

Package ‘ss3sim’

May 9, 2026

Type Package

Title Fisheries Stock Assessment Simulation Testing with Stock Synthesis

Version 1.0.3

Description Develops a framework for fisheries stock assessment simulation testing with Stock Synthesis (SS) as described in Anderson et al. (2014) <[doi:10.1371/journal.pone.0092725](https://doi.org/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, rmarkdown

VignetteBuilder knitr

Depends R (>= 3.3)

Imports foreach, r4ss (>= 1.35.0), gtools, ggplot2, bbmle, grDevices, graphics, stats, utils

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Author Kelli F. Johnson [aut, cre],
Sean C. Anderson [aut] (ORCID: <<https://orcid.org/0000-0001-9563-1937>>),
Kathryn Doering [aut],
Cole Monnahan [aut],
Christine Stawitz [aut],
Ian Taylor [aut],
Curry Cunningham [ctb],
Allan Hicks [ctb],
Felipe Hurtado-Ferro [ctb],
Peter Kuriyama [ctb],
Roberto Licandeo [ctb],
Carey McGilliard [ctb],

Melissa Murdian [ctb],
 Kotaro Ono [ctb],
 Merrill Rudd [ctb],
 Cody Szuwalski [ctb],
 Juan Valero [ctb],
 Athol Whitten [ctb]

Maintainer Kelli F. Johnson <kelli.johnson@noaa.gov>

Repository CRAN

Date/Publication 2019-11-08 22:40:02 UTC

Contents

add_colnames	4
add_CPUE	5
add_nulls	5
add_tv_parlines	6
calculate_data_units	6
calculate_re	7
case_comp	8
case_deparse	9
case_fishing	10
case_index	10
case_tv	11
change_data	12
change_e	15
change_em_binning	17
change_e_fcast_yrs	18
change_f	19
change_f_par	21
change_lcomp_constant	22
change_o	23
change_pop_bin	24
change_rec_devs	25
change_rec_devs_par	26
change_retro	27
change_tail_compression	29
change_tv	30
check_data	32
check_data_str_range	32
cleanup_ss3	33
clean_data	33
copy_ss3models	34
create_argfiles	35
expand_scenarios	37
facet_form	38
fill_across	38
get_args	39

get_bin	39
get_bin_info	40
get_caseargs	40
get_caseval	42
get_fish600_casefolder	42
get_model_folder	43
get_nll_components	43
get_recdevs	44
get_results_all	44
get_results_derived	45
get_results_scalar	46
get_results_scenario	46
get_results_timeseries	47
get_sigmar	48
get_ss_ver_dl	48
get_ss_ver_file	49
id_scenarios	49
plot_scalar_boxplot	50
plot_scalar_points	51
plot_ts_boxplot	52
plot_ts_lines	54
plot_ts_points	55
profile_fmsh	56
remove_CPUE	58
remove_q_ctl	58
rename_ss3_files	59
run_ss3model	60
run_ss3sim	61
sample_agecomp	64
sample_calcomp	66
sample_comp	68
sample_index	69
sample_lcomp	71
sample_mlacomp	73
sample_wtatage	75
sanitize_admb_options	76
scalar_dat	76
setup_parallel	77
ss3sim	77
ss3sim_base	78
standardize_bounds	82
substr_r	83
ts_dat	84
vbgf_func	84
verify_input	85
verify_plot_arguments	86

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

Description

Add missing columns to each data frame in the list allowing for the use `rbind` to create a single data frame. The code is based on `rbind.fill`, but we didn't want to depend on that package for just one function given it had not been updated since 2016.

Usage

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

Arguments

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

Value

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

Author(s)

Kelli Faye Johnson

Examples

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

add_CPUE	<i>Add a q setup line into an SS control file</i>
----------	---

Description

This function adds a q setup line to an SS 3.30 control file

Usage

```
add_CPUE(ctl.in, ctl.out = NULL, overwrite = FALSE,
         q = data.frame(fleet = 3, link = 1, link_info = 0, extra_se = 0,
                       biasadj = 0, float = 0, LO = -20, HI = 20, INIT = 0, PRIOR = 0, PR_SD =
                       99, PR_type = 0, PHASE = 1, env_var = 0, use_dev = 0, dev_mnyr = 0,
                       dev_mxyr = 0, dev_PH = 0, Block = 0, Blk_Fxn = 0, name = NULL))
```

Arguments

ctl.in	An SS control file name to read in.
ctl.out	The SS control file to read out.
overwrite	Logical. Overwrite an existing file with the same name as ctl.out?
q	a dataframe containing the q parameter lines to add.

Value

A modified SS control file.

Author(s)

Kelli Johnson

add_nulls	<i>Add NULL values to non-existent list elements</i>
-----------	--

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

<code>add_tv_parlines</code>	<i>Add short time varying parameter lines. At time of writing, this method will work for MG, selectivity, and catchability time varying, but not for SR</i>
------------------------------	---

Description

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

Usage

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

Arguments

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

Value

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

<code>calculate_data_units</code>	<i>Given sampling arguments, calculate super set of fleets, years, and data types.</i>
-----------------------------------	--

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`.
`mlcomp_params` Named lists containing the arguments for `sample_mlcomp`.
`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

```
## 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))
```

`calculate_re`*Calculate relative error*

Description

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

Usage

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

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 <code>_em</code> and <code>_om</code> as names.
add	Logical: should the relative error columns be added to <code>dat</code> or should the original EM and OM columns be dropped? If <code>FALSE</code> 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 <code>dat</code> if you wished. The default is to return all columns.

Value

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

Author(s)

Sean Anderson and Cole Monnahan

See Also

[get_results_all](#), [link{get_results_scenario}](#)

Examples

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

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 <code>length(fleets)</code> , where each element of the list contains a vector of sample sizes for each year for that given fleet.
years	A list of length <code>length(fleets)</code> , 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., <code>c("agecomp", "lcomp")</code>) if you want the case files to be the same for length and age compositions.
case	The casenumber you want to write to. If <code>case = 1</code> and <code>type = "agecomp"</code> , 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	<i>Turn an argument describing an object into a character.</i>
--------------	--

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	<i>Write a case file for fishing data to the disk.</i>
--------------	--

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:fishery (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 ss.par file, where $\text{length}(fvals) == \text{length}(years_alter)$ 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	<i>Write a case file for index data to the disk.</i>
------------	--

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 <code>length(fleets)</code> , 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 <code>case = 1</code> , 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	<i>Write time varying casefiles to the disk</i>
---------	---

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 <code>file.path(dir_models, species, "om", "ss3.ct1")</code> leads to valid <code>ss3.ct1</code> operating model files.
nyears	The length time-series included in the model. The length of <code>perc_change</code> must equal <code>nyears</code> .
verbose	Useful for debugging to print output to screen. Default is <code>FALSE</code> .

