Package: EcoEnsemble (via r-universe)

October 29, 2026

Title A General Framework for Combining Ecosystem Models

Version 1.2.0

Description Fit and sample from the ensemble model described in Spence et al (2018): ``A general framework for combining ecosystem models"<doi:10.1111/faf.12310>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Biarch true

URL https://github.com/CefasRepRes/EcoEnsemble

BugReports https://github.com/CefasRepRes/EcoEnsemble/issues

Depends R (>= 3.5.0)

Imports methods, Rcpp, matrixcalc, RcppParallel (>= 5.0.1), rstan (>= 2.26.0), rstantools (>= 2.1.1), dplyr, ggplot2, reshape2, tibble, cowplot, posterior

LinkingTo BH (>= 1.66.0), Rcpp, RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

SystemRequirements GNU make

Suggests rmarkdown, kableExtra, knitr, testthat (>= 3.0.0), mgcv

Config/testthat/edition 3

VignetteBuilder knitr

Collate 'DiscrepancyPrior-IndLTPrior-class.R' 'DiscrepancyPrior-IndSTPrior-class.R' 'DiscrepancyPrior-ShaSTPrior-class.R' 'DiscrepancyPrior-TruthPrior-class.R' 'EcoEnsemble-package.R' 'EnsemblePrior-class.R' 'EnsembleData-class.R' 'EnsembleFit-class.R' 'EnsembleSample-class.R' 'Prior_sampling.R' 'RcppExports.R' 'data.R' 'fit_ensemble.R' 'fit_ensemble_withdrivers.R' 'generate_simulator_stan_data.R'

'generate_simulator_stan_data_withdrivers.R'
'get_samples_array.R' 'get_stan_outputs.R'
'get_stan_outputs_withdrivers.R' 'plot_functions.R'
'plot_functions_withdrivers.R' 'prior_functions.R'
'stan_sampling_utils.R' 'stanmodels.R' 'validation_functions.R'
'validation_functions_withdrivers.R'
'validation_prior_functions.R'

NeedsCompilation yes

Author Michael A. Spence [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3445-7979>), James A. Martindale
[aut] (ORCID: <https://orcid.org/0000-0002-1913-5592>), Michael
J. Thomson [aut] (ORCID:
<https://orcid.org/0000-0003-0284-0129>), Thomas I. J. Bartos
[aut], Luke E. K. Broadbent [aut]

Maintainer Michael A. Spence <michael.spence@cefas.gov.uk>

Config/pak/sysreqs make

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

Date/Publication 2026-04-27 18:45:13 UTC

RemoteUrl https://github.com/cran/EcoEnsemble

RemoteRef HEAD

RemoteSha 97375fdd6f472b9c82e127d95e226bfe5788d7b1

Contents

EcoEnsemble-package 3
EnsembleData 4
EnsembleData-class 5
EnsembleFit 6
EnsembleFit-class 6
EnsemblePrior-class 7
EnsembleSample 8
EnsembleSample-class 9
fit_ensemble_model 9
generate_sample 11
generate_sample_array 13
get_mcmc_ensemble_model 15
IndLTPrior 16
IndLTPrior-class 21
IndSTPrior-class 22
KalmanFilter_back 23
plot.EnsembleSample 24
prior_ensemble_model 25
sample_prior 26
ShaSTPrior-class 28
Sigma_ewe 29

Sigma_fs 29
Sigma_lm 30
Sigma_miz 30
Sigma_obs 31
SSB_ewe 31
SSB_fs 32
SSB_lm 32
SSB_miz 33
SSB_obs 34
TruthPrior-class 34

Index **36**

`EcoEnsemble-package` *A general framework for combining ecosystem models*

Description

The `EcoEnsemble` package implements the framework for combining ecosystem models laid out in Spence et al (2018).

Details

The ensemble model can be implemented in three main stages:

1. Eliciting priors on discrepancy terms: This is done by using the `EnsemblePrior` constructor.
2. Fitting the ensemble model: Using `fit_ensemble_model` with simulator outputs, observations and prior information. The ensemble model can be fit, obtaining either the point estimate, which maximises the posterior density, or running Markov chain Monte Carlo to generate a sample from the posterior denisty of the ensemble model.
3. Sampling the latent variables from the fitted model: Using `generate_sample` with the fitted ensemble object, the discrepancy terms and the ensemble's best guess of the truth can be generated. Similarly to `fit_ensemble_model`, this can either be a point estimate or a full sample.

Author(s)

Maintainer: Michael A. Spence <`michael.spence@cefas.gov.uk`> (ORCID)

Authors:

- James A. Martindale (ORCID)
- Michael J. Thomson (ORCID)
- Thomas I. J. Bartos
- Luke E. K. Broadbent

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

Spence, M. A., J. L. Blanchard, A. G. Rossberg, M. R. Heath, J. J. Heymans, S. Mackinson, N. Serpetti, D. C. Speirs, R. B. Thorpe, and P. G. Blackwell. 2018. "A General Framework for Combining Ecosystem Models." Fish and Fisheries 19: 1013-42. https://onlinelibrary.wiley.com/doi/abs/10.1111/faf.12310

See Also

Useful links:

- https://github.com/CefasRepRes/EcoEnsemble
- Report bugs at https://github.com/CefasRepRes/EcoEnsemble/issues

---

`EnsembleData` *Constructor for the* `EnsembleData` *class*

Description

A constructor for the `EnsembleData` class. This is used to convert input data into the required form for `fit_ensemble_model`.

Usage

```
EnsembleData(observations, simulators, priors, drivers = FALSE, MMod)
```

Arguments

`observations`	A `list` of length 2 containing observations and a covariance matrix. The first element is a `data.frame` or `matrix` with each column giving observations of each output of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is is a $d \times d$ `matrix` where d is the number of columns of the observations data frame / matrix. This matrix is the covariance matrix of the observations.
`simulators`	A `list` with length equal to the number of simulators. For each simulator, there is a `list` of 2 objects containing the simulator output and covariance matrix. The first element is a `data.frame` or `matrix` with each column giving a simulator outputs of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is a $n_k \times n_k$ `matrix` where n_k is the number of columns of the simulators output data frame / matrix. This matrix is the covariance matrix of the simulator outputs.
`priors`	An `EnsemblePrior` object specifying the prior distributions for the ensemble.
`drivers`	A `logical` indicating whether drivers have been used in combination with simulators. Default value is FALSE.
`MMod`	Not currently implemented.

Value

An object of class `EnsembleData`

See Also

`EnsembleData`, `EnsemblePrior`, `fit_ensemble_model`

Examples

```
ensemble_data <- EnsembleData(observations = list(SSB_obs, Sigma_obs),
                              simulators = list(list(SSB_ewe, Sigma_ewe, "EwE"),
                                                list(SSB_fs,  Sigma_fs, "FishSUMS"),
                                                list(SSB_lm,  Sigma_lm, "LeMans"),
                                                list(SSB_miz, Sigma_miz, "mizer")),
                               priors = EnsemblePrior(4))
```

EnsembleData-class *A class to hold the Ensemble data*

Description

A class that holds the observation data, simulator outputs, and prior information to convert into the required form for `fit_ensemble_model`.

Slots

`stan_input` A `list` of parameters in the correct form to fit the ensemble model in Stan.

`observations` A `list` of length 2 containing observations and a covariance matrix. The first element is a `data.frame` or `matrix` with each column giving observations of each output of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is the covariance matrix of the observations.

`simulators` A `list` with length equal to the number of simulators. For each simulator, there is a `list` of 2 objects containing the simulator output and covariance matrix. The first element is a `data.frame` or `matrix` with each column giving a simulator outputs of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is the covariance matrix of the simulator outputs.

