Package 'ss3sim'

July 2, 2014

Type Package

Title Fisheries stock assessment simulation testing with Stock Synthesis

Version 0.8.2

Date 2014-03-09

Description The ss3sim package develops a framework for fisheries stock assessment simulation testing with Stock Synthesis 3 (SS3).

License MIT + file LICENSE

LazyData true

Suggests knitr, ggplot2, doParallel, foreach

VignetteBuilder knitr

Depends R (>= 3.0.0),

Imports plyr (>= 1.8), r4ss (>= 1.20), gtools (>= 2.7.1), lubridate (>= 1.3.0), reshape2 (>= 1.2.2)

Author Sean Anderson [aut, cre],Cole Monnahan [aut],Kelli Johnson [aut],Kotaro Ono [aut],Juan Valero [aut],Curry Cunningham [aut],Felipe Hurtado-Ferro [aut],Roberto Licandeo [aut],Carey McGilliard [aut],Melissa Muradian [ctb],Cody Szuwalski [aut],Katyana Vert-pre [aut],Athol Whitten [aut]

Maintainer Sean Anderson <sean@seananderson.ca>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-10 00:46:23

R topics documented:

bias_ss3 *Perform a single bias adjustment run*

Description

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

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

Usage

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

Arguments

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

Value

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

Author(s)

Carey McGilliard

References

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

See Also

run_bias_ss3, run_ss3sim, ss3sim_base

calculate_runtime *Calculate run time*

Description

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

Usage

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

Arguments

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

Author(s)

Cole Monnahan

change_agecomp *Sample age compositions from expected values*

Description

Take a `data.SS_new` file containing expected values and sample to create observed age compositions which are then written to file for use by the estimation model.

Usage

```
change_agecomp(infile, outfile, fleets = c(1, 2), Nsamp, years, cpar = 1,
  agebin_vector = NULL, write_file = TRUE)
```

Arguments

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

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

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

cpar — A numeric value or vector the same length as `fleets` controlling the variance of the Dirichlet distribution used for sampling. A value of `1` indicates the same standard deviation as a multinomial of the given `Nsamp`, `2` indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion.

agebin_vector — A numeric vector giving the new age bins to use. `agebin_vector` must be within the [min;max] of population bin. This feature alows dynamic binning by the user, but is not fully tested. Users should consult the vignette and carefully check the function bins the data as desired before proceeding with simulations.

Value

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

Author(s)

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

See Also

change_lcomp

Examples

```
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/example-om/data.ss_new")
infile <- r4ss::SS_readdat(f_in, section = 2, verbose = FALSE)
## Turn off age comps by specifying fleets=NULL
change_agecomp(infile=infile, outfile="test1.dat",
               fleets=NULL, cpar=c(5,NA), Nsamp=list(100,100),
               years=list(1995, 1995), write_file=FALSE)
## Generate with a smaller number of fleet taking samples
ex1 <- change_agecomp(infile=infile, outfile="test1.dat", fleets=c(2),
                      Nsamp=list(c(10,50)), years=list(c(1999,2000)),
                      write_file=FALSE)
## Generate with varying Nsamp by year for first fleet
ex2 <- change_agecomp(infile=infile, outfile="test2.dat", fleets=c(1,2),
                      Nsamp=list(c(rep(50, 5), rep(100, 5)), 50),
                      years=list(seq(1994, 2012, by=2),
                          2003:2012), write_file=FALSE)
## Generate with varying Nsamp by year for first fleet AND with different age bins
ex3 <- change_agecomp(infile=infile, outfile="test3.dat", fleets=c(1,2),
```

```
                      Nsamp=list(c(rep(50, 5), rep(100, 5)), 50),
                      years=list(seq(1994, 2012, by=2),
                          2003:2012), agebin_vector = seq(1,15,by=3),
                          write_file=FALSE)
plot(seq(0,15, by=3), as.numeric(ex3[1, -(1:9)]), type="b", col=2,
     xlab="Age Bin", ylab="Proportion of Age",
     main="Comparison of different age bin structures via agebin_vector")
lines(0:15, as.numeric(ex2[1, -(1:9)]), type="b", col=1)
legend("topright", legend=c("ex2", "ex3"), col=c(1,2), pch=1)

## Run three  cases showing Multinomial, Dirichlet(1) and over-dispersed
## Dirichlet for different levels of sample sizes
op <- par(mfrow = c(1,3))
for(samplesize in c(30, 100, 1000)){
    ex4 <- change_agecomp(infile=infile, outfile="test4.dat", fleets=c(1,2),
                          Nsamp=list(samplesize, samplesize),
                          write_file = FALSE,
                          years=list(2000,2000), cpar=c(NA, 1))
    ex5 <- change_agecomp(infile=infile, outfile="test5.dat", fleets=c(1,2),
                          Nsamp=list(samplesize, samplesize),
                          write_file = FALSE,
                          years=list(2000,2000), cpar=c(1, 1))
    ex6 <- change_agecomp(infile=infile, outfile="test6.dat", fleets=c(1,2),
                          Nsamp=list(samplesize, samplesize),
                          write_file = FALSE,
                          years=list(2000,2000), cpar=c(5, 1))
    true <- subset(infile$agecomp, FltSvy==1 & Yr == 2000)[-(1:9)]
    true <- true/sum(true)
    plot(0:15, subset(ex4, FltSvy==1)[1,-(1:9)], type="b", ylim=c(0,1),
         col=1, xlab="Age", ylab="Proportion", main=paste("Sample size=",samplesize))
    legend("topright", legend=c("Multinomial", "Dirichlet(1)", "Dirichlet(5)", "Truth"),
           lty=1, col=1:4)
    lines((0:15), subset(ex5, FltSvy==1)[1,-(1:9)], type="b", col=2)
    lines((0:15), subset(ex6, FltSvy==1)[1,-(1:9)], type="b", col=3)
    lines((0:15), true, col=4, lwd=2)
}
par(op)
```