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"))
filenames <- dir(file.path(temp_path, "om"), full.names = TRUE)
ig <- file.rename(filenames, gsub("codOM\\.|ss\\.\"", "ss3.", filenames))

verify_input(file.path(temp_path, "om"), type = "om")
ig <- file.rename(file.path(temp_path, "om", "om.ct1"),
  file.path(temp_path, "om", "ss3.ct1"))

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_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 sexes 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 = NULL, 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, nsex = 1)
```

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 specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.
fleets	A numeric vector of fleets
years	A numeric vector of years
types	A vector that can take combinations of the following entries: "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. For conditional age-at-length (CAAL) data, the last value provided to len_bins will be used for Lbin_lo and -1 will be used for Lbin_hi for the largest length bin category, i.e., row of CAAL data.
pop_binwidth	*Population length bin width. Note that this value must be smaller than the bin width specified in length composition data len_bins or SS will fail (see notes in the SS 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 SS 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 SS. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the SS manual for further information. A NULL value indicates no action, a negative value indicates to SS to ignore it (not use that feature).
nsex	An integer value of 1 or 2 specifying the number of sexes in the model. If 1, then females are the only included sex. This information can be found in the data file for a given model and dictates how the composition data are structured.

Details

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

Value

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

Which arguments to specify 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_par](#), [change_f](#), [change_o](#), [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(file.path(d, "models", "cod-om", "codOM.dat"),
  version = NULL, verbose = FALSE)

# Basic test with just length data, default bins:
out <- change_data(file_in, outfile = NULL, types = "len",
  years = years, fleets = fleets)
print(out$lbin_vector)
print(out$lcomp)

# Change the length bins:
out <- change_data(file_in, outfile = NULL, types = "len",
  years = years, fleets = fleets, len_bins = 3:6)
out$lbin_vector
out$lcomp

# Change the population length bins:
out <- change_data(file_in, outfile = NULL, types = "len",
  years = years, fleets = fleets, pop_binwidth = 1, pop_minimum_size = 5,
  pop_maximum_size = 210)
out$binwidth
out$maximum_size
out$minimum_size
```

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", par_name = NULL,
  par_int = "NA", par_phase = "NA", forecast_num = 0,
  verbose = FALSE, natM_type = NULL, natM_n_breakpoints = NULL,
  natM_lorenzen = NULL, natM_val = NULL)
```

Arguments

ctl_file_in	A string providing the path to the input SS .ctl file.
ctl_file_out	A string providing the path to the output SS control file. If the value is NULL, the file will not be written to the disk.
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	A string providing the path to the input SS forecast.ss file.
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.
verbose	When TRUE messages will be returned from the function. Often useful for debugging. The default is FALSE.
natM_type	Deprecated. Should have value NULL.
natM_n_breakpoints	Deprecated. Should have value NULL.
natM_lorenzen	Deprecated. Should have value NULL.
natM_val	Deprecated. Should have value NULL.

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 specify 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_par](#), [change_f](#), [change_o](#), [change_retro](#), [change_tv](#)

Examples

```
## Not run:

d <- system.file("extdata", "models", "cod-om", package = "ss3sim")
data.old <- r4ss::SS_readdat(
  system.file("extdata", "models", "cod-om", "codOM.dat",
    package = "ss3sim"),
  version = NULL, verbose = FALSE)
change_e(
  ctl_file_in = file.path(d, "codOM.ctl"),
  ctl_file_out = file.path(tempdir(), "change_e.ctl"),
  dat_list = data.old,
  for_file_in = file.path(d, "forecast.ss"),
  natM_type = NULL, natM_n_breakpoints = NULL,
  natM_lorenzen = NULL, natM_val = NULL,
  par_name = c("_steep", "SizeSel_P1_Fishery(1)"),
  par_int = c(0.3, 40), par_phase = c(3, 2),
  forecast_num = 0)
# clean up the temporary files
file.remove(file.path(tempdir(), "change_e.ctl"))
```

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

change_em_binning	<i>Change population and observed length composition bins in an SS estimation model</i>
-------------------	---

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 contains conditional age-at-length data then these data will be re-binned as well.

Usage

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

Arguments

dat_list	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 specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.
bin_vector	A numeric vector of new length bins to substitute into the 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

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_par](#), [change_f](#), [change_o](#), [change_retro](#), [change_tv](#)

Examples

```

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

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

#An example with lbin_method = 2
new_bin_vec <- seq(min(d$lbin_vector), max(d$lbin_vector), by = 4)
# add the max value if necessary.
if(new_bin_vec[length(new_bin_vec)] != d$lbin_vector[length(d$lbin_vector)]){
  new_bin_vec <- c(new_bin_vec,
    d$lbin_vector[length(d$lbin_vector)])
}
pop_bin_input <- 5
pop_min_size_input <- min(d$lbin_vector_pop) - 1
pop_max_size_input <- max(d$lbin_vector_pop) + 5
lbin_vec_pop <-seq(pop_min_size_input,
  pop_max_size_input,
  length.out = (pop_max_size_input - pop_min_size_input)/
    pop_bin_input + 1
)
l2 <- change_em_binning(dat_list = d,
  bin_vector = new_bin_vec,
  lbin_method = 2,
  #Note: need more inputs with lbin_method = 2
  pop_binwidth = pop_bin_input,
  pop_minimum_size = pop_min_size_input,
  pop_maximum_size = pop_max_size_input)

l2$lbin_method
# note bin width is now the same as the input
pop_bin_input
l2$binwidth
# note the minimum size has changed based on the input:
pop_min_size_input
l2$minimum_size
# so has max
l2$maximum_size
l2$lbin_vector
#other modified components:
l2$lbin_vector_pop
head(l2$lencomp)

```

Description

Check if forecast years and benchmark years within the forecast file are within the model start year and end year.

Usage

```
change_e_fcast_yrs(styr = 0, endyr_orig = 100, endyr_new = 100,
  fcast_list)
```

Arguments

styr	The model start year, an integer
endyr_orig	The original end year that the forecast file assumed, an integer
endyr_new	The new end year, an integer
fcast_list	forecast file read in using r4ss (is a list)

Value

A changed forecast list.

change_f	<i>Alter fishing mortality (F) using the SS control file</i>
----------	--

Description

Alter fishing mortality (F) for a Stock Synthesis simulation via changes to the control file. The argument years is the only argument that must be a vector, where other vectors, e.g., fisheries, will be repeated if a single value is provided.

Usage