`priors` An `EnsemblePrior` object specifying the prior distributions for the ensemble.

See Also

`EnsembleData`, `EnsemblePrior`, `fit_ensemble_model`

`EnsembleFit` *The constructor for the* `EnsembleFit` *object*

Description

The constructor for an `EnsembleFit` object. This function need not be called as an `EnsembleFit` object is constructed automatically by the `fit_ensemble_model` function. The `samples` slot contains the samples from the MCMC if a full sampling was completed, otherwise the `point_estimate` slot contains information about a point estimate.

Usage

```
EnsembleFit(ensemble_data, samples = NULL, point_estimate = NULL)
```

Arguments

`ensemble_data`	An `EnsembleData` object encapsulating the data used to fit the ensemble model.
`samples`	A `stanfit` object containing the samples drawn from the fitted model. The default value is `NULL`.
`point_estimate`	A `list` output of the optimised model. The default value is `NULL`.

Value

An object of class `EnsembleFit`

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

See Also

`EnsembleFit`, `fit_ensemble_model`,

`EnsembleFit-class` *A class to hold the samples or point estimates from the ensemble model.*

Description

An `EnsembleFit` object is returned by the `fit_ensemble_model` function. The object contains a slot for the `EnsembleData` object originally used to fit the ensemble model. The `samples` slot contains the samples from the MCMC if a full sampling was completed, otherwise the `point_estimate` slot contains information about a point estimate.

Slots

`ensemble_data` An `EnsembleData` object encapsulating the data used to fit the ensemble model.

`samples` A `stanfit` object containing the samples drawn from the fitted model. The default value is `NULL`.

`point_estimate` A `list` output of the optimised model. The default value is `NULL`.

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

See Also

`EnsembleFit`, `fit_ensemble_model`, `generate_sample`

`EnsemblePrior-class` *A class to hold the priors for the ensemble model.*

Description

An `EnsemblePrior` object encapsulates the prior information for the ensemble model.

Slots

`d` A `numeric` specifying the number of variables of interest in the ensemble.

`ind_st_params` A `list` containing a prior specification for the individual short-term discrepancies $z_k^{(t)}$. See details of the `EnsemblePrior()` constructor.

`ind_lt_params` A `list` containing a prior specification for the individual long-term discrepancies γ_k. See details of the `EnsemblePrior()` constructor.

`sha_st_params` A `list` containing a prior specification for the shared short-term discrepancies $\eta^{(t)}$. See details of the `EnsemblePrior()` constructor.

`sha_lt_params` A `numeric` containing the standard deviations for the normal prior used on the shared short-term discrepancy μ. If a single value is supplied, this is repeated for each variable

`truth_params` A `list` containing a prior specification for the processes on the truth $y^{(t)}$. See details of the `EnsemblePrior()` constructor. The default value is `TruthPrior(d)`.

`priors_stan_input` A `list` containing the prior data in the correct form to fit the model in Stan. This information is automatically generated by the constructor.

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

Lewandowski, Daniel, Dorota Kurowicka, and Harry Joe. 2009. "Generating Random Correlation Matrices Based on Vines and Extended Onion Method." Journal of Multivariate Analysis 100: 1989–2001.

`EnsembleSample` *A constructor for the* `EnsembleSample` *object*

Description

A constructor for the `EnsembleSample` class. These objects are generated automatically using the `generate_sample` function.

Usage

```
EnsembleSample(ensemble_fit, mle, samples)
```

Arguments

`ensemble_fit`	An `EnsembleFit` object containing the fitted ensemble model.
`mle`	An `array` of dimension $T \times (M+2) \times N_{sample}$ containing MLE point estimates from the `ensemble_fit` object, where T is the total time, M is the number of simulators and N_{sample} is the number of samples. For each time step, the `t`th element of the array is a `matrix` where each column is a sample and the rows are the variables: $$\left(y^{(t)}, \eta^{(t)}, z_1^{(t)}, z_2^{(t)}, \ldots, z_M^{(t)}\right)'$$ where $y^{(t)}$ is the ensemble model's prediction of the latent truth value at time t, $\eta^{(t)}$ is the shared short-term discrepancy at time t, $z_i^{(t)}$ is the individual short-term discrepancy of simulator i at time t.
`samples`	An `array` of dimension $T \times (M+2) \times N_{sample}$ containing samples from the `ensemble_fit` object, where T is the total time, M is the number of simulators and N_{sample} is the number of samples. For each time step, the `t`th element of the array is a `matrix` where each column is a sample and the rows are the variables: $$\left(y^{(t)}, \eta^{(t)}, z_1^{(t)}, z_2^{(t)}, \ldots, z_M^{(t)}\right)'$$ where $y^{(t)}$ is the ensemble model's prediction of the latent truth value at time t, $\eta^{(t)}$ is the shared short-term discrepancy at time t, $z_i^{(t)}$ is the individual short-term discrepancy of simulator i at time t.

Value

An object of class `EnsembleSample`

See Also

`EnsembleSample`, `generate_sample`

EnsembleSample-class *A class to hold samples of the ensemble model*

Description

EnsembleSample objects are generated using the generate_sample function.

Slots

ensemble_fit An EnsembleFit object containing the fitted ensemble model.

mle An array of dimension $T \times (M+2) \times N_{sample}$ containing MLE point estimates from the ensemble_fit object, where T is the total time, M is the number of simulators and N_{sample} is the number of samples. For each time step, the tth element of the array is a matrix where each column is a sample and the rows are the variables:

$$\left(y^{(t)}, \eta^{(t)}, z_1^{(t)}, z_2^{(t)}, \ldots, z_M^{(t)}\right)'$$

where $y^{(t)}$ is the ensemble model's prediction of the latent truth value at time t, $\eta^{(t)}$ is the shared short-term discrepancy at time t, $z_i^{(t)}$ is the individual short-term discrepancy of simulator i at time t.

samples An array of dimension $T \times (M+2) \times N_{sample}$ containing samples from the ensemble_fit object, where T is the total time, M is the number of simulators and N_{sample} is the number of samples. For each time step, the tth element of the array is a matrix where each column is a sample and the rows are the variables:

$$\left(y^{(t)}, \eta^{(t)}, z_1^{(t)}, z_2^{(t)}, \ldots, z_M^{(t)}\right)'$$

where $y^{(t)}$ is the ensemble model's prediction of the latent truth value at time t, $\eta^{(t)}$ is the shared short-term discrepancy at time t, $z_i^{(t)}$ is the individual short-term discrepancy of simulator i at time t.

See Also

EnsembleSample, generate_sample

fit_ensemble_model *Fits the ensemble model*

Description

fit_ensemble_model runs an MCMC of the ensemble model. This process can take a long time depending on the size of the datasets.

