operating or estimation models?
admb_options	Any additional options to pass to the SS3 command.
hess	Calculate the Hessian on estimation model runs?
ignore.stdout	Passed to system. If TRUE then ADMB output is not printed on screen. This will be slightly faster. Set to FALSE to help with debugging.
admb_pause	A length of time (in seconds) to pause after running the simulation model. This can be necessary on certain computers where file writing can be slightly delayed. For example, on computers where the files are written over a network connection. If the output files haven't finished writing before R starts looking for the output then the simulation will crash with an error about missing files. The default value is set to 0.01 seconds, just to be safe.
ss_mode	Which version of the SS3 executable should be run? "safe" or "optimized"? Safe mode is useful for model building and testing. Optimized will be slightly faster for running simulations. Default is safe mode.
show.output.on.console	Logical: passed on to system.
...	Anything else to pass to system.

Details

ss3sim requires you to place the SS3 executable(s) in your path. See the vignette vignette("ss3sim-vignette") for details on this process. The executables themselves can be downloaded from: https://www.dropbox.com/sh/zg0sec6j20sfyyz/AACQiuk787qW882U2euVKoPna

There are two versions of the executables: safe and optimized (opt). Safe mode is best used during model development and testing. Optimized mode will be slightly faster and can be used for simulation if desired. The default is safe mode. **ss3sim** assumes that you at least have the safe-mode SS3 binary in your path. If you wish to use optimized mode then you must also have the opt executable version in your path. These executables must be named exactly as they are at the link above. You can choose the SS mode by setting the argument ss_mode to "safe" or "optimized".

Author(s)

Sean C. Anderson

See Also

ss3sim_base, run_ss3sim

`run_ss3sim` *Master function to run SS3 simulations*

Description

This is the main high-level wrapper function for running **ss3sim** simulations. This function first deals with parsing a scenario ID into case arguments, reads the appropriate case files, and then passes these arguments on to `ss3sim_base` to run a simulation. Alternatively, you might choose to run `ss3sim_base` directly and skip the case-file setup.

Usage

```
run_ss3sim(iterations, scenarios, case_folder, om_dir, em_dir,
  case_files = list(F = "F", D = c("index", "lcomp", "agecomp")),
  user_recdevs = NULL, parallel = FALSE, parallel_iterations = FALSE, ...)
```

Arguments

`iterations`	Which iterations to run. A numeric vector. For example `1:100`.
`scenarios`	Which scenarios to run. A vector of character objects. For example `c("D0-F0-cod", "D1-F1-cod")`. Also, see `expand_scenarios` for a shortcut to specifying the scenarios. See `get_caseargs` and the vignette for details on specifying the scenarios.
`case_folder`	The folder containing the plain-text case files.
`om_dir`	The folder containing the SS3 operating model configuration files.
`em_dir`	The folder containing the SS3 estimation model configuration files.
`case_files`	A named list that relates the case IDs to the files to return. **The default list specifies only the required fishing mortality and data scenarios. To specify other cases you will need to extend this named list**. This argument is passed to `get_caseargs`. See that function for details and examples of how to specify this. The introduction vignette also explains how to specify the case files.
`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.

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. See the example below.

parallel_iterations
: Logical. By default `parallel = TRUE` will run scenarios in parallel. If you set `parallel = TRUE` and `parallel_iterations = TRUE` then the iterations will be run in parallel. This would be useful if you were only running one scenario but you wanted to run it faster.

...
: Anything else to pass to `ss3sim_base`. This could include `bias_adjust` and `bias_nsim`. Also, you can pass additional options to the `SS3` command through the argument `admb_options`.

Details

The operating model folder should contain: `forecast.ss`, `yourmodel.ctl`, `yourmodel.dat`, `ss3.par`, and `starter.ss`. The files should be the versions that are returned from an SS run as `.ss_new` files. This is important because it creates consistent formatting which many of the functions in this package depend on. Rename the `.ss_new` files as listed above (and in all lowercase). The estimation model folder should contain all the same files listed above except the `ss3.par` and `yourmodel.dat` files, which are unnecessary but can be included if desired. See the vignette for details on modifying an existing `SS3` model to run with **ss3sim**. Alternatively, you might consider modifying one of the built-in model configurations.

Value

The output will appear in whatever your current R working directory is. There will be folders named after your scenarios. They will look like this:

- `D0-F0-cod/bias/1/om`
- `D0-F0-cod/bias/1/em`
- `D0-F0-cod/bias/2/om`
- ...
- `D0-F0-cod/1/om`
- `D0-F0-cod/1/em`
- `D0-F0-cod/2/om`
- ...

