Package: FLfse (via r-universe)

November 23, 2026

Type Package

Title Fisheries Sampling Evaluation In FLR

Version 1.0.4

Maintainer Simon Fischer <simon.fischer@cefas.co.uk>

Description Fisheries Sampling Evaluation in FLR. A package to interface betwen the SAM stock assessments and FLR, and aid the inclusion of SAM into Management Strategy Evaluation (MSE), including the set up of operating models mimicking SAM.

License GPL-3

Encoding UTF-8

LazyData true

Imports methods, foreach, stats, utils, FLCore (>= 2.6.5)

Suggests spict (>= 1.2.1), stockassessment (>= 0.9.0), TMB, mgcv, doParallel

RoxygenNote 7.3.3

Repository https://ices-tools-dev.r-universe.dev

Date/Publication 2026-07-07 09:32:33 UTC

RemoteUrl https://github.com/flr/FLfse

RemoteRef HEAD

RemoteSha 59396ceabf1c533b7c5c1723dc0b0187c0064ab9

RemoteSubdir FLfse

Contents

cod4_stk 2
FLR_SAM 3
FLR_SPiCT 6
getpars 7
had4_stk 8
her4_stk 9

mac.27.nea_stk_2019 10
ple7a_stk 11
SAM_uncertainty 11
SAM2FLStock 13
whg4_stk 16

Index **18**

`cod4_stk` *North Sea cod*

Description

Cod (*Gadus morhua*) in Subarea 4, Division 7.d, and Subdivision 20 (North Sea, eastern English Channel, Skagerrak). Assessment input data for North Sea cod as used by ICES WGNSSK 2018 and 2019. Includes the stock, the indices and the SAM configuration. See the example(s) below for how to run the assessment.

Usage

```
cod4_stk

cod4_stk_2019

cod4_stk_2019

cod4_idx

cod4_idx_2019

cod4_conf_sam
```

Format

An object of class `FLStock` of dimension 6 x 56 x 1 x 1 x 1 x 1.

An object of class `FLStock` of dimension 6 x 57 x 1 x 1 x 1 x 1.

An object of class `FLStock` of dimension 6 x 57 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 2.

An object of class `FLIndices` of length 2.

An object of class `list` of length 20.

Source

https://www.stockassessment.org

Examples

```
# Replicate the 2018 assessment:
fit <- FLR_SAM(stk = cod4_stk,
               idx = cod4_idx,
               conf = cod4_conf_sam)

# Replicate the 2019 assessment:
fit <- FLR_SAM(stk = cod4_stk_2019,
               idx = cod4_idx_2019,
               conf = cod4_conf_sam)
```

FLR_SAM *Run SAM with FLR objects.*

Description

This function runs a SAM assessment using FLR objects as input. Stock and fishery data is extracted from an object of class FLStock, survey indices from FLIndex or FLIndices. Additional model configurations can be passed to SAM.

Usage

```
FLR_SAM(
  stk,
  idx,
  conf = NULL,
  conf_full = FALSE,
  par_ini = NULL,
  DoParallel = FALSE,
  NA_rm = TRUE,
  idx_weight = FALSE,
  ...
)

## S4 method for signature 'FLStock,FLIndices'
FLR_SAM(
  stk,
  idx,
  conf = NULL,
  conf_full = FALSE,
  par_ini = NULL,
  DoParallel = FALSE,
  NA_rm = TRUE,
  idx_weight = FALSE,
  ...
)
```

```
## S4 method for signature 'FLStock,FLIndex'
FLR_SAM(
  stk,
  idx,
  conf = NULL,
  conf_full = FALSE,
  par_ini = NULL,
  DoParallel = FALSE,
  NA_rm = TRUE,
  idx_weight = FALSE,
  ...
)
```

Arguments

stk	Object of class FLStock with stock and fishery data.
idx	Object of class FLIndices or FLIndex object with survey index time series.
conf	Optional configurations passed to SAM. Defaults to NULL. If provided, should be a list.
conf_full	Use provided configuration object in full without ANY checking (see Details for more information).
par_ini	Optional starting parameters for SAM. See details for more information.
DoParallel	Optional, defaults to FALSE. If set to TRUE, will perform iterations of stock in parallel. See Details below for description.
NA_rm	Remove trailing years with NAs, defaults to TRUE.
idx_weight	Use index weights (index variance)? Defaults to FALSE. If required, should be slot of idx.
...	Additional arguments passed to sam.fit(), e.g. newtonsteps