---

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

Description

Takes SS3 `.ctl`, `.dat`, and `forecast.ss` files and changes which parameters are estimated, how natural mortality is estimated, and if forecasts are performed. The function can be called by itself or within run_ss3sim to alter an estimation model `.ctl` file.

Usage

```
change_e(ctl_file_in = pastef("em.ctl"), ctl_file_out = pastef("em.ctl"),
```

```
  dat_file_in = pastef("ss3.dat"), for_file_in = "forecasts.ss",
  natM_type = "1Parm", natM_n_breakpoints = NULL, natM_lorenzen = NULL,
  natM_val = c(NA, NA), par_name = NULL, par_int = "NA",
  par_phase = "NA", forecast_num = 0, run_change_e_full = TRUE)
```

Arguments

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

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, .dat, and, forecast.ss files.

Author(s)

Kelli Johnson

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)

d <- system.file("extdata", package = "ss3sim")
ctl_file <- paste0(d, "/models/cod-om/codOM.ctl")
change_e(ctl_file_in = ctl_file, ctl_file_out = "change_e.ctl",
         dat_file_in = "ss3.dat", for_file_in = "forecast.ss",
         natM_type = "n_breakpoints", natM_n_breakpoints = c(1, 4),
         natM_lorenzen = NULL, natM_val = c(.2, 3, 0.4, 5),
         par_name = c("_steep", "SizeSel_1P_1_Fishery"),
         par_int = c(0.3, 40), par_phase = c(3, 2),
         forecast_num = 0, run_change_e_full = TRUE )
# clean up
file.remove("change_e.ctl")
setwd(wd)

## End(Not run)
```

---

change_f *Alter the fishing mortality (F) values in an* ss3.par *file*

Description

Takes an SS3 .par file and changes the F values for specified years.

Usage

```
change_f(years, years_alter, fvals, file_in = "ss3.par",
  file_out = "ss3.par")
```

Arguments

years	Vector of years for which F values are specified
years_alter	Vector of years for the which F values will be altered
fvals	Vector of F values to be entered into ss3.par file
file_in	Input SS3 par file.
file_out	Output SS3 par file.

Value

A modified SS3 .par file.

Author(s)

Curry James Cunningham

Examples

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

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

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

---

change_index *Sample the biomass with observation error*

Description

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

Usage

```
change_index(infile, outfile, fleets, years, sds_obs, make_plot = FALSE,
  write_file = TRUE)
```

Arguments

infile	An SS data object as read in from SS_readdat in the **r4ss** package. Make sure you select option section=2.
outfile	A character string of the new .dat file name to be created. Must end in .dat.
fleets	A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from outfile (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to fleets=NULL.
years	A list the same length as fleets giving the years as numeric vectors. If no fleet collected samples, keep the value to years=NULL.
write_file	A switch for whether to write outfile to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.
sds_obs	A list the same length as fleets. The list should contain either single values or numeric vectors of the same length as the number of years which represent the standard deviation of the observation error. Single values are repeated for all years.
make_plot	A logical switch for whether to make a crude plot showing the results. Useful for testing and exploring the function.

Value

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

Author(s)

Cole Monnahan, Kotaro Ono

Examples

```
## Not run:
# Find the example data location:
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/example-om/data.ss_new")
infile <- r4ss::SS_readdat(f_in, section = 2, verbose = FALSE)
outfile <- "test.dat"
ex1 <- change_index(infile, outfile, fleets=c(2,3),
                    years=list(1938:2012, 1938:2012) ,
                    sds_obs=list(1e-6, 1e-6), write_file=FALSE,
                    make_plot = TRUE)
ex2 <- change_index(infile, outfile, fleets=c(2,3),
                    years=list(1938:2012, 1938:2012) ,
                    sds_obs=list(.05, .05), write_file=FALSE,
                    make_plot = TRUE)
library(ggplot2)
ggplot(ex1, aes(x=year, y=obs, group=index, ymin=0,
                colour=as.factor(index)))+geom_line() + geom_point(data=ex2,
```

```
                aes(x=year, y=obs, colour=as.factor(index), group=index))
## Exclude a fleet and have varying sds_obs by year
ex3 <- change_index(infile, outfile, fleets=c(2,NA),
                    years=list(1938:2012, 1950),
                    sds_obs=list(seq(.001, .1, len=75), .1),
                    write_file=FALSE)
ggplot(ex3, aes(x=year, y=obs, group=index, ymin=0,
                colour=as.factor(index)))+geom_point()

## End(Not run)
```

`change_lcomp` *Sample length compositions from expected values*

Description

Take a `data.SS_new` file containing expected values and sample to create observed length compositions which are then written to file for use by the estimation model.

Usage