Author(s)

Sean C. Anderson

See Also

`ss3sim_base`, `run_ss3model`, `run_bias_ss3`, `get_caseargs`, `expand_scenarios`

Examples

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

# Find the data in the ss3sim package:
d <- system.file("extdata", package = "ss3sim")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
case_folder <- paste0(d, "/eg-cases")

# Without bias adjustment:
run_ss3sim(iterations = 1, scenarios = "D0-F0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em)
unlink("D0-F0-cod", recursive = TRUE) # clean up

# An example specifying the case files:
run_ss3sim(iterations = 1, scenarios = "D0-F0-E0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  case_files = list(F = "F", D = c("index", "lcomp",
      "agecomp"), E = "E"))
unlink("D0-F0-cod", recursive = TRUE) # clean up

# With bias adjustment:
# (Note that bias_nsim should be bigger, say 5 or 10, but it is set
# to 2 here so the example runs faster.)
run_ss3sim(iterations = 1, scenarios = "D1-F0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = TRUE, bias_nsim = 2)

# Restarting the previous run using the existing bias-adjustment
# output
run_ss3sim(iterations = 2:3, scenarios = "D1-F0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = FALSE, bias_already_run = TRUE)
unlink("D1-F0-cod", recursive = TRUE)

# A run with deterministic process error for model checking:
recdevs_det <- matrix(0, nrow = 100, ncol = 20)
run_ss3sim(iterations = 1:20, scenarios = "D0-E100-F0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = TRUE, bias_nsim = 2, user_recdevs = recdevs_det)
unlink("D0-E100-F0-cod", recursive = TRUE)

# An example of a run using parallel processing across 2 cores:
require(doParallel)
registerDoParallel(cores = 2)
require(foreach)
getDoParWorkers() # check how many cores are registered
```

```
# parallel scenarios:
run_ss3sim(iterations = 1, scenarios = c("D0-F0-cod",
    "D1-F0-cod"), case_folder = case_folder,
  om_dir = om, em_dir = em, parallel = TRUE)
unlink("D0-F0-cod", recursive = TRUE)
unlink("D1-F0-cod", recursive = TRUE)

# parallel iterations:
run_ss3sim(iterations = 1:2, scenarios = c("D0-F0-cod"),
  case_folder = case_folder, om_dir = om, em_dir = em,
  parallel = TRUE, parallel_iterations = TRUE)
unlink("D0-F0-cod", recursive = TRUE)

# parallel iterations with bias adjustment:
run_ss3sim(iterations = 1:2, scenarios = c("D0-F0-cod"),
  case_folder = case_folder, om_dir = om, em_dir = em, parallel = TRUE,
  parallel_iterations = TRUE, bias_adjust = TRUE, bias_nsim = 2)
unlink("D0-F0-cod", recursive = TRUE)

# Return to original working directory:
setwd(wd)

## End(Not run)
```

`sample_agecomp` *Sample age compositions from expected values*

Description

Take a `data.SS_new` file containing expected values and sample to create observed age compositions which are then written to file for use by the estimation model. If used with `run_ss3sim` the case file should be named `agecomp`. A suggested (default) case letter is `D` for data.

Usage

```
sample_agecomp(dat_list, outfile, fleets = c(1, 2), Nsamp, years, cpar = 1,
  ESS = NULL, write_file = TRUE, keep_conditional = TRUE)
```

Arguments

`dat_list`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`outfile`	A character string of the new file name to be created. Must end in `.dat` or equal `wtatage.ss`.
`fleets`	*A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from `outfile` (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to `fleets=NULL`.

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 indicates the same standard deviation as a multinomial of the given Nsamp, 2 indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion.

ESS An effective sample size to write to file (but not use in sampling). The default of NULL means to use the true (internally calculated) ESS, which is Nsamp for the multinomial case or given by the formula under cpar for the Dirichlet case. Must match the structure of the years arguments.

write_file A logical switch for whether to write outfile to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.

keep_conditional
A logical if conditional age-at-length data should be kept or removed entirely from the .dat 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.

Value

A modified .dat file if write_file=TRUE. A list object containing the modified .dat file is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan and Kotaro Ono; modified from a version by Roberto Licandeo and Felipe Hurtado-Ferro

See Also

Other sampling functions: clean_data, sample_calcomp, sample_index, sample_lcomp, sample_mlacomp, sample_wtatage

Examples

```
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/models/cod-om/codOM.dat")
dat_list <- r4ss::SS_readdat(f_in, verbose = FALSE)
dat_list <- change_fltname(dat_list)

## Turn off age comps by specifying fleets=NULL
sample_agecomp(dat_list=dat_list, outfile="test1.dat",
               fleets=NULL, cpar=c(5,NA), Nsamp=list(100,100),
               years=list(1995, 1995), write_file=FALSE)

## Generate with a smaller number of fleet taking samples
ex1 <- sample_agecomp(dat_list=dat_list, outfile="test1.dat", fleets=c(2),
                      Nsamp=list(c(10,50)), years=list(c(26,27)),
                      write_file=FALSE)