Usage

```
fit_ensemble_model(
  observations,
  simulators,
  priors,
  full_sample = TRUE,
  control = list(adapt_delta = 0.95),
  drivers = FALSE,
  sampler = c("explicit", "kalman"),
  MMod,
  ...
)
```

Arguments

observations	A `list` of length 2 containing observations and a covariance matrix. The first element is a `data.frame` or `matrix` with each column giving observations of each output of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is is a $d \times d$ `matrix` where d is the number of columns of the observations data frame / matrix. This matrix is the covariance matrix of the observations.
simulators	A `list` with length equal to the number of simulators. For each simulator, there is a `list` of 2 objects containing the simulator output and covariance matrix. The first element is a `data.frame` or `matrix` with each column giving a simulator outputs of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is a $n_k \times n_k$ `matrix` where n_k is the number of columns of the simulators output data frame / matrix. This matrix is the covariance matrix of the simulator outputs.
priors	An `EnsemblePrior` object specifying the prior distributions for the ensemble.
full_sample	A `logical` that runs a full sampling of the posterior density of the ensemble model if `TRUE`. If `FALSE`, returns the point estimate which maximises the posterior density of the ensemble model.
control	If creating a full sample, this is a named `list` of paramaters to control Stan's sampling behaviour. See the documentation of the `stan()` function in the `rstan` package for details. The default value is `list(adapt_delta = 0.95)`. If optimizing, this value is ignored.
drivers	A `logical` indicating whether drivers have been used in combination with simulators. Default value is FALSE.
sampler	A character string choosing between the "explicit" latent sampler (default) and the legacy "kalman" implementation.
MMod	Not currently implemented.
...	Additional arguments passed to the function `rstan::sampling` or `rstan::optimizing`.

Value

An `EnsembleFit` object.

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

See Also

EnsembleFit, EnsembleSample

Examples

```
fit <- fit_ensemble_model(observations = list(SSB_obs, Sigma_obs),
               simulators = list(list(SSB_ewe, Sigma_ewe, "EwE"),
                                 list(SSB_fs,  Sigma_fs, "FishSUMS"),
                                 list(SSB_lm,  Sigma_lm, "LeMans"),
                                 list(SSB_miz, Sigma_miz, "Mizer")),
               priors = EnsemblePrior(4,
               ind_st_params = IndSTPrior(parametrisation_form = "lkj",
               var_params= list(1,1), cor_params = 10, AR_params = c(2, 2))),
               full_sample = FALSE) #Only optimise in this case
```

`generate_sample` *Generate samples from a fitted ensemble model.*

Description

Methods to generates samples of the latent variables from a fitted ensemble model.

Usage

```
generate_sample(fit, num_samples = 1)

get_transformed_data(fit)

get_parameters(ex.fit, x = 1)

get_mle(x = 1, ex.fit, transformed_data, time)

gen_sample(x = 1, ex.fit, transformed_data, time)

get_transformed_data_dri(fit)
```

Arguments

`fit`	An `EnsembleFit` object.
`num_samples`	A `numeric` specifying the number of samples to be generated. The default is 1.

ex.fit	A list containing the samples / point estimate from the EnsembleFit object.
x	A numeric specifying which sample from the posterior to use. The default is 1.
transformed_data	A list of transformed input data.
time	A numeric specifying the time for which the ensemble model was run.

Details

The samples are created using the methods described in Strickland et. al. (2009) and Durbin and Koopman (2002).

Value

generate_sample gives a list of length 2, the first element being the MLE of latent variables and the second element being a set of samples of the latent variables.

- If fit is a sampling of the ensemble model parameters, then:
  - mle is a time$\times(3M+2)\times$ num_samples array, where M is the number of simulators and num_samples is the number of samples from the ensemble model, giving the MLE of the latent variables for each available sample from the ensemble model.
  - sample is a time$\times(3M+2)\times$ num_samples array, giving a sample of the latent variables for each available sample of the ensemble model.
- If fit is a point estimate of the ensemble model parameters, then:
  - mle is a time$\times(3M+2)\times$ 1 array giving the MLE of the latent variables for the point estimate of the ensemble model.
  - sample is a time$\times(3M+2)\times$ num_samples array, giving num_samples samples of the latent variables for the single point estimate of the ensemble model.

get_transformed_data gives a list of transformed input data.

get_parameters gives a list of ensemble parameters from the requested sample.

get_mle If fit is a sampling of the ensemble model parameters, then this is a time$\times(3M+2)\times$ num_samples array. If fit is a point estimate of the ensemble model parameters, then this is a time$\times(3M+2)\times$ 1 array giving the MLE of the latent variables for the point estimate of the ensemble model.

gen_sample If fit is a sampling of the ensemble model parameters, then this is a time$\times(3M+2)\times$ num_samples array, giving a sample of the latent variables for each available sample of the ensemble model. If fit is a point estimate of the ensemble model parameters, then this is a time$\times(3M+2)\times$ num_samples array.

References

Durbin, J. and Koopman, S. J. (2002). "A simple and efficient simulation smoother for state space time series analysis." Biometrika, 89(3), 603–616.

Chris M.Strickland, Ian. W.Turner, Robert Denhamb, Kerrie L.Mengersena. Efficient Bayesian estimation of multivariate state space models Computational Statistics & Data Analysis Volume 53, Issue 12, 1 October 2009, Pages 4116-4125

Examples

```
fit <- fit_ensemble_model(observations = list(SSB_obs, Sigma_obs),
               simulators = list(list(SSB_ewe, Sigma_ewe, "EwE"),
                                 list(SSB_fs,  Sigma_fs, "FishSUMS"),
                                 list(SSB_lm,  Sigma_lm, "LeMans"),
                                 list(SSB_miz, Sigma_miz, "Mizer")),
               priors = EnsemblePrior(4,
               ind_st_params = IndSTPrior(parametrisation_form = "lkj",
               var_params= list(1,1), cor_params = 10, AR_params = c(2, 2))),
               full_sample = FALSE) #Only optimise in this case
samples <- generate_sample(fit, num_samples = 2000)
```

---

generate_sample_array *Generate samples from a fitted ensemble model to get MCMC effective sample size diagnostics*

Description

Methods to generates samples of the latent variables from a fitted ensemble model in the same order as the MCMC and to calculate diagnostics effective sample size for the MCMC.

Usage

```
generate_sample_array(fit)

get_mle_array(ex.fit, transformed_data, time)

get_parameters_array(ex.fit)

get_param_idx(arr, stan_name)

gen_sample_array(ex.fit, transformed_data, time)

get_ESS_diag(fit, only_voi = TRUE)

calc_ess(sammy)
```

Arguments

`fit`	An `EnsembleFit` object.
`ex.fit`	A named `numeric`.
`transformed_data`	A `list` of transformed input data.
`time`	A `numeric` specifying the time for which the ensemble model was run.
`arr`	A `named numeric`

stan_name	A character object
only_voi	A logical indicating whether only the effective sample size of the variables of interest should be returned. Default value is TRUE.
sammy	An array of size the number of parameters × num_samples × nchains.

Details

Effective sample size is calculated from these samples, only for the quantity of interest if desired.

Value

generate_sample_array gives an array of size time * (MM + M + 2)*N × num_samples × nchains where M is the number of simulators, MM is the number of drivers (usually 1), N is the number of variables of interest num_samples is the number of samples from the ensemble model for each chain and nchains is the number of chains in the EnsembleFit object.

get_parameters_array gives a list of ensemble parameters from the requested sample.

get_mle_array gives an array of size time * (MM + M + 2)*N × num_samples × nchains.

gen_sample_array gives a list of two arrays of dimensions time * (MM + M + 2)*N × num_samples × nchains

get_ESS_diag and calc_ess gives a list of length two containing ESS_bulk and ESS_tail which has the bulk effective sample size and the tail effective sample size. Each list is of dimensions time × (MM + M + 2)*N.

See Also

See generate_sample() for information about the sampling and ess_bulk and ess_tail for information about the effective sample size calculations.

Examples

```
fit <- fit_ensemble_model(observations = list(SSB_obs, Sigma_obs),
               simulators = list(list(SSB_ewe, Sigma_ewe, "EwE"),
                                 list(SSB_fs,  Sigma_fs, "FishSUMS"),
                                 list(SSB_lm,  Sigma_lm, "LeMans"),
                                 list(SSB_miz, Sigma_miz, "Mizer")),
               priors = EnsemblePrior(4,
               ind_st_params = IndSTPrior(parametrisation_form = "lkj",
               var_params= list(1,1), cor_params = 10, AR_params = c(2, 2))))
get_ESS_diag(fit)
```

get_mcmc_ensemble_model *Return the compiled ensemble model Stan object.*

Description

Gets the unfit, compiled `stanmodel` object encoding the ensemble model. This allows for manual fitting of the ensemble model directly using `rstan::sampling`.