```
change_f(years, fisheries, fvals, seasons = 1, ses = 0.005,
  ctl_file_in, ctl_file_out = "control_fishing.ss")
```

Arguments

years	*Vector of integers that will map to each fvals specifying which year the fishing level pertains to.
fisheries	*Vector of integers that will map to each fvals specifying which fleet the fishing level pertains to. A single value will be repeated for every value in years or $\text{length}(\text{years}) == \text{length}(\text{fisheries})$ must be true.
fvals	*Vector of F values to be entered into the SS control file. A single value will be repeated for every value in years or $\text{length}(\text{years}) == \text{length}(\text{fvals})$ must be true.

seasons	Vector of seasons to be entered into the SS control file. A single value will be repeated for every value in years or <code>length(years) == length(ses)</code> must be true. The default is 1, which will be applied to all fisheries in all years.
ses	Vector of fishing level standard errors (ses) to be entered into the SS control file. A single value will be repeated for every value in years or <code>length(years) == length(ses)</code> must be true. The default is 0.005, which will be applied to all fisheries in all years.
ctl_file_in	A string providing the path to the input SS .ctl file.
ctl_file_out	A string providing the path to the output SS control file. If the value is NULL, the file will not be written to the disk.

Details

Using the control file depends on (1) the starter file is set up to read parameters from the control file rather than the par file and (2) the data file having a dummy catch entry for every year, fishery combination that will be specified in the control file. F values currently in the control file will be removed and the newly specified values will replace them. Users do not need to specify values for years in which there will be zero fishing because SS will be parameterized to assume no fishing in missing years.

The control file is currently read in using `readLines` but will eventually shift to using code specific to Stock Synthesis to alter a structured list. If used with `run_ss3sim`, the case file should be named F. A suggested (default) case letter is F.

Value

Modified SS control file.

Which arguments to specify 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 Faye Johnson

See Also

Other change functions: [change_data](#), [change_em_binning](#), [change_e](#), [change_f_par](#), [change_o](#), [change_retro](#), [change_tv](#)

Examples

```
d <- system.file(file.path("extdata", "models"), package = "ss3sim")
change_f(years = 1:50, fisheries = 1, fvals = 0.2,
  ctl_file_in = file.path(d, "cod-om", "codOM.ctl"),
  ctl_file_out = file.path(tempdir(), "control_fishing.ss"))
```

change_f_par	<i>Alter the fishing mortality (F) values in an SS3 .par file.</i>
--------------	--

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_par(years, years_alter, fvals, par_file_in = "ss.par",
             par_file_out = "ss.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:fishery (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 ss.par file, where <code>length(fvals) == length(years_alter)</code> must be true.
par_file_in	A string providing the path to the input SS .par file.
par_file_out	A string providing the path to the output SS .par file.

Value

A modified SS3 .par file.

Which arguments to specify 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_f](#), [change_o](#), [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_par(years = 1:49, years_alter = 2, fvals = 9999, par_file_in =
par_file, par_file_out = paste0(temp_path, "/test.par"))
```

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 `outfile` 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, outfile = NULL)
```

Arguments

<code>lcomp_constant</code>	<i>*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.</i>
<code>dat_list</code>	An SS data list object as read in from SS_readdat in the r4ss package. Make sure you select option <code>section=2</code> .
<code>outfile</code>	A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.

Details

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

Value

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

Which arguments to specify 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_o

Methods to include parameters in an SS3operating model

Description

change_o takes an SS3 .ctlfile and implements parameter value changes that are NOT time varying. change_o is specifically set up to work with an operating model .ctl file.

Usage

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

Arguments

change_o_list	*A list of named vectors. Names correspond to parameters in the operating model and the vectors correspond to deviations. Alternatively, par_name and par_init can be passed to this function.
ctl_file_in	A string providing the path to the input SS .ctl file.
ctl_file_out	A string providing the path to the output SS control file. If the value is NULL, the file will not be written to the disk.
par_name	*A character vector of parameter names to pass in. NULL unless want to use instead of change_o_list.
par_int	*A numeric vector of parameter initial values to pass in. NULL unless want to use instead of change_o_list. Must have the same length and be in the same order as par_names, as the names should correspond with their initial values.

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

Value

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

Which arguments to specify 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_o_list

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

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

Passing arguments to change_o through run_ss3sim

(1) 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_o` in your case file. For example, you might want to use M for natural mortality, S for selectivity, or G for growth.

Author(s)

Kathryn Doering

See Also

Other change functions: [change_data](#), [change_em_binning](#), [change_e](#), [change_f_par](#), [change_f](#), [change_retro](#), [change_tv](#)

change_pop_bin

Set up population length bin structure

Description

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

Usage

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

Arguments

dat_list	An SS data list object as read in from SS_readdat in the r4ss package. Make sure you select option section=2.
binwidth	A numeric value specifying the width of the size bins.
minimum_size	The smallest size bin.
maximum_size	The largest size bin.
maximum_age	The highest age. Used to structure the maximum age of the population and the ageing-error matrix, which will be assumed to have no bias and maximum precision for any added ages.

Details

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

Value

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

change_rec_devs	<i>Replace recruitment deviations</i>
-----------------	---------------------------------------

Description

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

Usage

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

Arguments

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

Details

This function does not need to be specified in a case file if you are running an ss3sim simulation using [run_ss3sim](#).

Value

A modified SS control file.

Author(s)

Kelli Faye Johnson

Examples

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

change_rec_devs_par *Replace recruitment deviations*

Description

This function replaces the recruitment deviations in the ss.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_par(recdevs_new, par_file_in = "ss.par",
  par_file_out = "ss.par")
```

Arguments

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

Details

This function does not need to be specified in a case file if you are running an ss3sim simulation through case files with [run_ss3sim](#).

Value

A modified SS3 .par file.

Author(s)

Cole Monnahan

change_retro	<i>Alter a starter file for a retrospective analysis</i>
--------------	--

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 SS starter .ss file.
 str_file_out A string providing the path to the output SS 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 specify 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_par](#), [change_f](#), [change_o](#), [change_tv](#)

Examples

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

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

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

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

`change_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, outfile = NULL)
```

Arguments

`tail_compression`

*The new `tail_compression` value to be used. Must be a numeric value, as a proportion. For example 0.1 means 10 percent. See the 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`.

`outfile`

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

Value

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

Which arguments to specify 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	<i>Methods to include time-varying parameters in an SS3 operating model</i>
-----------	---

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")
```

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 SS .ctl file.
ctl_file_out	A string providing the path to the output SS control file. If the value is NULL, the file will not be written to the disk.
dat_file_in	A string providing the path to the input SS .dat file.
dat_file_out	A string providing the path to the output SS .dat 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 .ctl and .dat files if ctl_file_out and dat_file_out are not NULL. The function also returns a list of the modified .ctl and .dat R objects invisibly.