## Generate with varying Nsamp by year for first fleet
ex2 <- sample_agecomp(dat_list=dat_list, outfile="test2.dat", fleets=c(1,2),
                      Nsamp=list(c(rep(50, 5), rep(100, 5)), 50),
                      years=list(seq(26, 44, 2),
                          c(26:100)), write_file=FALSE)

## Run three  cases showing Multinomial, Dirichlet(1) and over-dispersed
## Dirichlet for different levels of sample sizes
op <- par(mfrow = c(1,3))
set.seed(1)
for(samplesize in c(30, 100, 1000)){
    ex4 <- sample_agecomp(dat_list=dat_list, outfile="test4.dat", fleets=c(1,2),
                          Nsamp=list(samplesize, samplesize),
                          write_file = FALSE,
                          years=list(50,50), cpar=c(NA, 1))
    ex5 <- sample_agecomp(dat_list=dat_list, outfile="test5.dat", fleets=c(1,2),
                          Nsamp=list(samplesize, samplesize),
                          write_file = FALSE,
                          years=list(50,50), cpar=c(1, 1))
    ex6 <- sample_agecomp(dat_list=dat_list, outfile="test6.dat", fleets=c(1,2),
                          Nsamp=list(samplesize, samplesize),
                          write_file = FALSE,
                          years=list(50,50), cpar=c(5, 1))
    true <- subset(dat_list$agecomp, FltSvy==1 & Yr == 50)[-(1:9)]
    true <- true/sum(true)
    mult <- subset(ex4$agecomp, FltSvy==1)[1,-(1:9)]
    mult <- mult/sum(mult)
    plot(1:15, mult, type="b", ylim=c(0,1),
         col=1, xlab="Age", ylab="Proportion", main=paste("Sample size=",
         samplesize))
    legend("topright", legend=c("Multinomial", "Dirichlet(1)",
                                "Dirichlet(5)", "Truth"),
           lty=1, col=1:4)
    lines((1:15), subset(ex5$agecomp, FltSvy==1)[1,-(1:9)], type="b", col=2)
    lines((1:15), subset(ex6$agecomp, FltSvy==1)[1,-(1:9)], type="b", col=3)
    lines((1:15), true, col=4, lwd=2)
```

```
}
par(op)
```

`sample_calcomp` *Sample conditional age-at-length (CAL) data and write to file for use by the EM.*

Description

Sample conditional age-at-length (CAL) data and write to file for use by the EM.

Usage

```
sample_calcomp(dat_list, outfile, fleets = c(1, 2), years,
  write_file = TRUE, Nsamp)
```

Arguments

`dat_list`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`outfile`	A character string of the new file name to be created. Must end in `.dat` or equal `wtatage.ss`.
`fleets`	*A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from `outfile` (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to `fleets=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`.
`write_file`	A logical switch for whether to write `outfile` to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.
`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`.

Details

Take a `data.SS_new` file containing expected values and sample from true lengths, using length comp sample sizes, to get realistic sample sizes for age bins given a length. Only the multinomial distribution is currently implemented. xIf no fish are sampled then that row is discarded. A value of NULL for fleets indicates to delete the data so the EM If used with `run_ss3sim` the case file should be named `calcomp`.

Value

A modified `.dat` file if `write_file=TRUE`. A list object containing the modified `.dat` file is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to `run_ss3sim`. If one of these arguments is not specified in a case file, then a value of `NULL` will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Note

This function is only reliable when using multinomial length compositions for the matching fleet. The real-valued length compositions resulting from the Dirichlet distribution cause difficulties in the sampling code. See the vignette for more.

Author(s)

Cole Monnahan, Kotaro Ono

See Also

Other sampling functions: `clean_data`, `sample_agecomp`, `sample_index`, `sample_lcomp`, `sample_mlacomp`, `sample_wtatage`

`sample_index` *Sample the biomass with observation error*

Description

This function creates an index of abundance sampled from the expected available biomass for given fleets in given years. Let B_y be the biomass from the operating model for year y. Then the sampled value is calculated as: B_y*exp(rnorm(1, 0, `sds_obs`)-`sds_obs`^2/2). The second term adjusts the random samples so that their expected value is B_y (i.e. the log-normal bias correction). If used with `run_ss3sim` the case file should be named `index`. A suggested (default) case letter is `D` for data.

Usage

```
sample_index(dat_list, outfile, fleets, years, sds_obs, make_plot = FALSE,
  write_file = TRUE)
```

Arguments

dat_list	An SS data list object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.
outfile	A character string of the new file name to be created. Must end in .dat or equal wtatage.ss.
fleets	*A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from outfile (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to fleets=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.
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.
make_plot	A logical switch for whether to make a crude plot showing the results. Useful for testing and exploring the function.
write_file	A logical switch for whether to write outfile to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.

Value

A modified .dat file if write_file=TRUE. A list object containing the modified .dat file is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan, Kotaro Ono

See Also

Other sampling functions: clean_data, sample_agecomp, sample_calcomp, sample_lcomp, sample_mlacomp, sample_wtatage

Examples