Usage

```
get_mcmc_ensemble_model(
  priors,
  likelihood = TRUE,
  drivers = FALSE,
  sampler = c("explicit", "kalman")
)
```

Arguments

priors	An `EnsemblePrior` object specifying the prior distributions for the ensemble for which the compiled `stanmodel` object will be obtained.
likelihood	A `logical` that returns the compiled `stanmodel` object including the likelihood (the Kalman filter) for given priors if `TRUE`. If `FALSE` returns the compiled `stanmodel` object without the likelihood for sampling from the prior.
drivers	A `logical` indicating whether drivers have been used in combination with simulators. Default value is FALSE.
sampler	A character string selecting which state-space sampler to use. Either `explicit` (the default), which samples latent states directly in the Stan program, or `kalman` which retains the Kalman filter implementation.

Value

The `stanmodel` object encoding the ensemble model.

Examples

```
priors <- EnsemblePrior(4)
mod <- get_mcmc_ensemble_model(priors)

ensemble_data <- EnsembleData(observations = list(SSB_obs, Sigma_obs),
                              simulators = list(list(SSB_ewe, Sigma_ewe, "EwE"),
                                                list(SSB_fs,  Sigma_fs, "FishSUMS"),
                                                list(SSB_lm,  Sigma_lm, "LeMans"),
                                                list(SSB_miz, Sigma_miz, "mizer")),
                               priors = priors)
```

```
out <- rstan::sampling(mod, ensemble_data@stan_input, chains = 1)
```

IndLTPrior *Constructor for the* EnsemblePrior *class*

Description

Constructors for the EnsemblePrior class and related classes. These functions are used to encode prior information for the ensemble model. The IndSTPrior, IndLTPrior, ShaSTPrior, and TruthPrior constructors encapsulate prior information.

Usage

```
IndLTPrior(
  parametrisation_form = "lkj",
  var_params = list(1, 1),
  cor_params = 1
)

IndSTPrior(
  parametrisation_form = "hierarchical",
  var_params = list(-3, 1, 8, 4),
  cor_params = list(0.1, 0.1, 0.1, 0.1),
  AR_params = c(2, 2)
)

ShaSTPrior(
  parametrisation_form = "lkj",
  var_params = list(1, 10),
  cor_params = 1,
  AR_params = c(2, 2)
)

TruthPrior(
  d,
  initial_mean = 0,
  initial_var = 100,
  rw_covariance = list(2 * d, diag(d))
)

EnsemblePrior(
  d,
  ind_st_params = IndSTPrior(),
  ind_lt_params = IndLTPrior(),
  sha_st_params = ShaSTPrior(),
  sha_lt_params = 5,
```

```
  truth_params = TruthPrior(d)
)
```

Arguments

parametrisation_form	The parametrisation by which the covariance matrix of the noise of the AR process (in the case of IndSTPrior and ShaSTPrior objects or the covariance of the distribution of long-term discrepancies for IndLTPrior objects) is decomposed. The default is hierarchical for IndSTPrior objects, and lkj otherwise. See details.
var_params	The parameters characterising the variance of the AR process (in the case of IndSTPrior and ShaSTPrior objects or the variance of the distribution of long-term discrepancies for IndLTPrior objects) on the discrepancy. The default value is list(-3, 1, 8, 4) for IndSTPrior objects, list(1, 1) for IndLTPrior objects, and list(1, 10) for ShaSTPrior objects. See details.
cor_params	The parameters characterising the correlations of the AR process (or the distribution of long-term discrepancies) on the short-term discrepancies. The default value in this case is to use list(0.1,0.1,0.1,0.1) for IndSTPrior objects, and 1 for IndLTPrior and ShaSTPrior objects. See details.
AR_params	The parameters giving the beta parameters for the prior distribution on the autoregressive parameter of the AR(1) process. The default is c(2,2). See details.
d	A numeric specifying the number of variables of interest in the ensemble.
initial_mean	A numeric giving the mean of the normal distribution giving the prior on the initial value of the random walk. This is the same value for each variable Default value is 0.
initial_var	A numeric giving the variance of the normal distribution giving the prior on the initial value of the random walk. This is the same value for each variable Default value is 100.
rw_covariance	A list of length 2 containing the inverse-Wishart parameters for the covariance of the random walk of the truth. The default value is list(2*d, diag(d)).
ind_st_params	An IndSTPrior object specifying priors for the individual short-term discrepancies $z_k^{(t)}$. The default value is IndSTPrior("hierarchical", list(-3, 1, 8, 4), list(0.1, 0.1, 0.1, 0.1), c(2, 2)).
ind_lt_params	An IndLTPrior object specifying priors for the individual long-term discrepancies γ_k. The default value is IndLTPrior("lkj",list(1, 1), 1).
sha_st_params	A ShaSTPrior object specifying priors for the shared short-term discrepancies $\eta^{(t)}$. The default value is ShaSTPrior("lkj",list(1, 10), 1, c(2,2)).
sha_lt_params	A numeric of length d or 1 containing the standard deviations for the normal prior used on the shared short-term discrepancy μ. If a single value is supplied, this is repeated for each variable of interest. The default value is 5.
truth_params	A TruthPrior object specifying priors for the processes on the truth $y^{(t)}$. The default value is TruthPrior(d).

Details

`IndSTPrior` and `ShaSTPrior` discrepancy prior parameter objects contain 4 slots corresponding to:

1. `parametrisation_form` - A `character` specifying how the priors are parametrised. Currently supported priors are `'lkj'`, `'inv_wishart'`, `'beta'`, `'hierarchical'`, or `'hierarchical_beta_conjugate'` (`'hierarchical'` and `'hierarchical_beta_conjugate'` are only supported for `IndSTPrior` objects).
2. `var_params` - The prior parameters for the discrepancy variances, either a `list` of length `2` or a `numeric` of length `4`. See below.
3. `cor_params` - The correlation matrix parameters, either a `list` of length `2`, a `numeric` of length `3` or a `numeric` of length `4`. See below.
4. `AR_params` - Parameters for the autoregressive parameter as a `numeric` of length `2`.

`IndLTPrior` discrepancy prior parameter objects contain the slots `parametrisation_form`, `var_params`, and `cor_params`.

There are currently five supported prior distributions on covariance matrices. As in Spence et. al. (2018), the individual and shared short-term discrepancy covariances, Λ_k and Λ_η, as well as the individual long-term discrepancy covariance, Λ_γ, are decomposed into a vector of variances and a correlation matrix

$$\Lambda = \sqrt{\text{diag}(\pi)} P \sqrt{\text{diag}(\pi)},$$

where π is the vector of variances for each variable of interest (VoI), and P is the correlation matrix.