```
change_lcomp(infile, outfile, fleets = c(1, 2), Nsamp, years, cpar = 1,
  lengthbin_vector = NULL, write_file = TRUE)
```

Arguments

`infile`	An SS data object as read in from `SS_readdat` in the **r4ss** package. Make sure you select option `section=2`.
`outfile`	A character string of the new `.dat` file name to be created. Must end in `.dat`.
`fleets`	A numeric vector giving the fleets to be used. This order also pertains to other arguments. A missing value excludes that fleet from `outfile` (i.e. it turns it off so no samples are written). If none of the fleet collected samples, keep the value to `fleets=NULL`.
`years`	A list the same length as `fleets` giving the years as numeric vectors. If no fleet collected samples, keep the value to `years=NULL`.
`write_file`	A switch for whether to write `outfile` to disk. Can be turned off to speed up testing or exploration of the function. The new data are returned invisibly, as in the examples below.
`Nsamp`	A numeric list of the same length as `fleets`. Either single values or vectors of the same length as the number of years can be passed through. Single values are repeated for all years. If no fleet collected samples, keep the value to `Nsamp=NULL`.
`cpar`	A numeric value or vector the same length as `fleets` controlling the variance of the Dirichlet distribution used for sampling. A value of `1` indicates the same standard deviation as a multinomial of the given `Nsamp`, `2` indicates twice, etc. Values greater than one indicate overdispersion, and less underdispersion.

`lengthbin_vector`
: A numeric vector giving the new length bins to use. `lengthbin_vector` must be within the [min;max] of population bin. This feature allows dynamic binning by the user, but is not fully tested. Users should consult the vignette and carefully check the function bins the data as desired before proceeding with simulations.

Value

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

Author(s)

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

See Also

change_agecomp

Examples

```
d <- system.file("extdata", package = "ss3sim")
f_in <- paste0(d, "/example-om/data.ss_new")
infile <- r4ss::SS_readdat(f_in, section = 2, verbose = FALSE)

## Generate with constant sample size across years
ex1 <- change_lcomp(infile=infile, outfile="test1.dat", fleets=c(1,2),
                    Nsamp=list(100,50), years=list(seq(1994, 2012, by=2),
                                       2003:2012), write_file = FALSE)

## Generate with varying Nsamp by year for first fleet
ex2 <- change_lcomp(infile=infile, outfile="test2.dat", fleets=c(1,2),
                    Nsamp=list(c(rep(50, 5), rep(100, 5)), 50),
                    years=list(seq(1994, 2012, by=2),
                        2003:2012), write_file = FALSE)

## Generate with constant sample size across years AND with different length
## bins (same as ex1 except bins)
ex3 <- change_lcomp(infile=infile, outfile="test3.dat", fleets=c(1,2),
                    Nsamp=list(100,50), years=list(seq(1994, 2012, by=2),
                          2003:2012), lengthbin_vector = seq(9,30,by=2),
                          write_file = FALSE)

plot(seq(8,30,by=2), as.numeric(ex3[1, -(1:6)]), type="b", col=2,
     xlab="Length Bin", ylab="Proportion of length",
     main="Comparison of different length bin structures via lengthbin_vector")
lines(seq(8, 30, by=.5), as.numeric(ex1[1, -(1:6)]), type="b", col=1)
legend("topright", legend=c("ex1", "ex3"), col=c(1,2), pch=1)

## Plot distributions for a particular year to compare multinomial
## vs. overdispersed Dirichlet
```

```
temp.list <- temp.list2 <- list()
for(i in 1:40){
    temp.list[[i]] <-
      change_lcomp(infile=infile, outfile="test1.dat", fleets=c(2), cpar=c(3),
                    Nsamp=list(100), years=list(1995),
                    write_file=FALSE)
    temp.list2[[i]] <-
        change_lcomp(infile=infile, outfile="test1.dat", fleets=c(2),
                    cpar=c(NA), Nsamp=list(100), years=list(1995),
                    write_file=FALSE)
}
## Organize the data for plotting
x1 <- reshape2::melt(do.call(rbind, temp.list)[,-(1:6)[-3]], id.vars="FltSvy")
x2 <- reshape2::melt(do.call(rbind, temp.list2)[,-(1:6)[-3]], id.vars="FltSvy")
op <- par(mfrow=c(2,1))
with(x1, boxplot(value~variable, las=2, ylim=c(0,.6), ylab="Proportion",
                 main="Overdispersed (cpar=3)",  xlab="length bin"))
temp <- as.numeric(subset(infile$lencomp, Yr==1995 & FltSvy == 2)[-(1:6)])
points(temp/sum(temp), pch="-", col="red")
with(x2, boxplot(value~variable, las=2, ylim=c(0,.6), ylab="Proportion",
                 main="Multinomial", xlab="length bin"))
temp <- as.numeric(subset(infile$lencomp, Yr==1995 & FltSvy == 2)[-(1:6)])
points(temp/sum(temp), pch="-", col="red")
par(op)
```

`change_rec_devs` *Replace recruitment deviations*

Description

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

Usage

```
change_rec_devs(recdevs_new, file_in = "ss3.par", file_out = "ss3.par")
```

Arguments

`recdevs_new`	A vector of new recruitment deviations.
`file_in`	Input SS3 par file.
`file_out`	Output SS3 par file.

Value

A modified SS3 `.par` file.

Author(s)

Cole Monnahan

Examples

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

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

---

change_retro *Alter a starter file for a retrospective analysis*

Description

A retrospective analysis tests the effect of peeling back the number of operating model years observable to the estimation model. This function alters the SS3 starter file to run a retrospective analysis.

Usage