```
## Not run:
# Find the example data location:
d <- system.file("extdata", package = "ss3sim")
```

```
f_in <- paste0(d, "/example-om/data.ss_new")
dat_list <- r4ss::SS_readdat(f_in, section = 2, verbose = FALSE)
dat_list <- change_fltname(dat_list)
outfile <- "test.dat"
ex1 <- sample_index(dat_list, outfile, fleets=c(2,3),
                    years=list(1938:2012, 1938:2012) ,
                    sds_obs=list(1e-6, 1e-6), write_file=FALSE,
                    make_plot = TRUE)
ex2 <- sample_index(dat_list, outfile, fleets=c(2,3),
                    years=list(1938:2012, 1938:2012) ,
                    sds_obs=list(.05, .05), write_file=FALSE,
                    make_plot = TRUE)
library(ggplot2)
ggplot(ex1, aes(x=year, y=obs, group=index, ymin=0,
                colour=as.factor(index)))+geom_line() + geom_point(data=ex2,
                aes(x=year, y=obs, colour=as.factor(index), group=index))
## Exclude a fleet and have varying sds_obs by year
ex3 <- sample_index(dat_list, outfile, fleets=c(2,NA),
                    years=list(1938:2012, 1950),
                    sds_obs=list(seq(.001, .1, len=75), .1),
                    write_file=FALSE)
ggplot(ex3, aes(x=year, y=obs, group=index, ymin=0,
                colour=as.factor(index)))+geom_point()

## End(Not run)
```

`sample_lcomp` *Sample length compositions from expected values*

Description

Take a `data.SS_new` file containing expected values and sample to create observed length compositions which are then written to file for use by the estimation model. If used with `run_ss3sim` the case file should be named `lcomp`. A suggested (default) case letter is `D` for data.

Usage

```
sample_lcomp(dat_list, outfile, fleets = c(1, 2), Nsamp, years, cpar = 1,
  ESS = NULL, write_file = TRUE)
```

Arguments

`dat_list`	An SS data list object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`outfile`	A character string of the new file name to be created. Must end in `.dat` or equal `wtatage.ss`.
`fleets`	*A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from `outfile` (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to `fleets=NULL`.

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 indicates the same standard deviation as a multinomial of the given Nsamp, 2 indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion.
ESS	An effective sample size to write to file (but not use in sampling). The default of NULL means to use the true (internally calculated) ESS, which is Nsamp for the multinomial case or given by the formula under cpar for the Dirichlet case. Must match the structure of the years arguments.
write_file	A logical switch for whether to write outfile to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.

Value

A modified .dat file if write_file=TRUE. A list object containing the modified .dat file is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan and Kotaro Ono; modified from a version by Roberto Licandeo and Felipe Hurtado-Ferro

See Also

Other sampling functions: clean_data, sample_agecomp, sample_calcomp, sample_index, sample_mlacomp, sample_wtatage

Examples

```
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/models/cod-om/codOM.dat")
dat_list <- r4ss::SS_readdat(f_in, verbose = FALSE)
dat_list <- change_fltname(dat_list)

## Generate with constant sample size across years
ex1 <- sample_lcomp(dat_list=dat_list, outfile="test1.dat", fleets=c(1,2),
```

```
                    Nsamp=list(100,50), years=list(seq(26, 100, by=2),
                                          80:100), write_file = FALSE)

## Generate with varying Nsamp by year for first fleet
ex2 <- sample_lcomp(dat_list=dat_list, outfile="test2.dat", fleets=c(1,2),
                    Nsamp=list(c(rep(50, 5), rep(100, 5)), 50),
                    years=list(seq(26, 44, by=2),
                        80:100), write_file = FALSE)

## Not run:
## Plot distributions for a particular year to compare multinomial
## vs. overdispersed Dirichlet
temp.list <- temp.list2 <- list()
for(i in 1:40){
    temp.list[[i]] <-
      sample_lcomp(dat_list=dat_list, outfile="test1.dat", fleets=c(2), cpar=c(3),
                    Nsamp=list(100), years=list(1995),
                    write_file=FALSE)
    temp.list2[[i]] <-
        sample_lcomp(dat_list=dat_list, outfile="test1.dat", fleets=c(2),
                    cpar=c(NA), Nsamp=list(100), years=list(1995),
                    write_file=FALSE)
}
## Organize the data for plotting
x1 <- reshape2::melt(do.call(rbind, temp.list)[,-(1:6)[-3]], id.vars="FltSvy")
x2 <- reshape2::melt(do.call(rbind, temp.list2)[,-(1:6)[-3]], id.vars="FltSvy")
op <- par(mfrow=c(2,1))
with(x1, boxplot(value~variable, las=2, ylim=c(0,.6), ylab="Proportion",
                 main="Overdispersed (cpar=3)",  xlab="length bin"))
temp <- as.numeric(subset(dat_list$lencomp, Yr==1995 & FltSvy == 2)[-(1:6)])
points(temp/sum(temp), pch="-", col="red")
with(x2, boxplot(value~variable, las=2, ylim=c(0,.6), ylab="Proportion",
                 main="Multinomial", xlab="length bin"))
temp <- as.numeric(subset(dat_list$lencomp, Yr==1995 & FltSvy == 2)[-(1:6)])
points(temp/sum(temp), pch="-", col="red")
par(op)