Selecting `'lkj'`, `'inv_wishart'`, `'beta'`, `'hierarchical'` or `'hierarchical_beta_conjugate'` refers to setting LKJ, inverse Wishart, beta or hierarchical (with gamma-distributed hyperparameters or beta-conjugate-distributed hyperparameters) prior distributions on the covariance matrix respectively. The variance parameters should be passed through as the `var_params` slot of the object and the correlation parameters should be passed through as the `cor_params`. For `'lkj'`, `'inv_wishart'`, and `'beta'` selections, variances are parameterised by gamma distributions, so the `var_params` slot should be a `list` of length two, where each element gives the shape and rate parameters for each VoI (either as a single value which is the same for each VoI or a `numeric` with the same length as the number of VoI). For example, setting `var_params = list(c(5,6,7,8), c(4,3,2,1))` would correspond to a `Gamma(5, 4)` prior on the variance of the first VoI, a `Gamma(6, 3)` prior on the variance of the second VoI, etc... The correlations should be in the following form:

- If `'lkj'` is selected, then `cor_params` should be a `numeric` η giving the LKJ shape parameter, such that the probability density is given by (Lewandowski et. al. 2009)

  $$f(\Sigma|\eta) \propto \det(\Sigma)^{\eta-1}.$$

  Variances are parameterised by gamma distributions.

- If `'inv_wishart'` is selected, then `cor_params` should be a `list` containing a scalar value η (giving the degrees of freedom) and a symmetric, positive definite matrix S (giving the scale matrix). The dimensions of S should be the same as the correlation matrix it produces (i.e $d \times d$ where d is the number of VoI). The density of an inverse Wishart is given by

  $$f(W|\eta, S) = \frac{1}{2^{\eta d/2}\Gamma_N\left(\frac{\eta}{2}\right)} |S|^{\eta/2} |W|^{-(\eta+d+1)/2} \exp\left(-\frac{1}{2}\text{tr}\left(SW^{-1}\right)\right),$$

where Γ_N is the multivariate gamma function and $\mathrm{tr}\,(\mathrm{X})$ is the trace of X. Note that inverse Wishart distributions act over the space of all covariance matrices. When used for a correlation matrix, only the subset of valid covariance matrices that are also valid correlation matrices are considered. Variances are parameterised by gamma distributions.

- If `'beta'` is selected, then `cor_params` should be a `list` containing two symmetric `d`×`d` matrices A and B giving the prior success parameters and prior failure parameters respectively. The correlation between the `i`th and `j`th VoI is $\rho_{i,j}$ with

$$\frac{1}{\pi}\tan^{-1}\frac{\rho_{i,j}}{\sqrt{1-\rho_{i,j}^2}}+\frac{1}{2}\sim\mathrm{beta}(A_{i,j},B_{i,j}).$$

  Variances are parameterised by gamma distributions.

- If `'hierarchical'` or `'hierarchical_beta_conjugate'` is selected, then variances are parameterised by log-normal distributions:

$$\log\pi_{k,i}\sim\mathrm{N}(\mu_i,\sigma_i^2)$$

  with priors

$$\mu_i\sim\mathrm{N}(\alpha_\pi,\beta_\pi),$$
$$\sigma_i^2\sim\mathrm{InvGamma}(\gamma_\pi,\delta_\pi).$$

  The `var_params` slot should then be a `numeric` of length 4, giving the $\alpha_\pi,\beta_\pi,\gamma_\pi,\delta_\pi$ hyperparameters respectively. Correlations ($\rho_{k,i,j}$ where $\rho_{k,i,j}$ is the correlation between VoI i and j for the kth simulator) are parameterised by hierarchical beta distributions.

$$\frac{\rho_{k,i,j}+1}{2}\sim\mathrm{beta}(c_{k,i,j},d_{k,i,j})$$

  with priors

$$c_{k,i,j}\sim\mathrm{gamma}(\alpha_\rho,\beta_\rho),$$
$$d_{k,i,j}\sim\mathrm{gamma}(\gamma_\rho,\delta_\rho).$$

  NOTE: These options is only supported for the individual short-term discrepancy terms.

- If `'hierarchical'` is selected, then the `cor_params` slot should be a `numeric` of length 4 giving the $\alpha_\rho,\beta_\rho,\gamma_\rho,\delta_\rho$ hyperparameters. respectively. NOTE: This option is only supported for the individual short-term discrepancy terms.

- If `'hierarchical_beta_conjugate'` is selected, then the `cor_params` slot should be a `numeric` of length 3. Denoting the values by r,s,k, they map to the hyperparameters p,q,k of the beta conjugate distribution via $k=k$, $p^{-1/k}=(1+e^{-s})(1+e^{-r})$ and $q^{1/k}=e^{-r}(1+e^{-s})^{-1}(1+e^{-r})^{-1}$. The density of the beta conjugate distribution is defined up to a constant of proportionality by

$$p(\alpha_\rho,\beta_\rho\,|\,p,q,k)\propto\frac{\Gamma(\alpha_\rho+\beta_\rho)^k p^{\alpha_\rho}q^{\beta_\rho}}{\Gamma(\alpha_\rho)^k\Gamma(\beta_\rho)^k}\,.$$

  NOTE: This option is only supported for the individual short-term discrepancy terms. Priors may also be specified for the autoregressive parameters for discrepancies modelled using autoregressive processes (i.e. for `IndSTPrior` and `ShaSTPrior` objects). These are parametrised via beta distributions such that the autoregressive parameter $R\in(-1,1)$ satisfies

$$\frac{R+1}{2}\sim\mathrm{Beta}(\alpha,\beta)$$

.

In addition to priors on the discrepancy terms, it is also possible to add prior information on the truth. We require priors on the truth at $t = 0$. By default, a $N(0, 10)$ prior is used on the initial values., however this can be configured by the `truth_params` argument. The covariance matrix of the random walk of the truth Λ_y can be configured using an inverse-Wishart prior. The `truth_params` argument should be a `TruthPrior` object.

Value

`EnsemblePrior` returns an object of class `EnsemblePrior`. `IndSTPrior` returns an object of class `IndSTPrior`. `IndLTPrior` returns an object of class `IndLTPrior`. `ShaSTPrior` returns an object of class `ShaSTPrior`. `TruthPrior` returns an object of class `TruthPrior`.

References

Spence et. al. (2018). A general framework for combining ecosystem models. *Fish and Fisheries*, 19(6):1031-1042.

Examples

```
##### Different forms of the individual long term discrepancy priors
#LKJ(10) priors on correlation matrices and gamma(5, 3) priors on the variances
ist_lkj <- IndSTPrior("lkj", list(5, 3), 10)#

#Same as above but with an additional beta(2, 4) prior on
#the autoregressive parameter of the AR process.
ist_lkj <- IndSTPrior("lkj", list(5, 3), 10, AR_params = c(2, 4))

#Same as above but with different variance priors for 5 different variables of interest.
#This encodes that there is a gamma(1, 1) prior on the variance of the first variable,
#a gamma(23, 1) on the second variable etc...
ist_lkj <- IndSTPrior("lkj", list(c(1,23,24,6,87), c(1,1,1,1,5)), 10, AR_params = c(2, 4))

#Hierarchical priors with gamma(1,2) and gamma(10, 1) on the variance hyperparameters and
#gamma(3,4), gamma(5,6) on the correlation hyperparameters
ist_hie <- IndSTPrior("hierarchical", list(1,2,10,1), list(3,4,5,6))

#Hierarchical priors with gamma(1,2) and gamma(10, 1) on the variance hyperparameters and
#the beta conjugate prior with parameters (p = 0.75, q = 0.75, k = 0.2) on the
#correlation hyperparameters
ist_hie_beta_conj <- IndSTPrior("hierarchical_beta_conjugate",
list(1,2,10,1), list(0.75,0.75,0.2))

#Inverse Wishart correlation priors. Gamma(2, 1/3) priors are on the variances and
#inv-Wishart(5, diag(5)) on the correlation matrices.
ist_inW <- IndSTPrior("inv_wishart", list(2, 1/3),list(5, diag(5)))

##### TruthPrior
#Simple default truth prior with 7 variables of interest
truth_def <- TruthPrior(7)
# A more fine-tuned truth prior for an ensemble with 7 species.
truth_cus <- TruthPrior(7, initial_mean = 2, initial_var = 10, rw_covariance = list(10, diag(7)))
```

```
#The default priors for an ensemble with 8 variables of interest
priors <- EnsemblePrior(8)

#With 4 variables of interest.
priors <- EnsemblePrior(4)

#Defining custom priors for a model with 4 species.
num_species <- 5
priors <- EnsemblePrior(
  d = num_species,
  ind_st_params = IndSTPrior("lkj",  list(3, 2), 3, AR_params = c(2,4)),
  ind_lt_params = IndLTPrior(
    "beta",
    list(c(10,4,8, 7,6),c(2,3,1, 4,4)),
    list(matrix(5, num_species, num_species),
         matrix(0.5, num_species, num_species))
  ),
  sha_st_params = ShaSTPrior("inv_wishart",list(2, 1/3),list(5, diag(num_species))),
  sha_lt_params = 5,
  truth_params = TruthPrior(d = num_species, initial_mean = 5, initial_var = 10,
                            rw_covariance = list(10, diag(10)))
)
```