Details

Stock and fishery data is extracted from stk, survey indices from idx. Stock data used are:

- catch numbers at age (catch.n slot from stk)
- maturity ogive (mat slot from stk)
- stock weight at age (stock.wt slot from stk)
- catch weight at age (catch.wt slot from stk)
- discards weigth at age (discards.wt slot from stk)
- landings weight at age (landings.wt slot from stk)
- natural mortality at age (m slot from stk)
- proportion of fishing mortality before spawning at age (harvest.spwn slot from stk)
- proportion of natural mortality before spawning at age (catch.n slot from stk)

- landing fraction at age (calculated as landings numbers at age divided by the catch numbers at age)

Trailing years without data in `stk` are removed, unless turned of by setting `NA_rm = FALSE`.

Survey indices are extracted from `idx`, using the `index` slot(s). SSB indices can be used. To define an index as SSB index, the first (age) dimension of the `index` slot of `idx` has to be of length 1 and the name of this dimension can be either missing (`NA`), non-numeric (e.g. "ssb") or `-1`.

Additional configurations can be passed as a list to SAM with the `conf` argument. If argument `conf_full` is set to `TRUE`, the configuration is passed straight on to SAM without any checking. If argument `conf_full` is set to `FALSE` (default), then `FLR_SAM` first generates a default model configuration with `stockassessment`'s `setup.sam.data` and additional configurations available in `conf` replace default configurations. For details about possible configurations and format see `'help("defcon", package = "stockassessment")` and https://github.com/fishfollower/SAM.

The function can handle input objects with multiple iterations (`iter` dimension in `stk` and `idx`). If multiple iterations are provided for `stk` but not for `idx`, `idx` will be inflated, and vice versa. If the assessment fails for some iterations, the error messages are returned for these iterations. If argument `DoParallel` is set to `TRUE`, the individual iterations are processed in parallel. This uses parallel computing provided by the package `DoParallel`. The parallel workers need to be set up and registered before calling the function. See `?DoParallel`.

Argument `par_ini` allows the provision of initial parameter values for SAM and can speed up the model. Either a single set of parameters can be supplied and they are recycled if neccessary. Alternatively, a list of initial parameters can be supplied, one for each iteration of the stock/index. If the dimensions of initial values for numbers/fishing mortality at age differ from the data, redundant years are automatically removed and if years are missing, the values from the last provided year are recycled.

The default console output generated by SAM is not printed but saved. It is stored as an attribute of the fit and can be accessed with `attr(fit, "messages")`.

Tagging data can be provided in the usual SAM format and should be stored as an attribute of `attr(catch.n(stk), "recap")`. Additional tagging configurations can be supplied as a list with the attribute `attr(catch.n(stk), "recap_conf")`, e.g. `list(map = list(logitRecapturePhi = factor(c(1, 1))))`, which are then passed on to `sam.fit()`.

Weights for the catch numbers can be supplied as an attribute of `attr(catch.n(stk), "weight")` and should be formatted as `FLQuant` objects.

Value

An object of class `sam` (for single iteration) or `sam_list` (list of `sam` objects for multiple iterations) with the model results.

Warning

This methods requires the `stockassessment` package and all its dependencies to be installed. For details how to obtain `stockassessment`, see https://github.com/fishfollower/SAM/.

Examples

```
# fit SAM to North Sea cod
```

```
fit <- FLR_SAM(stk = cod4_stk, idx = cod4_idx)

# use WGNSSK 2017 configuration
fit <- FLR_SAM(stk = cod4_stk, idx = cod4_idx, conf = cod4_conf_sam)

# fit SAM to Irish Sea plaice
fit <- FLR_SAM(stk = ple7a_stk, idx = ple7a_idx, conf = ple7a_conf_sam)
```

FLR_SPiCT *Run SPiCT with FLR objects.*

Description

This function extracts the catch from an FLStock object and the survey indices/index from a FLIndices or FLIndex object and runs a SPiCT (Surplus Production in Continuous Time) stock assessment.