Which arguments to specify 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 in 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, Kelli Johnson, and Kathryn Doering

See Also

Other change functions: [change_data](#), [change_em_binning](#), [change_e](#), [change_f_par](#), [change_f](#), [change_o](#), [change_retro](#)

Examples

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

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

```

setwd("cod-om")

change_tv(change_tv_list =
  list("NatM_p_1_Fem_GP_1" = c(rep(0, 20), rep(.1, 80)),
       "SR_BH_steep"=rnorm(100, 0, 0.05)),
  ctl_file_in = "codOM.ctl",
  ctl_file_out = "example.ctl",
  dat_file_in = "codOM.dat",
  dat_file_out = "example.dat")

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

## End(Not run)

```

check_data	<i>Check that the SS data file looks correct</i>
------------	--

Description

Check that the SS data file looks correct

Usage

```
check_data(x)
```

Arguments

x An SS data list object as read in by [SS_readdat](#).

check_data_str_range	<i>Check input arguments for data</i>
----------------------	---------------------------------------

Description

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

Usage

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

Arguments

all_params A named list of the parameters containing at a minimum year and fleet values
dat_list An SS 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,
  mlacom_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 <code>sample_index</code> .
lcomp_params	Named lists containing the arguments for <code>sample_lcomp</code> .
agecomp_params	Named lists containing the arguments for <code>sample_agecomp</code> .
calcomp_params	Named lists containing the arguments for <code>sample_calcomp</code> .
mlcomp_params	Named lists containing the arguments for <code>sample_mlcomp</code> .
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_mlcomp](#), [sample_wtatage](#)

copy_ss3models	<i>Copy the operating and estimation models and create a folder structure</i>
----------------	---

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 <code>vignette("ss3sim-vignette")</code> or <code>get_caseargs</code> 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 invisible 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)
```

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

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

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 case letters, case numbers, and species codes. Scenarios are passed to [run_ss3sim](#) and [get_results_all](#). Case letters 'D' and 'F' are mandatory and provide the data sampling and fishing history for the operating model.

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 C. 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")
```

facet_form	<i>A helper function for building a ggplot facet. Used internally by the plotting functions.</i>
------------	--

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	<i>Fill in matrix across rows of weight-at-age data by interpolation</i>
-------------	--

Description

Function that fills in matrix across rows of wtage 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	<i>Take a csv file, read it, and turn the first column into the list names and the second column into the list values.</i>
----------	--

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	<i>Get SS3 binary/executable location in package</i>
---------	--

Description

Get SS3 binary/executable location in package

Usage

```
get_bin(bin_name = "ss")
```

Arguments

bin_name	Name of SS3 binary, defaults to "ss_safe"
----------	---

Value

The path to an SS3 binary. If using the GitHub version of the package, this will be an internal binary. Otherwise, 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	<i>Get the parameter values for change_bin</i>
--------------	--

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	<i>Take a scenario ID and return argument lists</i>
--------------	---

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	<i>Take a scenario ID and a case type and return the case number</i>
-------------	--

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	<i>Get the folder location of the FISH600 case files</i>
------------------------	--

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	<i>Get the folder location of an included SS3 model configuration</i>
------------------	---

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	<i>Get negative log likelihood (NLL) values from a report file list</i>
--------------------	---

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	<i>Return a set of recruitment deviations</i>
-------------	---

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	<i>Extract SS3 simulation output</i>
-----------------	--------------------------------------

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 iterations. It writes two data.frames to file: one for single scalar values (e.g., MSY) and a second that contains output for each year of the same model (timeseries, e.g., biomass(year)). These can always be joined later.

Usage

```
get_results_all(directory = getwd(), overwrite_files = FALSE,
  user_scenarios = NULL, 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 iterations are run.
user_scenarios	A character vector of scenarios that should be read in. Default is NULL, which indicates find all scenario folders in directory.
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 iterations inside a supplied scenario folder. The function writes 3 .csv files to the scenario folder: (1) scalar metrics with one value per iteration (e.g. R_0 , h), (2) a timeseries data ('ts') which contains multiple values per iteration (e.g. SSB_y for a range of years y), and (3) [currently disabled and not tested] residuals on the log scale from the surveys across all iterations. 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 iterations 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 <- file.path(d, "eg-cases")
om <- file.path(d, "models", "cod-om")
em <- file.path(d, "models", "cod-em")
run_ss3sim(iterations = 1:2, scenarios =
  c("D0-F0-cod"),
  case_folder = case_folder, om_dir = om, em_dir = em,
  case_files = list(F = "F",
                    D = c("index", "lcomp", "agecomp")),
  bias_adjust = FALSE)
get_results_scenario(c("D0-F0-cod"), overwrite_files = TRUE)

#clean up
unlink("D0-F0-cod", recursive = TRUE)

## 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 Variability About Recruitment Deviations (σ_R)

Description

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

Usage

```
get_sigmar(om)
```

Arguments

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

Author(s)

Kelli Johnson

get_ss_ver_dl

Get the ss version (either 3.24 or 3.30) from a dat_list list.

Description

Get the SS version from a list dat_list that was a originally read in using [SS_readdat](#).

Usage

```
get_ss_ver_dl(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.

get_ss_ver_file	<i>Get the ss version (either 3.24 or 3.30) from an ss file</i>
-----------------	---

Description

Get the SS version from the top line in an SS file. as done in [SS_readdat](#).

Usage

```
get_ss_ver_file(file)
```

Arguments

file	Input SS3 control file, either a starter, control, or data SS file.
------	---

id_scenarios	<i>Identify ss3sim scenarios within a directory</i>
--------------	---

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

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 iteration.

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 iteration.

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 iteration.

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 iteration.

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 iteration.

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 (F) levels to determine the F at maximum sustainable yield (Fmsy).

Usage

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

Arguments

om_in	A full or relative path to a directory that contains an ss3sim operating model.
results_out	A full or relative path to a directory where the results will be saved. The directory will be created if it doesn't already exist.
start	Lower fishing mortality levels that will be explored.
end	Upper fishing mortality levels that will be explored.
by_val	Interval at which F will be incremented between start and end.
verbose	When TRUE messages will be returned from the function. Often useful for debugging. The default is FALSE.

Details

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

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

Value

Creates a plot and a table with catches and F values. Also, invisibly returns a table of F and catch as a data frame.