## End(Not run)
```

`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,
  write_file = TRUE, mean_outfile = NULL, verbose = TRUE)
```

Arguments

dat_list	An SS data list object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.
outfile	A character string of the new file name to be created. Must end in .dat or equal wtatage.ss.
ctl_file_in	A path to the control file, outputed 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 numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from outfile (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to fleets=NULL.
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.
write_file	A logical switch for whether to write outfile to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.
mean_outfile	A path to write length and age data for external estimation of parametric growth. If NULL no file will be written. This file is used by change_e to externally estimate growth parameters. Filename must contain "vbgf" to be used by change_e. 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.SS_new file, read in by **r4ss** function 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 write_file=TRUE. A list object containing the modified .dat file is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan, Kelli Johnson

See Also

Other sampling functions: clean_data, sample_agecomp, sample_calcomp, sample_index, sample_lcomp, sample_wtatage

Examples

```
temp_path <- file.path(tempdir(), "ss3sim-test")
dir.create(temp_path, showWarnings = FALSE)
wd <- getwd()
setwd(temp_path)
d <- system.file("extdata/models/cod-om", package = "ss3sim")
dat_in <- file.path(d, "codOM.dat")
dat_list <- r4ss::SS_readdat(dat_in, verbose = FALSE)
dat_list <- change_fltname(dat_list)
dat_list <- change_data(dat_list, outfile = NULL, write_file = FALSE,
  fleets = 1, years = 1990:2010, types = c("age", "mla"))
dat_list <- change_fltname(dat_list)
ctl_file_in <- file.path(d, "codOM.ctl")

out <- sample_mlacomp(dat_list, outfile = NULL, ctl_file_in = ctl_file_in,
                      fleets = 1, Nsamp = 30, years = list(1992),
                      verbose = FALSE, mean_outfile = "test.csv", write_file = FALSE)

setwd("..")
unlink("ss3sim-test", recursive = TRUE)
setwd(wd)
```

---

sample_wtatage *Sample empirial weight-at-age data and write to file for use by the EM*

Description

Take a data.SS_new file containing expected values and sample from true 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 converted to weight and averaged to get mean weight-at-age values. Missing ages and years are filled according to a specified function. These matrices are then written to file for the EM. By calling this function, **ss3sim** will turn on the empirical weight-at-age function (set maturity option to 5) automatically. See ss3sim_base for more details on how that is implemented. If used with run_ss3sim the case file should be named wtatage.

Usage

```
sample_wtatage(wta_file_in, outfile, dat_list, ctl_file_in, years,
  fill_fnc = fill_across, write_file = TRUE, 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 of the new file name to be created. Must end in .dat or equal wtatage.ss.
dat_list	An SS data list object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.
ctl_file_in	A path to the control file, outputed 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.
write_file	A logical switch for whether to write outfile to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.
fleets	*A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from outfile (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to fleets=NULL.
cv_wtatage	A user specified CV for growth. Default is NULL.

Value

A modified .wtatage.ss file if write_file = TRUE. A list object containing the modified .wtatage.ss file is returned invisibly.

Which arguments to specifiy in case files

All function argument descriptions that start with an asterisk (*) will be passed through the case files to run_ss3sim. If one of these arguments is not specified in a case file, then a value of NULL will be passed, which may or may not be an appropriate value. Other arguments will be ignored if specified.

Author(s)

Cole Monnahan, Allan Hicks, Peter Kuriyama

See Also

`fill_across`

Other sampling functions: `clean_data`, `sample_agecomp`, `sample_calcomp`, `sample_index`, `sample_lcomp`, `sample_mlacomp`

`sanitize_admb_options` *Check admb options to make sure there aren't flags there shouldn't be*

Description

Check admb options to make sure there aren't flags there shouldn't be

Usage

```
sanitize_admb_options(x, exclude = "-nohess")
```

Arguments

`x`	The admb options
`exclude`	A character object (not a vector)

Author(s)

Sean C. Anderson

`scalar_dat` *Example scalar data from the ss3sim vignette*

Description

Example scalar data from the ss3sim vignette

`setup_parallel` *Setup parallel processing*

Description

Setup parallel processing

Usage