`IndLTPrior-class` *A class to hold the priors for the ensemble model.*

Description

An `IndLTPrior` object encapsulates the prior information for the long-term discrepancies of the individual simulators within the ensemble model.

Details

Individual long-term discrepancies γ_k are drawn from a distribtion

$$\gamma_k \sim N(0, C_\gamma).$$

Accepted parametrisation forms for this discrepancy are `lkj`, `beta`, or `inv_wishart`. See details of the `EnsemblePrior()` constructor for more details.

Slots

`parametrisation_form` The parametrisation by which the covariance matrix of the noise of the AR process is decomposed.

`var_params` The parameters characterising the variance of the distribution of the individual long-term discrepancy.

`cor_params` The parameters characterising the correlations of the distribution of the individual long-term discrepancy.

See Also

IndSTPrior, ShaSTPrior, TruthPrior,

IndSTPrior-class *A class to hold the priors for the ensemble model.*

Description

An IndSTPrior object encapsulates the prior information for the short-term discrepancies of the individual simulators within the ensemble model.

Details

Individual short-term discrepancies $z_k^{(t)}$ are modelled as an AR(1) process so that

$$z_k^{(t+1)} \sim N(R_k z_k^{(t)}, \Lambda_k).$$

Accepted parametrisation forms for this discrepancy are lkj, beta, inv_wishart, hierarchical or hierarchical_beta_conjugate. See details of the EnsemblePrior() constructor for more details.

Slots

AR_param The parameters giving the beta parameters for the autoregressive parameter of the AR(1) process.

parametrisation_form The parametrisation by which the covariance matrix of the noise of the AR process is decomposed.

var_params The parameters characterising the variance of the AR process on the individual short-term discrepancy.

cor_params The parameters characterising the correlations of the AR process on the individual short-term discrepancy. .

See Also

IndLTPrior, ShaSTPrior, TruthPrior,

KalmanFilter_back *Backwards Kalman filter*

Description

Finds the most likely path through a dynamical linear model.

Usage

```
KalmanFilter_back(rhos, dee, R, Q, C, P, xhat, Time, y, obs)
```

Arguments

rhos	A numeric containing the diagonal elements of the transition matrix of the evolution equation.
dee	A numeric of the same length as rho containing the discrepancies or biases in the observation process.
R	A numeric representing the variances of the observation process.
Q	A matrix of dimensions length(rho) and length(rho) representing the covariance of the evolution process.
C	A a matrix of dimensions length(rho) and length(R) representing the observation operator of the observation process.
P	A matrix of dimensions length(rho) and length(rho) representing the covariance matrix of the first state of the evolution process.
xhat	A numeric of the same length as rho representing the expectation of the first state of the evolution process.
Time	A numeric The length of time of the dynamical linear model.
y	A matrix of dimensions Time and length(R) of observations of the observation process.
obs	A matrix of dimensions Time and length(R). 1 means in the i,jth element means that the jth output is observed at tim i.

Details

For the model with the evolution process

$$x_{t+1} \sim N(\rho \cdot x_t, Q)$$

and observation process

$$y_t \sim N(\rho(x_t + \delta), diag(R))$$

.

Using the sequential Kalman filter, the function gives the mostly path of x_t for all t.

Value

A `matrix` with dimensions `nrow(time)` and `length(xhat)` representing the most likely values of the latent variables.

References

Chui, C.K. & Chen, G. (2009) Kalman Filtering with Real-Time Applications. Springer, Berlin, Heidelberg, Fourth Edtion.

Kalman, R. E. (1960) A new approach to linear filtering and prediction problems. Trans. ASME, J. Basic Eng., 82, pp. 35-45.

Examples

```
fit <- fit_ensemble_model(observations = list(SSB_obs, Sigma_obs),
               simulators = list(list(SSB_ewe, Sigma_ewe, "EwE"),
                                 list(SSB_fs,  Sigma_fs, "FishSUMS"),
                                 list(SSB_lm,  Sigma_lm, "LeMans"),
                                 list(SSB_miz, Sigma_miz, "Mizer")),
               priors = EnsemblePrior(4,
               ind_st_params = IndSTPrior(parametrisation_form = "lkj",
               var_params= list(1,1), cor_params = 10, AR_params = c(2, 2))),
               full_sample = FALSE) #Only optimise in this case
transformed_data <- get_transformed_data(fit)
ex.fit <- fit@point_estimate$par
params <- get_parameters(ex.fit)
ret <- KalmanFilter_back(params$AR_params, params$lt_discrepancies,
                         transformed_data$all_eigenvalues_cov,params$SIGMA,
                         transformed_data$bigM, params$SIGMA_init, params$x_hat,
                         fit@ensemble_data@stan_input$time,transformed_data$new_data,
                         transformed_data$observation_available)
```

`plot.EnsembleSample` *Plot the ensemble output*

Description

Plots the latent variables predicted by the ensemble model, along with simulator outputs and observations.

Usage

```
## S3 method for class 'EnsembleSample'
plot(x, variable = NULL, quantiles = c(0.05, 0.95), ...)
```

Arguments

x	An EnsembleSample object.
variable	The name of the variable to plot. This can either be a character string in the same form as the observation variable, or an index for the column in the observations data frame.
quantiles	A numeric vector of length 2 giving the quantiles for which to plot ribbons if doing a full sampling of the ensemble model. The default is c(0.05,0.95).
...	Other arguments passed on to methods. Not currently used.

Value

The ggplot object.

Examples

```
priors <- EnsemblePrior(4)
prior_density <- prior_ensemble_model(priors, M = 4)
samples <- sample_prior(observations = list(SSB_obs, Sigma_obs),
             simulators = list(list(SSB_miz, Sigma_miz),
                               list(SSB_ewe, Sigma_ewe),
                               list(SSB_fs, Sigma_fs),
                               list(SSB_lm, Sigma_lm)),
             priors = priors,
             sam_priors = prior_density)
plot(samples) #Plot the prior predictive density.
plot(samples, variable="Herring")
```

---

prior_ensemble_model *Generate samples of parameters from prior distribution*

Description

Methods to generates samples of the parameters from the prior distribution of the ensemble model.