```
change_retro(startfile_in = "starter.ss", startfile_out = "starter.ss",
  retro_yr = 0)
```

Arguments

`startfile_in`	Input `starter.ss` file
`startfile_out`	Output `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.

Author(s)

Sean C. Anderson

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

Description

change_tv takes SS3 .ctl, .par, and .dat files and implements time-varying parameters using environmental variables. change_tv is specifically set up to work with an operating model .ctl file.

Usage

```
change_tv(change_tv_list, ctl_file_in = "control.ss_new",
  ctl_file_out = "om.ctl", dat_file_in = "ss3.dat",
  dat_file_out = "ss3.dat", par_file_in = "ss3.par",
  par_file_out = "ss3.par", starter_file_in = "starter.ss",
  starter_file_out = "starter.ss", report_file = "Report.sso")
```

Arguments

change_tv_list	A list of named vectors. Names correspond to parameters in the operating model that currently do not use environmental deviations and the vectors correspond to deviations. See the section "Specifying the change_tv_list" below for help on specifying this argument.
ctl_file_in	Input SS3 control file
ctl_file_out	Output SS3 control file
dat_file_in	Input SS3 data file
dat_file_out	Output SS3 data file
par_file_in	Input SS3 parameter file

`par_file_out` Output SS3 parameter file

`starter_file_in`
Input SS3 starter file

`starter_file_out`
Output SS3 starter file

`report_file` Input SS3 report file

Details

Although there are three ways to implement time-varying parameters within SS3, **ss3sim** and `change_tv` only use the environmental variable option. Within SS3, time-varying parameters work on an annual time-step. Thus, for models with multiple seasons, the time-varying parameters will remain constant for the entire year.

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

ss3sim uses annual recruitment deviations and may not work with a model that ties recruitment deviations to environmental covariates. If you need to compare the environment to annual recruitment deviations, the preferred option is to transform the environmental variable into an age 0 pre-recruit survey. See page 55 of the SS3 version 3.24f manual for more information.

Value

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

Specifying the `change_tv_list`

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

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

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

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

Passing arguments to `change_tv` through `run_ss3sim`

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

Author(s)

Kotaro Ono, Carey McGilliard, and Kelli Johnson

Examples

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

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

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

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

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

## End(Not run)
```

---

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

---

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

Description

Copy the operating and estimation models and create a folder structure

Usage

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

Arguments

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

Value

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-E0-F0-M0-R0-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-E0-F0-M0-R0-testing", "D1-E1-F0-M0-R0-testing"))
# (Note that all the scenario argument does here is affect the
# folder names.)

# Clean up:
unlink("D0-E0-F0-M0-R0-testing", recursive = TRUE)
unlink("D1-E0-F0-M0-R0-testing", recursive = TRUE)
unlink("D1-E1-F0-M0-R0-testing", recursive = TRUE)
```

---

create_argfiles *Create template argument input files*

Description

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

Usage

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

Arguments

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

Details

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

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

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

Author(s)

Sean Anderson

Examples

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

## End(Not run)
```

`expand_scenarios` *Create vectors of scenario IDs*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Cole Monnahan and Sean Anderson

See Also

run_ss3sim, get_results_all

Examples

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

---

extract_expected_data *Extract the expected data values*

Description

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

Usage

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

Arguments

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

Author(s)

Kotaro Ono

get_args

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

Description

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

Usage

```
get_args(file)
```

Arguments

file	The file name as character

get_caseargs

Take a scenario ID and return argument lists

Description

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

Usage

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

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-E0-F0-M0-R0-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 = "R", S = "S"))
```

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

Description

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

Usage

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

Arguments

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

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

Description

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

Usage

```
get_fish600_casefolder()
```

Value

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

get_model_folder *Get the folder location of an included SS3 model configuration*

Description

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

Usage

```
get_model_folder(folder_name)
```

Arguments

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

Value

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

Examples

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

get_recdevs *Return a set of recruitment deviations*

Description

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

Usage

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

Arguments

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

Value

A vector of standard normal recruitment deviations.

Examples

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

`get_results_all` *Extract SS3 simulation output*

Description

This high level function extracts results from SS3 model runs. Give it a directory which contains directories for different "scenario" runs, within which are replicates and potentially bias adjustment runs. It writes two data.frames to file: one for single scalar values (e.g. MSY) and a second that contains output for each year of the same model (timeseries, e.g. biomass(year)). These can always be joined later.

Usage

```
get_results_all(directory = getwd(), overwrite_files = FALSE,
  user_scenarios = NULL)
```

Arguments

`directory`	The directory which contains scenario folders with results.
`overwrite_files`	A switch to determine if existing files should be overwritten, useful for testing purposes or if new replicates are run.
`user_scenarios`	A character vector of scenarios that should be read in. Default is NULL, which indicates find all scenario folders in `directory`

Value

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

Author(s)

Cole Monnahan

See Also

Other get.results: `get_results_scalar`; `get_results_scenario`; `get_results_timeseries`

Examples

```
## Not run:
## Put this R script in a folder which contains the scenario folders, then run
## the code below.