Examples

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

## End(Not run)
```

remove_CPUE	<i>Remove a q setup line into an SS control file</i>
-------------	--

Description

This function removes a q setup line from a SS 3.30 control file

Usage

```
remove_CPUE(string, ctl.in, ctl.out, dat.in, dat.out, overwrite = FALSE)
```

Arguments

string	A string with the fleetname to remove.
ctl.in	An SS control file name to read in.
ctl.out	The SS control file to read out.
dat.in	An SS data file name to read in.
dat.out	An SS data file name to read out.
overwrite	Logical. Overwrite an existing file with the same name as ctl.out or data.out?

Value

A modified SS control file.

Author(s)

Kelli Johnson

remove_q_ctl	<i>Remove a q setup line into an SS control file only</i>
--------------	---

Description

This function removes a q setup line from a SS 3.30 control file

Usage

```
remove_q_ctl(string, ctl.in, filename = TRUE, ctl.out,
  overwrite = FALSE)
```

Arguments

string	A string with the fleetname to remove.
ctl.in	An SS control file name to read in if filename = TRUE otherwise an SS control file vector already read using readLines()
filename	Does function expect ctl.in to be a filename? defaults to TRUE. See ctl.in definition.
ctl.out	The SS control file to read out.
overwrite	Logical. Overwrite an existing file with the same name as ctl.out or data.out?

Value

A modified SS control file vector.

Author(s)

Kelli Johnson

rename_ss3_files	<i>Rename SS3-version-specific files</i>
------------------	--

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_ss3model	<i>Run an operating or estimation model for a specified set of scenario IDs</i>
--------------	---

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, 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.
show.output.on.console	Logical: passed on to <code>system</code> .
...	Anything else to pass to <code>system</code> .

Details

ss3sim requires you to place the SS executable 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#](https://www.dropbox.com/sh/zg0sec6j20sfyyz/AACQiuk787qW882U2euVKoPna)

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 <code>c("D0-F0-cod", "D1-F1-cod")</code> . 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. <code>user_recdevs</code> 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). <code>user_recdevs</code> will <i>not</i> be multiplied by sigma R and they will <i>not</i> be log-normal bias corrected. If <code>user_recdevs</code> 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 <code>ss3sim_base</code> . This could include <code>bias_adjust</code> . Also, you can pass additional options to the SS3 command through the argument <code>admb_options</code> .

Details

The operating model folder should contain: `forecast.ss`, `yourmodel.ct1`, `yourmodel.dat`, `ss.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 `ss.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/1/om
- D0-F0-cod/1/em
- D0-F0-cod/2/om
- ...

Author(s)

Sean C. Anderson

See Also

[ss3sim_base](#), [run_ss3model](#), [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)
```

```

on.exit(setwd(wd), add = TRUE)

# Find the data in the ss3sim package:
d <- system.file("extdata", package = "ss3sim")
om <- file.path(d, "models", "cod-om")
em <- file.path(d, "models", "cod-em")
case_folder <- file.path(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-E0-cod", recursive = TRUE) # clean up

# If try to use bias adjustment, a warning will be triggered and the run will
# proceed WITHOUT using bias adjustment (and may result in error.)
# run_ss3sim(iterations = 1, scenarios = "D1-F0-cod",
#   case_folder = case_folder, om_dir = om, em_dir = em,
#   bias_adjust = TRUE)

# A run with deterministic process error for model checking:
recdevs_det <- matrix(0, nrow = 101, ncol = 2)
run_ss3sim(iterations = 1:2, scenarios = "D0-E100-F0-cod",
  case_folder = case_folder,
  case_files = list(F = "F", D = c("index", "lcomp", "agecomp"), E = "E"),
  om_dir = om, em_dir = em,
  bias_adjust = FALSE, 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 = "D0-F0-cod",
#   case_folder = case_folder, om_dir = om, em_dir = em,
#   parallel = TRUE, parallel_iterations = TRUE)
# unlink("D0-F0-cod", recursive = TRUE)

```

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

sample_agecomp	<i>Sample age compositions from a Stock Synthesis data file</i>
----------------	---

Description

Extract age-composition data from a .ss_new data file and sample the data. It is assumed that the composition data will be expected values as written by Stock Synthesis in the second section of the data file, but one can also sample input data. The resulting age-composition data are assumed to represent observed age composition and will overwrite the age data in `dat_list`, which is returned invisibly. The data file can also be written to the disk, if a file path is provided to `outfile`, and used as simulated data by an estimation model. 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 = NULL, fleets, Nsamp, years,
               cpar = 1, ESS = NULL, keep_conditional = TRUE)
```

Arguments

<code>dat_list</code>	An SS data list object as read in from <code>SS_readdat</code> in the r4ss package. Make sure you select option <code>section=2</code> .
<code>outfile</code>	A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of <code>NULL</code> , which leads to increased speed because writing the file takes time and computing resources.
<code>fleets</code>	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of <code>fleets=NULL</code> leads to zero samples for any fleet.
<code>Nsamp</code>	*A numeric list of the same length as <code>fleets</code> . 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 <code>Nsamp=NULL</code> .
<code>years</code>	*A list the same length as <code>fleets</code> giving the years as numeric vectors. If no fleet collected samples, keep the value to <code>years=NULL</code> .
<code>cpar</code>	A numeric value or vector the same length as <code>fleets</code> controlling the variance of the Dirichlet distribution used for sampling. A value of 1 leads to the same standard deviation as a multinomial of the given <code>Nsamp</code> , 2 indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion. <code>NULL</code> or <code>NA</code> for a given fleet will lead to no dispersion.

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

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

Value

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

Which arguments to specify 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

See Also

Other sampling functions: [clean_data](#), [sample_calcomp](#), [sample_index](#), [sample_lcomp](#), [sample_mlacom](#), [sample_wtatage](#)

Examples

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

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

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

## Generate with varying Nsamp by year for first fleet
ex2 <- sample_agecomp(dat_list = dat_list, outfile = NULL,
```

```

fleets = c(1, 2),
Nsamp = list(c(rep(50, 5), rep(100, 5)), 50),
years = list(seq(26, 44, 2), c(26:100)))

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