Usage

```
FLR_SPiCT(stk, idx, conf = NULL)

## S4 method for signature 'FLStock,FLIndices'
FLR_SPiCT(stk, idx, conf = NULL)

## S4 method for signature 'FLStock,FLIndex'
FLR_SPiCT(stk, idx, conf = NULL)
```

Arguments

stk	Object of class FLStock with catch time series.
idx	Object of class FLIndices or FLIndex object with survey index time series.
conf	Optional configurations passed to SPiCT. Should be a list.

Details

The catch time series is obtained from the catch slot of stk.

The survey index/indices are obtained from the index slot(s) of idx. If the slot(s) contains an age age structure, the sum over all ages is used.

Additional configurations can be passed as a list to SPiCT with the conf argument. They are passed directly to SPiCT (fit.spict) without checking. Any configurations accepted by (fit.spict) can be used.

Value

An object of class spictcls with the model results.

Warning

This methods requires the spict package and all its dependencies to be installed. For details how to obtain spict, see https://github.com/mawp/spict/.

Examples

```
# fit SPiCT to Irish Sea plaice
fit <- FLR_SPiCT(stk = ple7a_stk, idx = ple7a_idx)
fit

# pass additional configuration, set time step to 1 per year
conf <- list(dteuler = 1)
fit <- FLR_SPiCT(stk = ple7a_stk, idx = ple7a_idx, conf = conf)
fit
```

---

getpars *Get parameter estimates from SAM model fit.*

Description

This function extracts the parameter estimates from a SAM model fit. These are useful e.g. as initial values in subsequent model fits and can improve model convergence/computing time.

Usage

```
getpars(fit)

## S4 method for signature 'sam'
getpars(fit)

## S4 method for signature 'sam_list'
getpars(fit)
```

Arguments

fit	A single SAM model fit of class sam or a list of fits of class sam_lst.

Value

A list with the model parameters of the SAM or a list of them in case of several supplied models

Examples

```
### fit SAM to North Sea cod
fit <- FLR_SAM(stk = cod4_stk, idx = cod4_idx, conf = cod4_conf_sam)

### extract parameters
pars <- getpars(fit)

### use them as starting values
fit2 <- FLR_SAM(stk = cod4_stk, idx = cod4_idx, conf = cod4_conf_sam,
                par_ini = pars)
```

---

had4_stk *North Sea haddock*

Description

Haddock (*Melanogrammus aeglefinus*) in Subarea 4, Division 6.a, and Subdivision 20 (North Sea, West of Scotland, Skagerrak). Assessment input data for North Sea haddock as used by ICES WGNSSK 2018. Includes the stock, the indices and the SAM configuration. See the example below for how to run the assessment.

Usage

```
had4_stk

had4_idx

had4_conf_sam
```

Format

An object of class `FLStock` of dimension 9 x 54 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 2.

An object of class `list` of length 20.

Source

https://www.stockassessment.org

Examples

```
# Replicate the 2018 assessment:
fit <- FLR_SAM(stk = had4_stk,
               idx = had4_idx,
               conf = had4_stk)
```

her4_stk *North Sea herring*

Description

Herring (Clupea harengus) in Subarea 4 and divisions 3.a and 7.d, autumn spawners (North Sea, Skagerrak and Kattegat, eastern English Channel). Assessment input data for North Sea herring as used by ICES HAWG 2019. Includes the stock, the indices and the SAM configuration.

Usage

```
her4_stk

her4_idx

her4_conf_sam
```

Format

An object of class `FLStock` of dimension 9 x 73 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 8.

An object of class `list` of length 21.

Details

Please note that the herring SAM assessment requires a different version of the stockassessment package. For details, see the readme at https://github.com/shfischer/FLfse

Source

https://github.com/ices-eg/wg_HAWG/tree/master/NSAS

Examples

```
# NOTE: the herring SAM assessment requires a different version of
# the stockassessment package
## Not run:
fit <- FLR_SAM(stk = her4_stk,
               idx = her4_idx,
               conf = her4_conf_sam)

## End(Not run)
```

`mac.27.nea_stk_2019` *Northeast Atlantic mackerel 2019*

Description

Mackerel (*Scomber scombrus*) in subareas 1-8 and 14, and in Division 9.a (the Northeast Atlantic and adjacent waters). Assessment input data for Northeast Atlantic mackerel as used by ICES WGWIDE 2019. Includes the stock, the indices and the SAM configuration. See the example below for how to run the assessment.

Usage

```
mac.27.nea_stk_2019

mac.27.nea_idx_2019

mac.27.nea_conf_2019
```

Format

An object of class `FLStock` of dimension 13 x 40 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 3.