```
setup_parallel()
```

ss3sim *ss3sim: Fisheries stock assessment simulation testing with Stock Synthesis*

Description

The **ss3sim** R package is designed to facilitate rapid, reproducible, and flexible simulation with the widely-used Stock Synthesis 3 (SS3) statistical catch-at-age stock assessment framework.

Details

An **ss3sim** simulation requires three types of input: (1) a base model of the underlying truth (an SS3 operating model), (2) a base model of how you will assess that truth (an SS3 estimation model), (3) and a set of cases that deviate from these base models that you want to compare (configuration arguments provided as plain-text cases files).

You can find examples of these SS3 operating and estimation models within the package data (`inst/extdata/models/`). The package data also contains example plain-text control files in the folder `inst/extdata/cases` and `inst/extdata/eg-cases`.

To carry out **ss3sim** simulations with the version from CRAN, you will need to have SS3 installed on your computer and the binary needs to be in the path that R sees. See the section "Installing the ss3sim R package" in the vignette `vignette("ss3sim-vignette")` for instructions on installing SS3. See the Appendix A "Putting SS3 in your path" in the vignette for instructions on making sure SS3 will work from within R.

The main **ss3sim** functions are divided into three types:

1. `change` and `sample` functions that manipulate SS3 configuration files. These manipulations generate an underlying "truth" (operating models) and control our assessment of those models (estimation models).

- `change_f`: Controls fishing mortality.
- `change_tv`: Adds time-varying features. For example, time-varying natural mortality, growth, or selectivity.
- `sample_lcomp`: Controls how length composition data are sampled.
- `sample_agecomp`: Controls how age composition data are sampled.
- `sample_index`: Controls how the fishery and survey indices are sampled.
- `change_e`: Controls which and how parameters are estimated.
- `change_retro`: Controls the number of years to discard for a retrospective analysis.
- `change_rec_devs`: Substitutes recruitment deviations.
- `change_lcomp_constant`: Set the robustification constant for length composition data.
- `change_tail_compression`: Replace tail compression value for length composition data.

2. `run` functions that conduct simulations. These functions generate a folder structure, call manipulation functions, run SS3 as needed, and save the output.

- `run_ss3sim`: Main function to run **ss3sim** simulations.

- ss3sim_base: Underlying base simulation function. Can also be called directly.

3. get functions for synthesizing the output.

- get_results_scenario: Extract the results for a single scenario.
- get_results_all: Extract results from a series of scenarios.

See the introductory vignette vignette("introduction",package = "ss3sim") for more extensive explanation of how to use the **ss3sim** R package.

ss3sim was developed by graduate students and post doctoral researchers at the University of Washington (School of Aquatic and Fishery Sciences and Quantitative Ecology and Resource Management departments) and Simon Fraser University. The authors of individual functions are listed within the function documentation and all contributors are listed in the DESCRIPTION file.

If you use **ss3sim** in a publication, please cite the package as indicated by running citation("ss3sim") in the R console.

---

ss3sim_base *Base wrapper function to run an ss3sim simulation*

Description

This function is a wrapper function that can call run_ss3model for the operating model, sample the output (add recruitment deviations, survey the data, etc.), and run the estimation model. ss3sim_base is the main internal function for **ss3sim**. It is intended to be used through run_ss3sim, but can also be used directly.

Usage