```

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

Description

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

Usage

```
sample_calcomp(dat_list, outfile = NULL, fleets = c(1, 2), years,
              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 specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.
fleets	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of fleets=NULL leads to zero samples for any fleet.
years	*A list the same length as fleets giving the years as numeric vectors. If no fleet collected samples, keep the value to years=NULL.
Nsamp	*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 !is.null(outfile). A list object containing the modified .dat file is returned invisibly.

Which arguments to specify 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_mlcomp](#), [sample_wtatage](#)

sample_comp

*Sample composition data from expected values***Description**

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

Usage

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

Arguments

data	A data frame with informational columns followed by columns of compositional data. The informational columns must include columns labeled 'Yr' and 'FltSvy' and end with a column labeled 'Nsamp'. Columns of compositional data should follow 'Nsamp'. Rows of compositional data do not need to sum to one.
Nsamp	*A numeric list of the same length as fleets. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to Nsamp=NULL.
fleets	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of fleets=NULL leads to zero samples for any fleet.
years	*A list the same length as fleets giving the years as numeric vectors. If no fleet collected samples, keep the value to years=NULL.
ESS	The final effective sample size (ESS) associated with the simulated data. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, NULL, leads to the true (internally calculated) #' ESS being used, which is Nsamp for the multinomial case or given by the formula under cpar for the Dirichlet case. At least one value must be provided for each fleet or a vector of year-specific values can be used for any given fleet. The argument accepts a list with entries, either a single integer or a vector of integers, for each fleet.
cpar	A numeric value or vector the same length as fleets controlling the variance of the Dirichlet distribution used for sampling. A value of 1 leads to the same standard deviation as a multinomial of the given Nsamp, 2 indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion. NULL or NA for a given fleet will lead to no dispersion.

Details

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

Value

A data frame of observed composition data.

Author(s)

Kelli Faye Johnson

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: $B_y \cdot \exp(\text{rnorm}(1, 0, \text{sds_obs}) - \text{sds_obs}^2/2)$. 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 = NULL, fleets, years, sds_obs,
             make_plot = FALSE)
```

Arguments

<code>dat_list</code>	An SS data list object as read in from <code>SS_readdat</code> in the <code>r4ss</code> package. Make sure you select option <code>section=2</code> .
<code>outfile</code>	A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of <code>NULL</code> , which leads to increased speed because writing the file takes time and computing resources.
<code>fleets</code>	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of <code>fleets=NULL</code> leads to zero samples for any fleet.
<code>years</code>	*A list the same length as <code>fleets</code> giving the years as numeric vectors. If no fleet collected samples, keep the value to <code>years=NULL</code> .
<code>sds_obs</code>	*A list the same length as <code>fleets</code> . 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.
<code>make_plot</code>	A logical switch for whether to make a crude plot showing the results. Useful for testing and exploring the function.

Value

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

Which arguments to specify 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 <- file.path(d, "example-om", "ss3_expected_values.dat")
dat_list <- r4ss::SS_readdat(f_in, version = NULL, verbose = FALSE)
# Note the initial expected values for the index data:
dat_list$CPUE # Only has expected values for fleet 2 in every other year from
# 76 to 100, so can only sample from fleet 2 during every other year between
# 76 and 100
sam_yrs <- seq(76, 100, by = 2)
ex1 <- sample_index(dat_list,
                    outfile = NULL,
                    fleets = 2,
                    years = list(sam_yrs),
                    sds_obs=list(seq(.001, .1,
                                     length.out = length(sam_yrs))))

ex1$CPUE
# could sample from less years, but not more:
ex2 <- sample_index(dat_list,
                    outfile = NULL,
                    fleets = 2,
                    years = list(sam_yrs[c(-1, -2)]),
                    sds_obs=list(seq(.001, .1,
                                     length.out = length(sam_yrs[c(-1, -2)]))))

ex2$CPUE
# Also, sd can be fixed across years:
ex3 <- sample_index(dat_list,
                    outfile = NULL,
                    fleets = 2,
```

```

                                years = list(sam_yrs),
                                sds_obs=list(0.01))
ex3$CPUE
# If fleet 1 also had expected values in the index that you wanted to sample:
# ex4 <- sample_index(dat_list,
#                    outfile = NULL,
#                    fleets = c(1,2),
#                    years = list(sam_yrs, sam_yrs),
#                    sds_obs=list(0.01, 0.01))

## End(Not run)

```

sample_lcomp

Sample length compositions from a Stock Synthesis data file

Description

Extract length-composition data from a `.ss_new` data file and sample the data. It is assumed that the composition data will be expected values as written by Stock Synthesis in the second section of the data file, but one can also sample input data. The resulting length-composition data are assumed to represent observed length composition and will overwrite the length data in `dat_list`, which is returned invisibly. The data file can also be written to the disk, if a file path is provided to `outfile`, and used as simulated data by an estimation model. If used with `run_ss3sim`, the case file should be named `agecomp`. A suggested (default) case letter is D for data.

Usage

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

Arguments

<code>dat_list</code>	An SS data list object as read in from <code>SS_readdat</code> in the <code>r4ss</code> package. Make sure you select option <code>section=2</code> .
<code>outfile</code>	A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of <code>NULL</code> , which leads to increased speed because writing the file takes time and computing resources.
<code>fleets</code>	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of <code>fleets=NULL</code> leads to zero samples for any fleet.
<code>Nsamp</code>	*A numeric list of the same length as <code>fleets</code> . 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 <code>Nsamp=NULL</code> .
<code>years</code>	*A list the same length as <code>fleets</code> giving the years as numeric vectors. If no fleet collected samples, keep the value to <code>years=NULL</code> .

cpar	A numeric value or vector the same length as fleets controlling the variance of the Dirichlet distribution used for sampling. A value of 1 leads to the same standard deviation as a multinomial of the given Nsamp, 2 indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion. NULL or NA for a given fleet will lead to no dispersion.
ESS	The final effective sample size (ESS) associated with the simulated data. The ESS is not used to generate the simulated data but can be used as an input sample size in subsequent models that estimate population parameters or status. The default, NULL, leads to the true (internally calculated) #' ESS being used, which is Nsamp for the multinomial case or given by the formula under cpar for the Dirichlet case. At least one value must be provided for each fleet or a vector of year-specific values can be used for any given fleet. The argument accepts a list with entries, either a single integer or a vector of integers, for each fleet.

Value

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

Which arguments to specify 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

See Also

[sample_agecomp](#) for more examples

Other sampling functions: [clean_data](#), [sample_agecomp](#), [sample_calcomp](#), [sample_index](#), [sample_mlacom](#), [sample_wtatage](#)

Examples

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

sample_mlacomp	<i>[BETA VERSION] Sample mean length (size-)-at-age data and write to file for use by the EM.</i>
----------------	---

Description

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

Usage

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

Arguments

dat_list	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 specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.
ctl_file_in	A path to the control file, output from an OM, containing the OM parameters for growth. These values are used to determine the uncertainty about size for fish sampled in each age bin.
fleets	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of fleets=NULL leads to zero samples for any fleet.
Nsamp	*A numeric list of the same length as fleets. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to Nsamp=NULL.
years	*A list the same length as fleets giving the years as numeric vectors. If no fleet collected samples, keep the value to years=NULL.
mean_outfile	A path to write length and age data for external estimation of parametric growth. If NULL no file will be written. This file is used by <code>change_e</code> to externally estimate growth parameters. Filename must contain "vbgf" to be used by <code>change_e</code> . 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 `!is.null(outfile)`. A list object containing the modified `.dat` file is returned invisibly.

Which arguments to specify 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