## Exploring ss3 model results
library(ggplot2)
library(ss3sim)

## This function reads in results for all runs in a particular directory
get_results_all(overwrite_files=FALSE)

## Read in the final results produced by above function
scalars <- read.csv("ss3sim_scalar.csv")
ts <- read.csv("ss3sim_ts.csv")

## NOTE: For my case I had run different F cases (F0,F1, F2) for the base
## case. Thus below I've grouped by F. You might want to group by something
## else, such as M, D, E, etc. depending on what you've run.

## Check convergence
g <- ggplot(scalars)
round(with(scalars, tapply(max_grad, species, mean)),3)
round(with(scalars, tapply(max_grad, species, median)),3)
## Plot w/ free ylim to see differences
g+geom_boxplot(aes(F,max_grad))+facet_grid(species~., scales="free")

## Calculate and plot a metric, e.g. relative error of SSB_MSY
scalars <- transform(scalars,
                     SSB_MSY=(SSB_MSY_em-SSB_MSY_om)/SSB_MSY_om)
g <- ggplot(scalars)
g+geom_boxplot(aes(x=FALSE,y=SSB_MSY))+facet_grid(species~.)

## steepness
scalars <- transform(scalars,
               SR_BH_steep=(SR_BH_steep_om-SR_BH_steep_em)/SR_BH_steep_om)
g <- ggplot(scalars)
g+geom_boxplot(aes(x=FALSE,y=SR_BH_steep))+facet_grid(species~.)

## SSB unfished
scalars <- transform(scalars,
         SSB_Unfished=(SSB_Unfished_om-SSB_Unfished_em)/SSB_Unfished_om)
g <- ggplot(scalars)
g+geom_boxplot(aes(x=FALSE,y=SSB_Unfished))+facet_grid(species~.)

##  log(R0)
scalars <- transform(scalars,
                     SR_LN_R0=(SR_LN_R0_om-SR_LN_R0_em)/SR_LN_R0_om)
g <- ggplot(scalars)
g+geom_boxplot(aes(F,SR_LN_R0))+facet_grid(species~.)

## Make some timeseries plots
```

```
##  Plot error in relative biomass by year
ts <- transform(ts, SpawnBio=(SpawnBio_em-SpawnBio_om)/SpawnBio_om)
g <- ggplot(ts, aes(x=year))+ ylab("Relative bias in biomass") + xlab("Year")
g+geom_jitter(aes(y=SpawnBio, group=replicate), size=.1, alpha=.3)+
    geom_smooth(method="loess",aes(y=SpawnBio), color="red") +
    facet_grid(species~., )+
    geom_hline(yintercept = 0, lty = 2)

## Look at recruitment
ts <- transform(ts, Recruit_0=(Recruit_0_em-Recruit_0_om)/Recruit_0_om)
g <- ggplot(ts, aes(x=year))+ ylab("Relative bias in recruitment") +
xlab("Year")
g+geom_jitter(aes(y=Recruit_0), size=.1, alpha=.3)+
    geom_smooth(method="loess",aes(y=Recruit_0), color="red") +
    facet_grid(species~., )+
    geom_hline(yintercept = 0, lty = 2)

## End(Not run)
```

`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

See Also

Other get.results: `get_results_all`; `get_results_scenario`; `get_results_timeseries`

get_results_scenario *Extract SS3 simulation results for one scenario.*

Description

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

Usage

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

Arguments

`scenario`	A single character giving the scenario from which to extract results.
`directory`	The directory which contains the scenario folder.
`overwrite_files`	A boolean (default is FALSE) for whether to delete any files previously created with this function. This is intended to be used if replicates were added since the last time it was called, or any changes were made to this function.

Author(s)

Cole Monnahan

See Also

Other get.results: `get_results_all`; `get_results_scalar`; `get_results_timeseries`

Examples

```
## Not run:
d <- system.file("extdata", package = "ss3sim")
case_folder <- paste0(d, "/eg-cases")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
run_ss3sim(iterations = 1:2, scenarios =
  c("D0-E0-F0-G0-R0-S0-M0-cod"),
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = FALSE)
get_results_scenario(c("D0-E0-F0-G0-R0-S0-M0-cod"))

## End(Not run)
```

`get_results_timeseries` *Extract time series from a model run.*

Description

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

Usage

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

Arguments

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

Author(s)

Cole Monnahan

See Also

Other get.results: `get_results_all`; `get_results_scalar`; `get_results_scenario`

`get_sigmar` *Get recruitment deviation sigma*

Description

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

Usage

```
get_sigmar(om)
```

Arguments

`om`	The name of the operating model, which should be the prefix of the `.ctl` file, eg. "myOM".

Author(s)

Kelli Johnson

is_f *Check if a string is a function*

Description

This function takes a character object and checks if it looks like an R function.

Usage