Usage

```
prior_ensemble_model(
  priors,
  M = 1,
  MM = NULL,
  full_sample = TRUE,
  control = list(adapt_delta = 0.95),
  ...
)
```

Arguments

priors	An EnsemblePrior object specifying the prior distributions for the ensemble.
M	A numeric that represents the number of simulators. The default is 1.
MM	A numeric that represents the number of drivers. The default is NULL.
full_sample	A logical that runs a full sampling of the prior density of the ensemble model if TRUE. If FALSE, returns the point estimate which maximises the prior density of the ensemble model.
control	If creating a full sample, this is a named list of parameters to control Stan's sampling behaviour. See the documentation of the stan() function in the rstan package for details. The default value is list(adapt_delta = 0.95). If optimizing, this value is ignored.
...	Additional arguments passed to the function rstan::sampling or rstan::optimizing.

Value

A list containing two items named samples and point_estimate. If full_sample==TRUE, samples is a stanfit and point_estimate is a NULL object, else samples is a NULL and point_estimate is a list object. It is possible to generate a point estimate for the prior if the individual short-term discrepancy prior follows a hierarchical parameterisation.

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

See Also

EnsembleFit

Examples

```
priors <- EnsemblePrior(4)
prior_density <- prior_ensemble_model(priors, M = 4)
```

---

sample_prior *Generate samples of latent variables from prior predictive distribution*

Description

Methods to generates samples of the latent variables from the prior predictive distribution of the ensemble model.

Usage

```
sample_prior(
  observations,
  simulators,
  priors,
  sam_priors,
  num_samples = 1,
  full_sample = TRUE,
  drivers = FALSE,
  ...
)
```

Arguments

observations	A list of length 2 containing observations and a covariance matrix. The first element is a data.frame or matrix with each column giving observations of each output of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is is a $d \times d$ matrix where d is the number of columns of the observations data frame / matrix. This matrix is the covariance matrix of the observations.
simulators	A list with length equal to the number of simulators. For each simulator, there is a list of 2 objects containing the simulator output and covariance matrix. The first element is a data.frame or matrix with each column giving a simulator outputs of interest and each row a time. Rows should be named with the times and columns should be named the variables. The second element is a $n_k \times n_k$ matrix where n_k is the number of columns of the simulators output data frame / matrix. This matrix is the covariance matrix of the simulator outputs.
priors	An EnsemblePrior object specifying the prior distributions for the ensemble.
sam_priors	A list containing two items named samples and point_estimate. samples is either a NULL or a stanfit object containing the samples drawn from the prior distribution of the ensemble model and point_estimate is either a NULL or a list object containing the optimised prior distribution of the ensemble model. If this object is missing then sample_prior generates it.
num_samples	A numeric specifying the number of samples to be generated. The default is 1.
full_sample	A logical that runs a full sampling of the prior density of the ensemble model if TRUE. If FALSE, returns the point estimate which maximises the prior density of the ensemble model.
drivers	A logical indicating whether drivers have been used in combination with simulators. Default value is FALSE.
...	Additional arguments passed to the function rstan::sampling or rstan::optimizing.

Details

The samples are created using the methods described in Strickland et. al. (2009) and Durbin and Koopman (2002).

Value

An `EnsembleSample` object.

References

Durbin, J. and Koopman, S. J. (2002). "A simple and efficient simulation smoother for state space time series analysis." Biometrika, 89(3), 603–616.

Chris M.Strickland, Ian. W.Turner, RobertDenhamb, Kerrie L.Mengersena. Efficient Bayesian estimation of multivariate state space models Computational Statistics & Data Analysis Volume 53, Issue 12, 1 October 2009, Pages 4116-4125

See Also

`EnsembleFit`, `EnsembleSample`, `generate_sample`, `prior_ensemble_model`

Examples

```
priors <- EnsemblePrior(4)
prior_density <- prior_ensemble_model(priors, M = 4)
samples <- sample_prior(observations = list(SSB_obs, Sigma_obs),
             simulators = list(list(SSB_miz, Sigma_miz),
                               list(SSB_ewe, Sigma_ewe),
                               list(SSB_fs, Sigma_fs),
                               list(SSB_lm, Sigma_lm)),
             priors = priors,
             sam_priors = prior_density)
plot(samples) #Plot the prior predictive density.
```

ShaSTPrior-class *A class to hold the priors for the ensemble model.*

Description

An `ShaSTPrior` object encapsulates the prior information for the short-term discrepancies of the shared discrepancy of the ensemble model.

Details

Shared short-term discrepancies $\eta^{(t)}$ are modelled as an AR(1) process so that

$$\eta^{(t+1)} \sim N(R_\eta \eta, \Lambda_\eta).$$

Accepted parametrisation forms for this discrepancy are `lkj`, `beta`, or `inv_wishart`. See details of the `EnsemblePrior()` constructor for more details.

Slots

`AR_param` The parameters giving the beta parameters for the main parameter of the AR(1) process.

`parametrisation_form` The parametrisation by which the covariance matrix of the noise of the AR process is decomposed.

`var_params` The parameters characterising the variance of the AR process on the shared short-term discrepancy.

`cor_params` The parameters characterising the correlations of the AR process on the shared short-term discrepancy.

`Sigma_ewe` *Ecopath with EcoSim Sigma*

Description

A 4x4 covariance matrix quantifying the parameter uncertainty of Ecopath with EcoSim

Usage

```
Sigma_ewe
```

Format

A 4x4 `matrix`.

References

Mackinson, S., Platts, M., Garcia, C., and Lynam, C. (2018). Evaluating the fishery and ecological consequences of the proposed North Sea multi-annual plan. PLOS ONE, 13(1), 1–23.

`Sigma_fs` *FishSUMS Sigma*

Description

A 3x3 covariance matrix quantifying the parameter uncertainty of FishSUMS

Usage

```
Sigma_fs
```

Format

A 3x3 `matrix`.

References

Spence, M. A., Blanchard, J. L., Rossberg, A. G., Heath, M. R., Heymans, J. J., Mackinson, S., Serpetti, N., Speirs, D. C., Thorpe, R. B., and Blackwell, P. G. (2018). A general framework for combining ecosystem models. Fish and Fisheries, 19(6), 1031–1042.

`Sigma_lm` *LeMans Sigma*

Description

LeMans Sigma

Usage

```
Sigma_lm
```

Format

A 4x4 `matrix`. A 4x4 covariance matrix quantifying the parameter uncertainty of LeMans

References

Thorpe, R. B., Le Quesne, W. J. F., Luxford, F., Collie, J. S., and Jennings, S. (2015). Evaluation and management implications of uncertainty in a multispecies size-structured model of population and community responses to fishing. Methods in Ecology and Evolution, 6(1), 49–58.

`Sigma_miz` *mizer Sigma*

Description

mizer Sigma

Usage

```
Sigma_miz
```

Format

A 4x4 `matrix`. A 4x4 covariance matrix quantifying the parameter uncertainty of mizer

References

Spence, M. A., Blackwell, P. G., and Blanchard, J. L. (2016). Parameter uncertainty of a dynamic multispecies size spectrum model. Canadian Journal of Fisheries and Aquatic Sciences, 73(4), 589–59

Sigma_obs *Stock assessment Sigma*

Description

A 4x4 covariance matrix quantifying the covariances of the stock assessment estimates of biomass.

Usage

```
Sigma_obs
```

Format

A 4x4 `matrix`.

References

Herring Assessment Working Group for the Area South of 62 N (HAWG).Technical report, ICES Scientific Reports. ACOM:07. 960 pp, ICES, Copenhagen.

Report of the Working Group on the Assessment of Demersal Stocks inthe North Sea and Skagerrak. Technical report, ICES Scientific Reports. ACOM:22.pp, ICES, Copenhagen.

SSB_ewe *Ecopath with EcoSim SSB*

Description

A dataset containing the predictions for spawning stock biomass of Norway Pout, Herring, Cod, and Sole in the North Sea between 1991-2050 under an MSY fishing scenario from EcoPath with EcoSim.