```
ss3sim_base(iterations, scenarios, f_params, index_params, lcomp_params,
  agecomp_params, calcomp_params = NULL, wtatage_params = NULL,
  mlacomp_params = NULL, em_binning_params = NULL, estim_params = NULL,
  tv_params = NULL, om_dir, em_dir, retro_params = NULL,
  data_params = NULL, call_change_data = TRUE, user_recdevs = NULL,
  user_recdevs_warn = TRUE, bias_adjust = FALSE, bias_nsim = 5,
  bias_already_run = FALSE, hess_always = FALSE, print_logfile = TRUE,
  sleep = 0, conv_crit = 0.2, seed = 21, keep_compreport = TRUE, ...)
```

Arguments

iterations	Which iterations to run. A numeric vector.
scenarios	Which scenarios to run.
f_params	A named list containing arguments for change_f. A mandatory case.
index_params	A named list containing arguments for sample_index. A mandatory case.
lcomp_params	A named list containing arguments for sample_lcomp. A mandatory case.
agecomp_params	A named list containing arguments for sample_agecomp. A mandatory case.

`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).

`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.

`retro_params` A named list containing arguments for `change_retro`.

`data_params` A named list containing arguments for `change_data`.

`call_change_data`
A boolean of whether to call `change_data` and modify the OM at each iteration. Defaults to `TRUE`. See the vignette for further information on why you might choose to turn this off.

`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` Run bias adjustment first? See `run_bias_ss3`.

`bias_nsim` If bias adjustment is run, how many simulations should the bias adjustment factor be estimated from? It will take the mean of the adjustment factors across these runs.

`bias_already_run`
If you've already run the bias runs for a scenario (the bias folders and `.dat` files already exist) then you can set this to `TRUE` to avoid re-running the bias adjustment routine.

`hess_always` If `TRUE` then the Hessian will always be calculated. If `FALSE` then the Hessian will only be calculated for bias-adjustment runs thereby saving time.

`print_logfile` Logical. Print a log file?

`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.

`conv_crit` The maximum percentage of bias iterations that can produce a non-invertible Hessian before a warning will be produced. If this percentage is exceeded then a file `WARNINGS.txt` will be produced. Currently, the simulations will continue to run.

`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.

`keep_compreport` Logical: should the SS3 file `CompReport.sso` be kept or deleted? `CompReport.sso` is often rather large and so deleting it can save space but the file is needed for some of the **r4ss** plots among other purposes.

`...` Anything extra to pass to `run_ss3model`. For example, you may want to pass additional options to `SS3` through the argument `admb_options`. Anything that doesn't match a named argument in `run_ss3model` will be passed to the `system` call that runs `SS3`. Also, see the argument `ss_mode` to choose between safe or optimized SS3 executables (default is safe mode).

Details

This function is written to be flexible. You can specify the fishing mortality, survey index, length composition, age composition, and time-varying parameters 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. They will look like this:

- `D0-F0-cod/bias/1/om`
- `D0-F0-cod/bias/1/em`
- `D0-F0-cod/bias/2/om`
- ...
- `D0-F0-cod/1/om`
- `D0-F0-cod/1/em`
- `D0-F0-cod/2/om`
- ...

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:
temp_path <- file.path(tempdir(), "ss3sim-base-example")
dir.create(temp_path, showWarnings = FALSE)
wd <- getwd()
setwd(temp_path)

# Find the data in the ss3sim package:
d <- system.file("extdata", package = "ss3sim")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
case_folder <- paste0(d, "/eg-cases")

# Pull in file paths from the package example data:
d <- system.file("extdata", package = "ss3sim")
om_dir <- paste0(d, "/models/cod-om")
em_dir <- paste0(d, "/models/cod-em")
a <- get_caseargs(folder = paste0(d, "/eg-cases"), scenario =
"F0-D0-M0-E0-cod")

ss3sim_base(iterations = 1, scenarios = "M0-F0-D0-E0-cod",
  f_params = a$F, index_params = a$index, lcomp_params = a$lcomp,
  agecomp_params = a$agecomp, tv_params = a$tv_params, estim_params = a$E,
  om_dir = om_dir, em_dir = em_dir)
unlink("M0-F0-D0-E0-cod", recursive = TRUE) # clean up

# Or, create the argument lists directly in R and skip the case file setup:

F0 <- list(years = 1913:2012, years_alter = 1913:2012, fvals = c(rep(0,
  25), rep(0.114, 75)))

index1 <- list(fleets = 2, years = list(seq(1974, 2012, by = 2)), sds_obs =
  list(0.1))

lcomp1 <- list(fleets = c(1, 2), Nsamp = list(100, 100), years =
  list(1938:2012, seq(1974, 2012, by = 2)), lengthbin_vector = NULL, cpar =
  c(1, 1))

agecomp1 <- list(fleets = c(1, 2), Nsamp = list(100, 100), years =
  list(1938:2012, seq(1974, 2012, by = 2)), agebin_vector = NULL, cpar =
  c(1, 1))

E0 <- list(natM_type = "1Parm", natM_n_breakpoints = NULL, natM_lorenzen =
  NULL, natM_val = c(NA,-1), par_name = "LnQ_base_3_CPUE", par_int = NA,
  par_phase = -1, forecast_num = 0)

M0 <- list(NatM_p_1_Fem_GP_1 = rep(0, 100))

ss3sim_base(iterations = 1, scenarios = "D1-E0-F0-M0-cod",
  f_params = F0, index_params = index1, lcomp_params = lcomp1,
  agecomp_params = agecomp1, estim_params = E0, tv_params = M0,
```

```
  om_dir = om, em_dir = em)

unlink("D1-E0-F0-M0-cod", recursive = TRUE) # clean up

setwd(wd)

## End(Not run)
```

`standardize_bounds` *Standardize the bounds of the estimation model control file.*

Description

Function to standardize the bounds of the control file in the estimation model. This function first checks to ensure the initial values in the estimation model control file are set to the true values of the `om_ctl_file` and if not sets them for every parameter. Next, the function adjusts the LO and HI values in the `em_ctl_file` to be a fixed percentage of the initial value for every parameter.