```
is_f(x)
```

Arguments

x	A character object

pastef *Paste with "/" as the separator*

Description

Paste with "/" as the separator

Usage

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

Arguments

...	Objects to paste together

run_bias_ss3 *Determine level of bias adjustment for SS3 runs*

Description

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

Usage

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

Arguments

dir	Folder for all of the bias adjustment runs (e.g. "M1-F1-D1-R1-cod/bias" which must contain numbered folders for the nsim runs, e.g. "M1-F1-D1-R1-cod/bias/1/", "M1-F1-D1-R1-cod/bias/2/", ..., "M1-F1-D1-R1-cod/bias/10/" if there are nsim = 10 bias adjustment runs)
outdir	Folder containing the run folders for a given scenario (e.g. "M1-F1-D1-R1-cod" that contains "M1-F1-D1-R1-cod/1/" "M1-F1-D1-R1-cod/2/", etc.)
nsim	number of bias adjustment runs conducted for a particular scenario (e.g. 10)
conv_crit	The maximum percentage of bias iterations that can produce a non-invertible Hessian before a warning will be produced. If this percentage is exceeded then a file WARNINGS.txt will be produced. Currently, the simulations will continue to run.

Details

This function:

- uses the **r4ss** package to read in output from n SS3 runs,
- uses Ian Taylor's **r4ss** function to find values for the n bias adjustment parameters for each run,
- takes the average over runs for each bias adjustment parameter
- writes out the unaveraged and averaged (AdjustBias.DAT and AvgBias.DAT, respectively) bias adjustment parameters to the dir folder
- takes a control.ss_new file from one of the n SS runs, changes the n bias adjustment parameters, and writes the whole updated control.ss_new file with new bias adjustment parameters to an em.ctl file

Author(s)

Carey McGilliard

References

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

See Also

run_ss3sim, ss3sim_base, run_ss3model, bias_ss3

Examples

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

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

```
case_folder <- paste0(d, "/eg-cases")
om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")
wd <- getwd()
setwd(temp_path)
# (Note that bias_nsim should be bigger, say 10, but it is set to 2
# here so the example runs faster.)
run_ss3sim(iterations = 1:1, scenarios = "D1-E0-F0-R0-M0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = TRUE, bias_nsim = 2)
setwd(wd)

## End(Not run)
```

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

Description

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

Usage

```
run_ss3model(scenarios, iterations, type = c("om", "em"), ss3path = NULL,
  admb_options = "", hess = FALSE, ignore.stdout = TRUE,
  admb_pause = 0.05, ...)
```

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?
`ss3path`	The path to your SS3 binary the binary is not in your path. For example, if `SS3` was in the folder `/usr/bin/` then `ss3path = "/usr/bin/"`. Make sure to append a slash to the end of this path. Defaults to `NULL`, which means the function will assume the binary is already in your path. See the vignette for details. `vignette("ss3sim-vignette")`
`hess`	Calculate the Hessian on estimation model runs?
`admb_options`	Any additional options to pass to the SS3 command.
`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.

... — Anything else to pass to `system`.

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(M = "M", F = "F", D = c("index", "lcomp", "agecomp"), R =
  "R", E = "E"), user_recdevs = NULL, parallel = FALSE, ...)
```

Arguments

iterations — Which iterations to run. A numeric vector. For example `1:100`.

scenarios — Which scenarios to run. A vector of character objects. For example `c("D0-E0-F0-R0-M0-cod", "D1-E0-`
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. If you are passing time-varying parameters beyond (or instead of) natural mortality (M), then you will need to adjust these values to reflect your scenarios. This argument is passed to `get_caseargs`. See that function for details and examples of how to specify this.

`user_recdevs` An optional matrix of recruitment deviations to replace the recruitment deviations built into the package. The columns represent run iterations and the rows represent years. `user_recdevs` can be a matrix of 0s for deterministic model checking. For traditional stochastic simulations these would be independent and normally distributed deviations with a standard deviation equal to the desired sigma R. Note that these recruitment deviations will be used verbatim (after exponentiation). `user_recdevs` will *not* be multiplied by sigma R and they will *not* be log-normal bias corrected. If `user_recdevs` are specified as anything besides `NULL` the package will issue a warning about this. Biased recruitment deviations can lead to biased model results.

`parallel` A logical argument that controls whether the scenarios are run in parallel. You will need to register multiple cores first with a package such as **doParallel** and have the **foreach** package installed. See the example below.

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

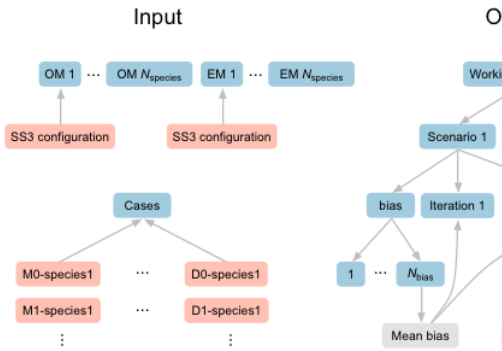
Details

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

Value

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

- `D0-E0-F0-M0-R0-cod/bias/1/om`
- `D0-E0-F0-M0-R0-cod/bias/1/em`
- `D0-E0-F0-M0-R0-cod/bias/2/om`
- ...
- `D0-E0-F0-M0-R0-cod/1/om`
- `D0-E0-F0-M0-R0-cod/1/em`
- `D0-E0-F0-M0-R0-cod/2/om`
- ...