Usage

```
SSB_ewe
```

Format

A `data.frame` with 60 rows and 4 variables:

`N.pout` Spawning stock biomass of Norway Pout, in log tonnes.

`Herring` Spawning stock biomass of Herring, in log tonnes.

`Cod` Spawning stock biomass of Cod, in log tonnes.

`Sole` Spawning stock biomass of Sole, in log tonnes.

References

ICES (2016). Working Group on Multispecies Assessment Methods (WGSAM). Technical report, International Council for Exploration of the Seas.

`SSB_fs` *FishSUMS SSB*

Description

A `data frame` containing the predictions for spawning stock biomass of Norway Pout, Herring, and Cod in the North Sea between 1984-2050 under an MSY fishing scenario from FishSUMS. Note that FishSUMS does not produce outputs for Sole.

Usage

```
SSB_fs
```

Format

A `data.frame` with 67 rows and 3 variables:

`N.pout` Spawning stock biomass of Norway Pout, in log tonnes.

`Herring` Spawning stock biomass of Herring, in log tonnes.

`Cod` Spawning stock biomass of Cod, in log tonnes.

References

Speirs, D., Greenstreet, S., and Heath, M. (2016). Modelling the effects of fishing on the North Sea fish community size composition. Ecological Modelling, 321, 35–45

`SSB_lm` *LeMans SSB*

Description

A `data.frame` containing the predictions for spawning stock biomass of Norway Pout, Herring, Cod, and Sole in the North Sea between 1986-2050 under an MSY fishing scenario from the LeMaRns package.

Usage

```
SSB_lm
```

Format

A `data.frame` with 65 rows and 4 variables:

`N.pout` Spawning stock biomass of Norway Pout, in log tonnes.

`Herring` Spawning stock biomass of Herring, in log tonnes.

`Cod` Spawning stock biomass of Cod, in log tonnes.

`Sole` Spawning stock biomass of Sole, in log tonnes.

References

Thorpe, R. B., Le Quesne, W. J. F., Luxford, F., Collie, J. S., and Jennings, S. (2015). Evaluation and management implications of uncertainty in a multispecies size-structured model of population and community responses to fishing. Methods in Ecology and Evolution, 6(1), 49–58.

`SSB_miz` *mizer SSB*

Description

A `data.frame` containing the predictions for spawning stock biomass of Norway Pout, Herring, Cod, and Sole in the North Sea between 1984-2050 under an MSY fishing scenario from mizer.

Usage

```
SSB_miz
```

Format

A data frame with 67 rows and 4 variables:

`N.pout` Spawning stock biomass of Norway Pout, in log tonnes.

`Herring` Spawning stock biomass of Herring, in log tonnes.

`Cod` Spawning stock biomass of Cod, in log tonnes.

`Sole` Spawning stock biomass of Sole, in log tonnes.

References

Blanchard, J. L., Andersen, K. H., Scott, F., Hintzen, N. T., Piet, G., and Jennings, S. (2014). Evaluating targets and trade-offs among fisheries and conservation objectives using a multispecies size spectrum model. Journal of Applied Ecology, 51(3), 612–622

SSB_obs *Stock assessment SSB*

Description

A `data.frame` containing the single species stock assessment estimates of spawning stock biomass of Norway Pout, Herring, Cod, and Sole in the North Sea between 1984-2017.

Usage

```
SSB_obs
```

Format

A data frame with 34 rows and 4 variables:

`N.pout` Spawning stock biomass of Norway Pout, in log tonnes.

`Herring` Spawning stock biomass of Herring, in log tonnes.

`Cod` Spawning stock biomass of Cod, in log tonnes.

`Sole` Spawning stock biomass of Sole, in log tonnes.

References

Herring Assessment Working Group for the Area South of 62 N (HAWG)(2020).Technical report, ICES Scientific Reports. ACOM:07. 960 pp, ICES, Copenhagen.

Report of the Working Group on the Assessment of Demersal Stocks in the North Sea and Skagerrak (2020). Technical report, ICES Scientific Reports. ACOM:22.pp, ICES, Copenhagen.

TruthPrior-class *A class to hold the priors for the truth model in the ensemble framework*

Description

An `TruthPrior` object encapsulates the prior information for the short-term discrepancies of the shared discrepancy of the ensemble model.

Details

The truth $\mathbf{y}^{(t)}$ is modelled as a random walk such that

$$\mathbf{y}^{(t+1)} \sim N(\mathbf{y}^{(t)}, \Lambda_y).$$

The covariance matrix Λ_y is parameterised by an inverse Wishart distribution (contained in the `rw_covariance` slot) and the initial value is modelled as drawn from a normal distribution.

Slots

`d` A `numeric` giving the number of variables of interest in the ensemble model.

`initial_mean` A `numeric` giving the standard deviation of the normal prior on the initial mean value of the random walk. This is the same standard deviation for each variable of interest.

`initial_var` A `list` of length `2` containing the shape and scale parameters (respectively) for the gamma priors on the variance of the initial value of the truth.

`rw_covariance` A `list` of length `2` containing the inverse-Wishart parameters for the covariance of the random walk of the truth.

Index

∗ **datasets**
Sigma_ewe, 29
Sigma_fs, 29
Sigma_lm, 30
Sigma_miz, 30
Sigma_obs, 31
SSB_ewe, 31
SSB_fs, 32
SSB_lm, 32
SSB_miz, 33
SSB_obs, 34

calc_ess *(*generate_sample_array*)*, 13

EcoEnsemble *(*EcoEnsemble-package*)*, 3
EcoEnsemble-package, 3
EnsembleData, 4, *5*
EnsembleData-class, 5
EnsembleFit, 6, *6, 7*, *11*, *26*, *28*
EnsembleFit-class, 6
EnsemblePrior, *5*
EnsemblePrior *(*IndLTPrior*)*, 16
EnsemblePrior-class, 7
EnsembleSample, 8, *8, 9*, *11*, *28*
EnsembleSample-class, 9
ess_bulk, *14*
ess_tail, *14*

fit_ensemble_model, *5–7*, 9

gen_sample *(*generate_sample*)*, 11
gen_sample_array
*(*generate_sample_array*)*, 13
generate_sample, *7–9*, 11, *28*
generate_sample(), *14*
generate_sample_array, 13
get_ESS_diag *(*generate_sample_array*)*, 13
get_mcmc_ensemble_model, 15
get_mle *(*generate_sample*)*, 11
get_mle_array *(*generate_sample_array*)*,
13
get_param_idx *(*generate_sample_array*)*,
13
get_parameters *(*generate_sample*)*, 11
get_parameters_array
*(*generate_sample_array*)*, 13
get_transformed_data *(*generate_sample*)*,
11
get_transformed_data_dri
*(*generate_sample*)*, 11

IndLTPrior, 16, *22*
IndLTPrior-class, 21
IndSTPrior, *22*
IndSTPrior *(*IndLTPrior*)*, 16
IndSTPrior-class, 22

KalmanFilter_back, 23

plot.EnsembleSample, 24
prior_ensemble_model, 25, *28*

sample_prior, 26
ShaSTPrior, *22*
ShaSTPrior *(*IndLTPrior*)*, 16
ShaSTPrior-class, 28
Sigma_ewe, 29
Sigma_fs, 29
Sigma_lm, 30
Sigma_miz, 30
Sigma_obs, 31
SSB_ewe, 31
SSB_fs, 32
SSB_lm, 32
SSB_miz, 33
SSB_obs, 34

TruthPrior, *22*
TruthPrior *(*IndLTPrior*)*, 16
TruthPrior-class, 34