```
d <- system.file(file.path("extdata", "models", "cod-om"),
  package = "ss3sim")
dat_in <- file.path(d, "codOM.dat")
dat_list <- r4ss::SS_readdat(dat_in, version = NULL, verbose = FALSE)
dat_list <- change_data(dat_list, outfile = NULL,
  fleets = 1, years = seq(dat_list$styr, dat_list$styr + 5),
  types = c("age", "mla"))
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(dat_list$styr + 5),
  verbose = FALSE, mean_outfile = NULL)
```

sample_wtatage	<i>Sample empirical weight-at-age data and write to file for use by the EM</i>
----------------	--

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, fleets, cv_wtatage = NULL)
```

Arguments

wta_file_in	The file to read weight-at-age from. Specifically to get the age-0 weight-at-age. This is typically wtatage.ss_new.
outfile	A character string specifying the file name to use when writing the information to the disk. The string must include the proper file extension. No file is written using the default value of NULL, which leads to increased speed because writing the file takes time and computing resources.
dat_list	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, output from an OM, containing the OM parameters for growth and weight/length relationship. These values are used to determine the uncertainty about weight for fish sampled in each age bin. Commonly control.ss_new
years	*A list the same length as fleets giving the years as numeric vectors. If no fleet collected samples, keep the value to years=NULL.
fill_fnc	*A function to fill in missing values (ages and years). The resulting weight-at-age file will have values for all years and ages. One function is fill_across.
fleets	*A vector of integers specifying which fleets to include. The order of the fleets pertains to the input order of other arguments. An entry of fleets=NULL leads to zero samples for any fleet.
cv_wtatage	A user specified CV for growth. Default is NULL.

Value

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

Which arguments to specify 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

<code>x</code>	The admb options
<code>exclude</code>	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	<i>Setup parallel processing</i>
----------------	----------------------------------

Description

Setup parallel processing

Usage

```
setup_parallel()
```

ss3sim	<i>ss3sim: Fisheries stock assessment simulation testing with Stock Synthesis</i>
--------	---

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,
  mlacom_params = NULL, em_binning_params = NULL,
  estim_params = NULL, tv_params = NULL, operat_params = NULL,
  om_dir, em_dir, retro_params = NULL, data_params = NULL,
  user_recdevs = NULL, user_recdevs_warn = TRUE, bias_adjust = FALSE,
  hess_always = FALSE, print_logfile = TRUE, sleep = 0, seed = 21,
  ...)
```

Arguments

iterations	Which iterations to run. A numeric vector.
scenarios	Which scenarios to run.
f_params	A named list containing arguments for <code>change_f</code> . A mandatory case.
index_params	A named list containing arguments for <code>sample_index</code> . A mandatory case.
lcomp_params	A named list containing arguments for <code>sample_lcomp</code> . A mandatory case.
agecomp_params	A named list containing arguments for <code>sample_agecomp</code> . A mandatory case.
calcomp_params	A named list containing arguments for <code>sample_calcomp</code> , for conditional age-at-length data. Currently CAL is not implemented in this version of ss3sim, so <code>calcomp_params</code> should be NULL.
wtatage_params	A named list containing arguments for <code>sample_wtatage</code> , for empirical weight-at-age data.
mlacomp_params	A named list containing arguments for <code>sample_mlacomp</code> , for mean length-at-age data.
em_binning_params	A named list containing arguments for <code>change_em_binning</code> .
estim_params	A named list containing arguments for <code>change_e</code> .
tv_params	A named list containing arguments for <code>change_tv</code> (time-varying).
operat_params	A named list containing arguments for <code>change_o</code> .
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 <code>change_retro</code> .
data_params	A named list containing arguments for <code>change_data</code> .
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. <code>user_recdevs</code> 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). <code>user_recdevs</code> will <i>not</i> be multiplied by sigma R and they will <i>not</i> be log-normal bias corrected. If <code>user_recdevs</code> 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 <code>user_recdevs</code> are specified.
bias_adjust	Run bias adjustment first?.
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.
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.
...	Anything extra to pass to run_ss3model . For example, you may want to pass additional options to SS3 through the argument <code>admb_options</code> . Anything that doesn’t match a named argument in run_ss3model will be passed to the system call that runs SS3.

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/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:
# Create a temporary folder for the output and set the working directory:
temp_path <- file.path(tempdir(), "ss3sim-base-example")
dir.create(temp_path, showWarnings = FALSE)
wd <- getwd()
setwd(temp_path)
on.exit(setwd(wd), add = TRUE)

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

```

em <- file.path(d, "models", "cod-em")
case_folder <- file.path(d, "eg-cases")

# Pull in file paths from the package example data:
d <- system.file("extdata", package = "ss3sim")
om_dir <- file.path(d, "models", "cod-om")
em_dir <- file.path(d, "models", "cod-em")
a <- get_caseargs(folder = file.path(d, "eg-cases"),
                  case_files = list(F = "F",
                                    D = c("index", "lcomp", "agecomp"),
                                    E = "E"),
                  scenario = "F0-D0-E0-cod")
ss3sim_base(iterations = 1,
            scenarios = "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("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 = 1:100,
           fisheries = 1,
           fvals = c(rep(0, 25), rep(0.114, 75)))

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

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

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

E0 <- list(natM_type = NULL, natM_n_breakpoints = NULL, natM_lorenzen = NULL,
           natM_val = NULL,
           par_name = c("LnQ_base_Fishery", "NatM_p_1_Fem_GP_1"),
           par_int = c(NA, NA), par_phase = c(-1, -1), forecast_num = 0)

ss3sim_base(iterations = 1,
            scenarios = "D1-E0-F0-cod", #name as desired
            f_params = F0,
            index_params = index1,
            lcomp_params = lcomp1,
            agecomp_params = agecomp1,
            estim_params = E0,

```

```

        om_dir = om,
        em_dir = em)

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

## 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

<code>percent_df</code>	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.
<code>dir</code>	A path to the directory containing the model files.
<code>em_ctl_file</code>	A string with the name of the estimation model control file. <code>em_ctl_file</code> must be located in <code>dir</code> .
<code>om_ctl_file</code>	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 <code>""</code> . <code>om_ctl_file</code> must be located in <code>dir</code> .
<code>verbose</code>	Detailed output to command line. Default is FALSE.
<code>estimate</code>	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.
<code>...</code>	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)
on.exit(setwd(wd), add = TRUE)

## 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
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 = ".",
  em_ctl_file = "em.ctl",
  om_ctl_file = "om.ctl")
unlink("om.ctl")
unlink("em.ctl")
unlink(temp_path, recursive = TRUE)

## 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	<i>Example time series data from the ss3sim vignette</i>
--------	--

Description

Example time series data from the ss3sim vignette

vbgf_func	<i>Predict length given VBGF parameters</i>
-----------	---

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	<i>Verify and standardize SS3 input files</i>
--------------	---

Description

This function verifies the contents of operating model (om) and estimation model (em) folders (i.e., it checks that the necessary SS input files are available). If the contents are correct, the .ctl and .dat files are renamed to standardized names and the starter.ss file is updated to reflect these names. If the contents are incorrect then a warning is issued and the simulation is aborted.