An illustration of the input and output file structure of an **ss3sim** simulation:

Author(s)

Sean C. Anderson

See Also

ss3sim_base, run_ss3model, run_bias_ss3, get_caseargs, expand_scenarios

Examples

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

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

# Without bias adjustment:
run_ss3sim(iterations = 1:1, scenarios = "D0-E0-F0-R0-M0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em)
unlink("D0-E0-F0-R0-M0-cod", recursive = TRUE) # clean up

# An example specifying the case files:
run_ss3sim(iterations = 1:1, scenarios = "D0-E0-F0-R0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  case_files = list(F = "F", D = c("index", "lcomp",
      "agecomp"), R = "R", E = "E"))
unlink("D0-E0-F0-R0-cod", recursive = TRUE) # clean up

# With bias adjustment:
# (Note that bias_nsim should be bigger, say 5 or 10, but it is set
# to 2 here so the example runs faster.)
run_ss3sim(iterations = 1:1, scenarios = "D1-E0-F0-R0-M0-cod",
```

```
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = TRUE, bias_nsim = 2)

# Restarting the previous run using the existing bias-adjustment
# output
run_ss3sim(iterations = 2:3, scenarios = "D1-E0-F0-R0-M0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = FALSE, bias_already_run = TRUE)
unlink("D1-E0-F0-R0-M0-cod", recursive = TRUE) # clean up

# A run with deterministic process error for model checking:
recdevs_det <- matrix(0, nrow = 100, ncol = 20)
run_ss3sim(iterations = 1:20, scenarios = "D0-E100-F0-R0-M0-cod",
  case_folder = case_folder, om_dir = om, em_dir = em,
  bias_adjust = TRUE, bias_nsim = 2, user_recdevs = recdevs_det)
unlink("D0-E100-F0-R0-M0-cod", recursive = TRUE) # clean up

# 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
run_ss3sim(iterations = 1, scenarios = c("D0-E0-F0-R0-M0-cod",
    "D1-E0-F0-R0-M0-cod"), case_folder = case_folder,
  om_dir = om, em_dir = em, parallel = TRUE)
unlink("D0-E0-F0-R0-M0-cod", recursive = TRUE) # clean up
unlink("D1-E0-F0-R0-M0-cod", recursive = TRUE) # clean up

# Return to original working directory:
setwd(wd)

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Author(s)

Sean C. Anderson

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

Description

Example scalar data from the ss3sim vignette

`setup_parallel` *Setup parallel processing*

Description

Setup parallel processing

Usage

```
setup_parallel()
```

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

Description

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

Details

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

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

To carry out **ss3sim** simulations, 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 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.
- change_lcomp: Controls how length composition data are sampled.
- change_agecomp: Controls how age composition data are sampled.
- change_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.