An object of class `list` of length 23.

Details

`mac.27.nea_stk_2019` includes tagging data, stored as an attribute in `attr(catch.n(mac.27.nea_stk_2019), "recap")`. An additional tagging configuration is stored in `attr(catch.n(mac.27.nea_stk_2019), "recap_conf")`. Also, weights for the catch numbers are supplied as an attribute in `attr(catch.n(mac.27.nea_stk_2019) "weight")`. All additional data is passed on automatically to the SAM assessment.

Source

https://www.stockassessment.org

Examples

```
# The 2019 WGWIDE Northeast Atlantic mackerel assessment can be reproduced
# with:
fit <- FLR_SAM(stk = mac.27.nea_stk_2019,
               idx = mac.27.nea_idx_2019,
               conf = mac.27.nea_conf_2019)
```

ple7a_stk *Irish Sea plaice*

Description

Plaice (*Pleuronectces platessa*) in Division 7.a (Irish Sea). Assessment input data for Irish Sea plaice as used by ICES WGCSE 2017 and 2019. Includes the stock, the indices and the SAM configuration. See the example below for how to run the assessment.

Usage

```
ple7a_stk

ple7a_idx

ple7a_stk_2019

ple7a_idx_2019

ple7a_conf_sam
```

Format

An object of class `FLStock` of dimension 8 x 36 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 3.

An object of class `FLStock` of dimension 8 x 38 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 3.

An object of class `list` of length 3.

Examples

```
# Replicate the 2019 assessment:
fit <- FLR_SAM(stk = ple7a_stk_2019,
               idx = ple7a_idx_2019,
               conf = ple7a_conf_sam)
```

SAM_uncertainty *Create replicates/iterations of SAM model fit based on variance-covariance matrix*

Description

This function use the uncertainty estimated by SAM to create replicates/iterations of assessment results. The function uses the variance-covariance matrix to quantify uncertainty.

Usage

```
SAM_uncertainty(
  fit,
  n = 1000,
  print_screen = FALSE,
  seed = NULL,
  idx = TRUE,
  idx_cov = TRUE,
  catch_est = TRUE,
  proc_error_est = TRUE
)

## S4 method for signature 'sam'
SAM_uncertainty(
  fit,
  n = 1000,
  print_screen = FALSE,
  seed = NULL,
  idx = TRUE,
  idx_cov = TRUE,
  catch_est = TRUE,
  proc_error_est = TRUE
)
```

Arguments

fit	A SAM model fit object of class sam.
n	Number of replicates
print_screen	If set to TRUE, print output of TMB::sdreport to screen.
seed	Random number seed for reproducibility.
idx	If set to TRUE (default) metrics relating to the indices are returned.
idx_cov	If set to TRUE, return covariance of survey index/indices.
catch_est	If set to TRUE, return catch estimates from SAM.
proc_error_est	If set to TRUE, return (survival) process error level from SAM.

Details

The returned objects are FLQuants where the iteration dimension contains the replicates. Each replicate is internally consistent, e.g. the fishing mortality matches the stock numbers of the same replicate.

The following metrics are returned:

- stock.n Stock numbers at age for all years, FLQuant
- harvest Fishing mortalities at age for all years, FLQuant

- catch.n Estimates of catch numbers at age for all years. This differs from the assessment input values. If the SAM model fit contains catch multipliers, the values returned here are corrected for this. Class FLQuant.
- catch_sd Standard deviation of the catch numbers at age, time invariant, FLQuant
- survey_catchability Catchability at age for all years for all survey indices, list of FLQuants
- survey_sd Standard deviation of all surveys at age, time invariant, list of FLQuants,
- survey_cov Covariance matrices of survey ages, one for each survey. Return object is a list of lists, the first level corresponds to the replicates, the second level to the surveys. If no covariance between ages is assumed in the SAM model, the diagonal in the covariance matrices is simply the square root of the standard deviation (in survey_sd)
- proc_error Standard deviation of the stock numbers at age, time invariant, class FLQuant. This corresponds to the survival process error assumed/estimated, i.e. quantifies how much the actual stock numbers at age deviate from the deterministic catch equation.

Value

A list of FLQuants with the elements: stock.n, harvest, catch.n, catch_sd, survey_catchability, survey_sd, survey_cov, proc_error.