Usage

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

Arguments

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

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 iteration.

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

- * **change functions**
 - change_data, 12
 - change_e, 15
 - change_em_binning, 17
 - change_f, 19
 - change_f_par, 21
 - change_o, 23
 - change_retro, 27
 - change_tv, 30
- * **data**
 - scalar_dat, 76
 - ts_dat, 84
- * **get-results**
 - get_results_all, 44
 - get_results_derived, 45
 - get_results_scalar, 46
 - get_results_scenario, 46
 - get_results_timeseries, 47
- * **sample functions**
 - change_em_binning, 17
- * **sampling functions**
 - clean_data, 33
 - sample_agecomp, 64
 - sample_calcomp, 66
 - sample_index, 69
 - sample_lcomp, 71
 - sample_mlcomp, 73
 - sample_wtatage, 75
- add_colnames, 4
- add_CPUE, 5
- add_nulls, 5
- add_tv_parlines, 6
- calculate_data_units, 6, 12
- calculate_re, 7
- case_comp, 8
- case_deparse, 9
- case_fishing, 10
- case_index, 10
- case_tv, 11
- change_data, 7, 12, 16, 17, 20, 22, 24, 28, 31, 33, 79
- change_e, 14, 15, 17, 20, 22, 24, 28, 31, 78, 79
- change_e_fcast_yrs, 18
- change_em_binning, 14, 16, 17, 20, 22, 24, 28, 31, 79
- change_f, 14, 16, 17, 19, 22, 24, 28, 31, 77, 79
- change_f_par, 14, 16, 17, 20, 21, 24, 28, 31
- change_lcomp_constant, 22, 78
- change_o, 14, 16, 17, 20, 22, 23, 28, 31, 79
- change_pop_bin, 24
- change_rec_devs, 25, 78
- change_rec_devs_par, 26
- change_retro, 14, 16, 17, 20, 22, 24, 27, 31, 78, 79
- change_tail_compression, 29, 78
- change_tv, 14, 16, 17, 20, 22, 24, 28, 30, 36, 77, 79
- check_data, 32
- check_data_str_range, 32
- clean_data, 12, 33, 65, 67, 70, 72, 74, 76
- cleanup_ss3, 33
- copy_ss3models, 34
- create_argfiles, 35
- expand_scenarios, 37, 61, 62
- facet_form, 38
- fill_across, 38, 39, 76
- get_args, 39
- get_bin, 39
- get_bin_info, 40
- get_caseargs, 35, 40, 61, 62
- get_caseval, 42
- get_fish600_casefolder, 42
- get_model_folder, 43
- get_nll_components, 43
- get_recdevs, 44, 80

- get_results_all, [7](#), [8](#), [37](#), [44](#), [45–48](#), [50](#), [51](#),
[53–55](#), [78](#), [86](#)
- get_results_derived, [45](#), [45](#), [46–48](#)
- get_results_scalar, [45](#), [46](#), [47](#), [48](#)
- get_results_scenario, [45](#), [46](#), [46](#), [48](#), [78](#)
- get_results_timeseries, [45–47](#), [47](#)
- get_sigmar, [48](#)
- get_ss_ver_dl, [48](#)
- get_ss_ver_file, [49](#)

- id_scenarios, [49](#)

- plot_scalar_boxplot, [50](#)
- plot_scalar_points, [51](#)
- plot_ts_boxplot, [52](#)
- plot_ts_lines, [54](#)
- plot_ts_points, [55](#)
- profile_fmasy, [56](#)

- rbind, [4](#)
- rbind.fill, [4](#)
- remove_CPUE, [58](#)
- remove_q_ctl, [58](#)
- rename_ss3_files, [59](#)
- run_ss3model, [60](#), [62](#), [78](#), [80](#)
- run_ss3sim, [14–16](#), [20–24](#), [26–29](#), [31](#), [36](#), [37](#),
[40](#), [45](#), [60](#), [61](#), [61](#), [64](#), [65](#), [67](#), [69–72](#),
[74–76](#), [78](#), [80](#)

- sample_agecomp, [7](#), [14](#), [34](#), [39](#), [64](#), [67](#), [70](#), [72](#),
[74](#), [76](#), [78](#), [79](#)
- sample_calcomp, [7](#), [34](#), [65](#), [66](#), [70](#), [72](#), [74](#), [76](#),
[79](#)
- sample_comp, [68](#)
- sample_index, [34](#), [65](#), [67](#), [69](#), [72](#), [74](#), [76](#), [78](#),
[79](#)
- sample_lcomp, [7](#), [14](#), [34](#), [39](#), [65](#), [67](#), [70](#), [71](#),
[74](#), [76](#), [77](#), [79](#)
- sample_mlcomp, [7](#), [34](#), [65](#), [67](#), [70](#), [72](#), [73](#), [76](#),
[79](#)
- sample_wtatage, [7](#), [34](#), [65](#), [67](#), [70](#), [72](#), [74](#), [75](#),
[79](#)

- sanitize_admb_options, [76](#)
- scalar_dat, [76](#)
- set.seed, [44](#)
- setup_parallel, [77](#)
- ss3sim, [77](#)
- ss3sim-package (ss3sim), [77](#)
- ss3sim_base, [12](#), [40](#), [60–62](#), [75](#), [78](#), [78](#)

- SS_changepars, [82](#)
- SS_output, [43](#), [45–48](#)
- SS_readdat, [12](#), [13](#), [15](#), [17](#), [22](#), [25](#), [29](#), [32](#), [34](#),
[48](#), [49](#), [64](#), [66](#), [69](#), [71](#), [73–75](#)
- standardize_bounds, [82](#)
- substr_r, [83](#)
- system, [60](#), [80](#)

- ts_dat, [84](#)

- vbgf_func, [84](#)
- verify_input, [85](#)
- verify_plot_arguments, [86](#)