Usage

```
standardize_bounds(percent_df, dir, em_ctl_file, om_ctl_file = "",
  verbose = FALSE, estimate = NULL, ...)
```

Arguments

`percent_df`	A `data.frame` with nine rows and three columns. The first column is the parameter. The second column is the percent of the initial parameter value LO is set to. The third column is the percent of the initial parameter value HI is set to.
`dir`	A path to the directory containing the model files.
`em_ctl_file`	A string with the name of the estimation model control file. `em_ctl_file` must be located in `dir`.
`om_ctl_file`	A string with the name of the operating model control file. If it is not given the part of the function which matches the OM and EM INIT values is ignored. Default is `""`. `om_ctl_file` must be located in `dir`.
`verbose`	Detailed output to command line. Default is `FALSE`.
`estimate`	A logical for which changed parameters are to be estimated. Used by `SS_changepars`, where in **r4ss** the default is `FALSE`, which turns all parameter estimation off. Here the default is `NULL`, which will leave parameter phases unchanged.
`...`	Any other arguments to pass to `SS_changepars`.

Author(s)

Christine Stawitz

Examples

```
## Not run:
temp_path <- file.path(tempdir(), "standardize-bounds-example")
dir.create(temp_path, showWarnings = FALSE)
wd <- getwd()
setwd(temp_path)

## Set to the path and filename of the OM and EM control files
OM.ctl <- system.file("extdata", "models", "cod-om", "codOM.ctl",
  package = "ss3sim")
EM.ctl <- system.file("extdata", "models", "cod-em", "codEM.ctl",
  package = "ss3sim")
file.copy(OM.ctl, "om.ctl")
file.copy(EM.ctl, "em.ctl")

## Use SS_parlines to get the proper names for parameters for the data frame
om.pars <- r4ss::SS_parlines(ctlfile="om.ctl")
em.pars <- r4ss::SS_parlines(ctlfile="em.ctl")

## Set percentages to make lower and upper bounds
lo.percent<-rep(.5,11)
hi.percent<-c(500,1000,1000,rep(500,8))

##Populate data frame using EM parameter names and percentages
percent_df<-data.frame(Label=as.character(em.pars[c(1:6,17,27:30),"Label"]),
  lo=lo.percent,hi=hi.percent)

##Run function
standardize_bounds(percent_df = percent_df, dir = temp_path, em_ctl_file = "em.ctl",
                   om_ctl_file = "om.ctl")
unlink(temp_path, recursive = TRUE)

setwd(wd)

## End(Not run)
```

---

substr_r *Substring from right*

Description

Substring from right

Usage

```
substr_r(x, n)
```

Arguments

x	A character object
n	The number of characters from the right to extract

References

http://stackoverflow.com/questions/7963898/extracting-the-last-n-characters-from-a-string-in-r

---

ts_dat *Example time series data from the ss3sim vignette*

Description

Example time series data from the ss3sim vignette

---

vbgf_func *Predict length given VBGF parameters*

Description

External estimation procedure for von Bertalanffy growth.

Usage

```
vbgf_func(L1, L.inf, k, ages, a3)
```

Arguments

L1	mean length at youngest age which is well sampled in the data (a3)
L.inf	Length at infinity
k	von bertalanffy growth rate parameter
ages	vector of ages in the data for which you want to predict mean length-at-age
a3	youngest age which is well sampled in the data

Value

a vector of lengths predicted which correspond to the input ages vector.

verify_input *Verify and standardize SS3 input files*

Description

This function verifies the contents of operating model (`om`) and estimation model (`em`) folders. 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.

Details

This is a helper function to be used within the larger wrapper simulation functions.

Value

Returns a version of the folder with sanitized files or an error if some files are missing.

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", package = "ss3sim")

om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/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 = paste0(temp_path, "/cod-om"), type = "om")
verify_input(model_dir = paste0(temp_path, "/cod-em"), type = "em")
unlink(temp_path, recursive = TRUE)
```

`verify_plot_arguments` *A helper function to check the correct input for the plotting functions.*

Description

A helper function to check the correct input for the plotting functions.

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. Column should be a factor (e.g. "F" or "species").
`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.
`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.
`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 should be interpreted as relative error. If `TRUE`, `ylim` is set to `c(-1,1)`, the y axis label is changed automatically, and a red line at y=0 is added.
`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** simulation scalar and timeseries (ts) objects. They are designed to quickly explore simulation output, rather than publication-level figures. The functions use the `aes_string` function within **ggplot2** such that arguments are passed as characters that refer to columns of `data`.

Note that there are some subtle differences between the functions. Scalar plots require a value for `x`, while for ts plots `x` is invalid because it is fixed internally as 'year', since it makes no sense to

use another column. Boxplots cannot have a color mapped to them like points or lines, and thus `color` is not a valid argument. The ts point and line plots are grouped internally by 'ID', which is a combination of scenario and replicate.

Value

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

Output

These functions print the `ggplot` object, but also return it invisibly for saving or printing again later.

Author(s)

Cole Monnahan

Index