Examples

```
### fit SAM to North Sea cod
fit <- FLR_SAM(stk = cod4_stk, idx = cod4_idx, conf = cod4_conf_sam)

### create 2 replicates
reps <- SAM_uncertainty(fit, n = 2)
```

SAM2FLStock *Coerce SAM output into* FLStock *object*

Description

This function takes the output from running the SAM stockassessment and converts them into an FLStock object.

Usage

```
SAM2FLStock(
  object,
  stk,
  uncertainty = FALSE,
  conf_level = 95,
  stock_only = FALSE,
  catch_estimate = FALSE,
  correct_catch = FALSE,
  mat_est = FALSE,
```

```
  stock.wt_est = FALSE,
  catch.wt_est = FALSE,
  m_est = FALSE,
  spinoutyear = FALSE
)

## S4 method for signature 'sam,missing'
SAM2FLStock(
  object,
  stk,
  uncertainty = FALSE,
  conf_level = 95,
  stock_only = FALSE,
  catch_estimate = FALSE,
  correct_catch = FALSE,
  mat_est = FALSE,
  stock.wt_est = FALSE,
  catch.wt_est = FALSE,
  m_est = FALSE,
  spinoutyear = FALSE
)

## S4 method for signature 'sam_list,missing'
SAM2FLStock(
  object,
  stk,
  uncertainty = FALSE,
  conf_level = 95,
  stock_only = FALSE,
  catch_estimate = FALSE,
  correct_catch = FALSE,
  mat_est = FALSE,
  stock.wt_est = FALSE,
  catch.wt_est = FALSE,
  m_est = FALSE,
  spinoutyear = FALSE
)

## S4 method for signature 'sam,FLStock'
SAM2FLStock(
  object,
  stk,
  uncertainty = FALSE,
  conf_level = 95,
  stock_only = FALSE,
  catch_estimate = FALSE,
  correct_catch = FALSE,
  mat_est = FALSE,
```

```
  stock.wt_est = FALSE,
  catch.wt_est = FALSE,
  m_est = FALSE,
  spinoutyear = FALSE
)

## S4 method for signature 'sam_list,FLStock'
SAM2FLStock(
  object,
  stk,
  uncertainty = FALSE,
  conf_level = 95,
  stock_only = FALSE,
  catch_estimate = FALSE,
  correct_catch = FALSE,
  mat_est = FALSE,
  stock.wt_est = FALSE,
  catch.wt_est = FALSE,
  m_est = FALSE,
  spinoutyear = FALSE
)
```

Arguments

object	Object of class sam with the results from a SAM stock assessment run. Alternatively, object of class sam_list, i.e. a list of sam objects.
stk	Optional. Object of class FLStock, to which the assessment results are added.
uncertainty	If set to TRUE, the estimated uncertainty from SAM will be added as attribute.
conf_level	Confidence level used when uncertainty is returned. Defaults to 95 (percent).
stock_only	Logical. If set to TRUE, catch data (numbers, weights) are ignored and only stock data (numbers, SSB, etc.) are returned
catch_estimate	Logical, return the catch estimated by SAM instead of the model input?
correct_catch	Logical, correct catch with catch multiplier estimated by SAM?
mat_est	Logical, return SAM estimates for maturity?
stock.wt_est	Logical, return SAM estimates for stock weights?
catch.wt_est	Logical, return SAM estimates for catch weights?
m_est	Logical, return SAM estimates for natural mortality?
spinoutyear	Logical, return SAM estimates of biological estimates beyond last data year?

Details

SAM2FLStock returns both the input data used for running SAM (e.g. catch) and the model estimates (stock numbers and fishing mortality). By default, the returned catch is the input catch provided to SAM. However, the catch as estimated by SAM can be returned by setting catch_estimate = TRUE. Also, estimates of biological data (stock weights, catch weights, natural mortality, maturity) can be returned if requested and available from the model fit. Setting uncertainty = TRUE

returns confidence intervals for catch, stock numbers and fishing mortality, saved as attributes in the corresponding slots of the FLStock output.

If an FLStock is provided as stk argument, then this is used as template. If the dimensions (years, iterations) differ between the SAM results and the provided stock template, the returned FLStock is expanded.

The object argument can either be a single SAM model fit or a list of SAM model fits (defined as class sam_list). If a list is provided, the output is an FLStock object where the different iterations correspond to the individual model fits.