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 package vignette `vignette("ss3sim-vignette")` 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 fun 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, estim_params, tv_params, om_dir, em_dir,
  retro_params = NULL, user_recdevs = NULL, bias_adjust = FALSE,
  bias_nsim = 5, bias_already_run = FALSE, hess_always = FALSE,
  print_logfile = TRUE, sleep = 0, conv_crit = 0.2, seed = 21, ...)
```

Arguments

iterations	Which iterations to run. A numeric vector.
scenarios	Which scenarios to run.
tv_params	A named list containing all the change_tv (time-varying) options.
f_params	A named list containing all the change_f options.
index_params	A named list containing all the change_index options.
lcomp_params	A named list containing all the change_lcomp options.
agecomp_params	A named list containing all the change_agecomp options.
retro_params	A named list containing all the change_retro options.
estim_params	A named list containing all the change_e options.
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.
user_recdevs	An optional matrix of recruitment deviations to replace the recruitment deviations built into the package. The columns represent run iterations and the rows represent years. user_recdevs can be a matrix of 0s for deterministic model checking. For traditional stochastic simulations these would be independent and normally distributed deviations with a standard deviation equal to the desired sigma R. Note that these recruitment deviations will be used verbatim (after exponentiation). user_recdevs will *not* be multiplied by sigma R and they will *not* be log-normal bias corrected. If user_recdevs are specified as anything besides NULL the package will issue a warning about this. Biased recruitment deviations can lead to biased model results.

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

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

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

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

`print_logfile` Logical. Print a log file?

`sleep` A time interval (in seconds) to pause on each iteration. Useful if you want to reduce average CPU time – perhaps because you're working on a shared server.

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

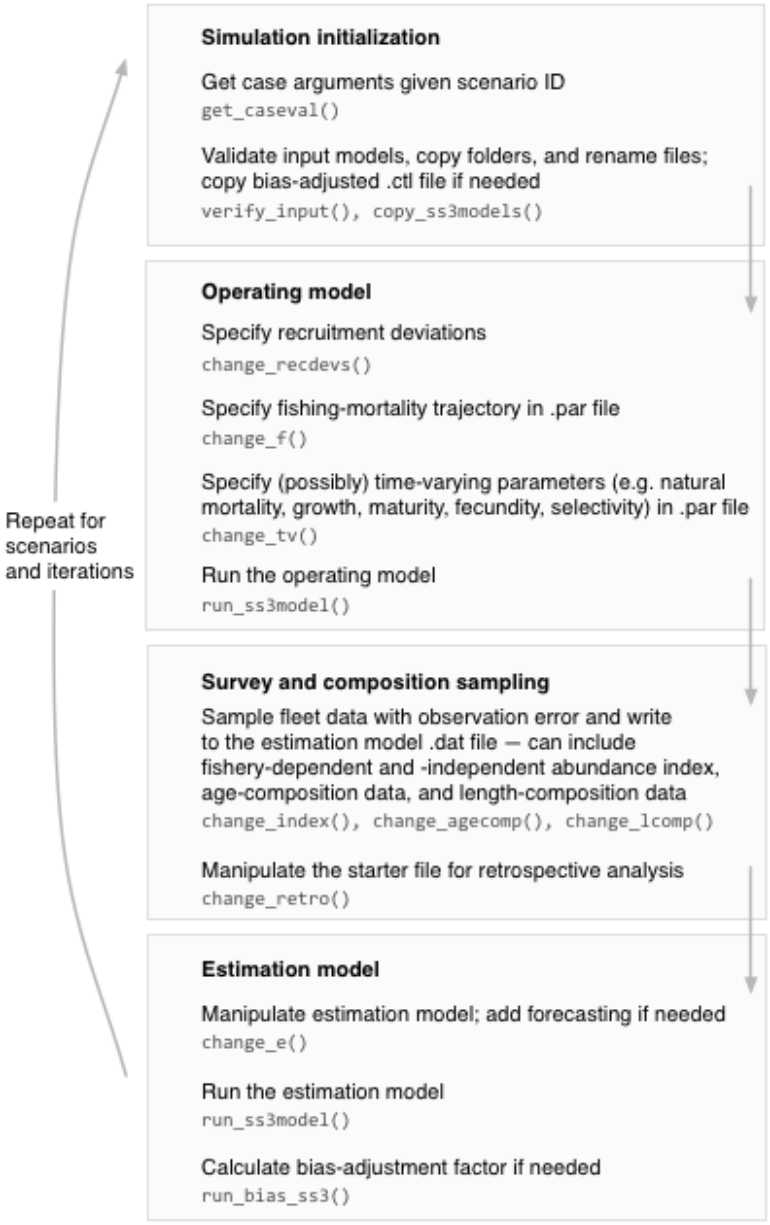
`seed` The seed value to pass to `get_recdevs` when generating recruitment deviations. The generated recruitment deviations depend on the iteration value, but also on the value of `seed`. A given combinatination of iteration, number of years, and `seed` value will result in the same recruitment deviations.

`...` Anything extra to pass to `run_ss3model`. For example, you may want to pass additional options to `SS3` through the argument `admb_options`. Anything that doesn't match a named argument in `run_ss3model` will be passed to the `system` call that runs `SS3`. If you are on a Windows computer then you might want to pass `show.output.on.console = FALSE` to make the simulations runs faster by not printing output to the console.

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`.

The steps carried out within `ss3sim_base`:



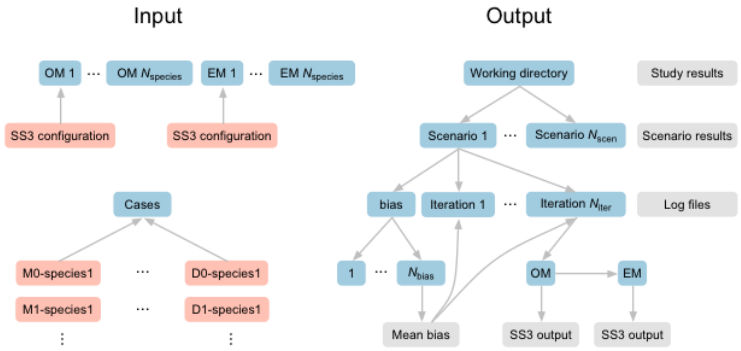
Value

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-E0-F0-M0-R0-cod/bias/1/om`
- `D0-E0-F0-M0-R0-cod/bias/1/em`
- `D0-E0-F0-M0-R0-cod/bias/2/om`
- ...

- D0-E0-F0-M0-R0-cod/1/om
- D0-E0-F0-M0-R0-cod/1/em
- D0-E0-F0-M0-R0-cod/2/om
- ...

The input and output file structure of an **ss3sim** simulation:



Author(s)

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

See Also

run_ss3sim

Examples

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

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

# Pull in file paths from the package example data:
d <- system.file("extdata", package = "ss3sim")
om_dir <- paste0(d, "/models/cod-om")
em_dir <- paste0(d, "/models/cod-em")
a <- get_caseargs(folder = paste0(d, "/eg-cases"), scenario =
"M0-F0-D0-R0-E0-cod")

ss3sim_base(iterations = 1, scenarios = "M0-F0-D0-R0-E0-cod",
```

```
f_params = a$F, index_params = a$index, lcomp_params = a$lcomp,
agecomp_params = a$agecomp, tv_params = a$tv_params, retro_params =
a$R, estim_params = a$E, om_dir = om_dir, em_dir
= em_dir)
unlink("M0-F0-D0-R0-E0-cod", recursive = TRUE) # clean up

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

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

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

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

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

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

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

R0 <- list(retro_yr = 0)

ss3sim_base(iterations = 1:20, scenarios = "D1-E0-F0-R0-M0-cod",
  f_params = F0, index_params = index1, lcomp_params = lcomp1,
  agecomp_params = agecomp1, estim_params = E0, tv_params = M0,
  retro_params = R0, om_dir = om, em_dir = em)

unlink("D1-E0-F0-R0-M0-cod", recursive = TRUE) # clean up

setwd(wd)

## End(Not run)
```

`substr_r` *Substring from right*

Description

Substring from right

Usage

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

Arguments

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

References

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

---

ts_dat *Example time series data from the ss3sim vignette*

Description

Example time series data from the ss3sim vignette

---

verify_input *Verify and standardize SS3 input files*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Curry James Cunningham; modified by Sean Anderson

Examples

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

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

om <- paste0(d, "/models/cod-om")
em <- paste0(d, "/models/cod-em")

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

# Verify the correct files exist and change file names:
verify_input(model_dir = paste0(temp_path, "/cod-om"), type = "om")
verify_input(model_dir = paste0(temp_path, "/cod-em"), type = "em")
```

Index