The function can handle SAM model fits with multiple fleets. In the returned FLStock, the fleets are combined into a single fleet. Some functionality (e.g. uncertainty bounds) might not work for multiple fleets.

Value

An object of class FLStock.

Examples

```
# fit SAM to North Sea cod
fit <- FLR_SAM(stk = cod4_stk, idx = cod4_idx, conf = cod4_conf_sam)

# coerce the output into FLStock
stk <- SAM2FLStock(fit)

# get catch estimates from model
stk <- SAM2FLStock(fit, catch_estimate = TRUE)

## Not run:
# use multi-fleet SAM model for western Baltic spring-spawning herring and
# load model fit from stockassessment.org
fit <- stockassessment::fitfromweb("WBSS_HAWG_2021")
stk <- SAM2FLStock(fit)

## End(Not run)
```

---

whg4_stk *North Sea whiting*

Description

Whiting (*Merlangius merlangus*) in Subarea 4 and Division 7.d (North Sea and eastern English Channel). Assessment input data for North Sea whiting as used by ICES WGNSSK 2018. Includes the stock, the indices and the SAM configuration. See the example below for how to run the assessment.

Usage

```
whg4_stk

whg4_idx

whg4_conf_sam
```

Format

An object of class `FLStock` of dimension 9 x 41 x 1 x 1 x 1 x 1.

An object of class `FLIndices` of length 2.

An object of class `list` of length 20.

Source

https://www.stockassessment.org

#' @examples # Replicate the 2017 assessment: fit <- FLR_SAM(stk = whg4_stk, idx = whg4_idx, conf = whg4_conf_sam)

Index

∗ **datasets**
    cod4_stk, 2
    had4_stk, 8
    her4_stk, 9
    mac.27.nea_stk_2019, 10
    ple7a_stk, 11
    whg4_stk, 16

cod4_conf_sam *(*cod4_stk*)*, 2
cod4_idx *(*cod4_stk*)*, 2
cod4_idx_2019 *(*cod4_stk*)*, 2
cod4_stk, 2
cod4_stk_2019 *(*cod4_stk*)*, 2

fit.spict, *6*
FLIndex, *4*, *6*
FLIndices, *4*, *6*
FLR_SAM, 3
FLR_SAM,FLStock,FLIndex-method
        *(*FLR_SAM*)*, 3
FLR_SAM,FLStock,FLIndices-method
        *(*FLR_SAM*)*, 3
FLR_SPiCT, 6
FLR_SPiCT,FLStock,FLIndex-method
        *(*FLR_SPiCT*)*, 6
FLR_SPiCT,FLStock,FLIndices-method
        *(*FLR_SPiCT*)*, 6
FLStock, *4*, *6*, *15*

getpars, 7
getpars,sam-method *(*getpars*)*, 7
getpars,sam_list-method *(*getpars*)*, 7

had4_conf_sam *(*had4_stk*)*, 8
had4_idx *(*had4_stk*)*, 8
had4_stk, 8
her4_conf_sam *(*her4_stk*)*, 9
her4_idx *(*her4_stk*)*, 9
her4_stk, 9

mac.27.nea_conf_2019
        *(*mac.27.nea_stk_2019*)*, 10
mac.27.nea_idx_2019
        *(*mac.27.nea_stk_2019*)*, 10
mac.27.nea_stk_2019, 10

ple7a_conf_sam *(*ple7a_stk*)*, 11
ple7a_idx *(*ple7a_stk*)*, 11
ple7a_idx_2019 *(*ple7a_stk*)*, 11
ple7a_stk, 11
ple7a_stk_2019 *(*ple7a_stk*)*, 11

SAM2FLStock, 13
SAM2FLStock,sam,FLStock-method
        *(*SAM2FLStock*)*, 13
SAM2FLStock,sam,missing-method
        *(*SAM2FLStock*)*, 13
SAM2FLStock,sam_list,FLStock-method
        *(*SAM2FLStock*)*, 13
SAM2FLStock,sam_list,missing-method
        *(*SAM2FLStock*)*, 13
SAM_uncertainty, 11
SAM_uncertainty,sam-method
        *(*SAM_uncertainty*)*, 11

whg4_conf_sam *(*whg4_stk*)*, 16
whg4_idx *(*whg4_stk*)*, 16
whg4_stk, 16