Package 'flocker'

July 22, 2025

Title Flexible Occupancy Estimation with Stan

Version 1.0-0

Description Fit occupancy models in 'Stan' via 'brms'. The full variety of 'brms' formula-based effects structures are available to use in multiple classes of occupancy model, including single-season models, models with data augmentation for never-observed species, dynamic (multiseason) models with explicit colonization and extinction processes, and dynamic models with autologistic occupancy dynamics. Formulas can be specified for all relevant distributional terms, including detection and one or more of occupancy, colonization, extinction, and autologistic depending on the model type. Several important forms of model post-processing are provided. References: Bürkner (2017) <doi:10.18637/jss.v080.i01>; Carpenter et al. (2017) <doi:10.18637/jss.v076.i01>; Socolar & Mills (2023) <doi:10.1101/2023.10.26.564080>.

License BSD_3_clause + file LICENSE

URL https://github.com/jsocolar/flocker, https://jsocolar.github.io/flocker/

BugReports https://github.com/jsocolar/flocker/issues

Depends R (>= 4.1.0)

Imports abind, assertthat, boot, brms (>= 2.20.3), loo (>= 2.0.0), MASS, matrixStats, stats, utils, withr

Suggests BH (>= 1.75.0-0), knitr, RcppEigen (>= 0.3.3.9.3), rmarkdown, rstan (>= 2.26.0), spelling, testthat (>= 2.1.0), tibble

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Language en-US

NeedsCompilation no

Author Jacob B. Socolar [aut, cre, cph],
Simon C. Mills [aut],
Paul-Christian Bürkner [ctb]

Maintainer Jacob B. Socolar <jacob.socolar@gmail.com>

Repository CRAN

Date/Publication 2024-02-05 20:20:05 UTC

Contents

`example_flocker_model_single`
Example single-season flocker model

Description

A fitted single-season occupancy model from flocker

Usage

```
example_flocker_model_single
```

Format

'example_flocker_model_single' A flocker_fit and brmsfit object

Source

data-raw/example_flocker_model.R

`fitted_flocker` *Posterior predictive distributions for modeled probabilities*

Description

Get expected values of the posterior predictive distribution for the modeled probabilities (occupancy, detection, colonization, extinction, autologistic).

Usage

```
fitted_flocker(
  flocker_fit,
  components = c("occ", "det", "col", "ex", "auto", "Omega"),
  new_data = NULL,
  unit_level = FALSE,
  summarise = FALSE,
  CI = c(0.05, 0.95),
  draw_ids = NULL,
  response = TRUE,
  re_formula = NULL,
  allow_new_levels = FALSE,
  sample_new_levels = "uncertainty"
)
```

Arguments

`flocker_fit`	A flocker_fit object.
`components`	a character vector specifying one or more of "occ", "det", "col", "ex", "auto", and "Omega" for which to obtain fitted values.
`new_data`	Optional new data at which to evaluate occupancy predictions. New data can be passed as a flocker_data object produced by `make_flocker_data` or as a dataframe with one row per desired prediction. If 'NULL' (the default) expected values are generated for the original data as formatted by make_flocker_data.
`unit_level`	Logical; defaults to FALSE. Relevant only when 'new_data' is not a dataframe (i.e. it is 'NULL' or a flocker_data object), and useful only for multiseason models with missing seasons. If FALSE, returns in the shape of the observation matrix/array with NAs for missing visits. If TRUE, returns in the shape of the first visit, and returns values for all units that are not part of a trailing block of never-visited units, including never-visited units that are part of series with subsequent visits.
`summarise`	if TRUE, return the expected value and upper and lower bound of the credible interval, otherwise return posterior draws.

CI	A vector of length 2 specifying the upper and lower bounds of the credible interval.
draw_ids	Vector of indices of the posterior draws to be used. If 'NULL' (the default) all draws are used in their native order.
response	Should results be returned on the response or logit scale? Defaults to 'TRUE', i.e. response scale. However, the autologistic parameter is not interpretable as a probability and is always returned on the logit scale regardless of the value of 'response'
re_formula	formula containing group-level effects to be considered in the prediction. If 'NULL' (default), include all group-level effects; if NA, include no group-level effects.
allow_new_levels	allow new levels for random effect terms in 'new_data'? Will error if set to 'FALSE' and new levels are provided in 'new_data'.
sample_new_levels	If new_data is provided and contains random effect levels not present in the original data, how should predictions be handled? Passed directly to 'brms::prepare_predictions', which see.

Details

The probabilities returned are conditional probabilities (e.g. detection conditional on occupancy, colonization conditional on previous non-occupancy, etc). These probabilities are not conditioned on the observed histories (e.g. the occupancy probability is not fixed to one at sites with a detection; it is estimated only based on the covariates).

Value

A list of sets of expected values (one per component). If 'new_data' is a dataframe, each element contains one row per row of 'new_data'. Otherwise, returns in the shape of the observation matrix/array used to format the flocker_data (but see 'unit_level' parameter for further details).

Examples

```
fitted_flocker(
  example_flocker_model_single,
  summarise = TRUE
)
```

---

flock *Fit an occupancy model*

Description

Fit an occupancy model

Usage

```
flock(
  f_occ = NULL,
  f_det,
  flocker_data,
  data2 = NULL,
  multiseason = NULL,
  f_col = NULL,
  f_ex = NULL,
  multi_init = NULL,
  f_auto = NULL,
  augmented = FALSE,
  threads = NULL,
  ...
)
```

Arguments

f_occ	A brms-type model formula for occupancy. If provided, must begin with "~".
f_det	A brms-type model formula for detection. Must begin with "~". OR, a brmsformula object including formulas for all of the relevant distributional parameters in the desired model (det and at least one of occ, colo, ex, autologistic, and Omega). The $formula element of the brmsformula must be the detection formula, beginning with det ~. This latter option unadvisable except when necessary (e.g. when a nonlinear formula is desired), as input checking is less thorough.
flocker_data	data, generally the output of make_flocker_data().
data2	additional data (e.g. a covariance matrix for a phylogenetic effect)
multiseason	Must be NULL (the default) or one of "colex" or "autologistic". If NULL, data must be formatted for a single-season model. Otherwise, the data must be formatted for a multiseason model. If "colex", a colonization-extinction model will be fit, and ‘f_col‘ and ‘f_ex‘ must be specified. If "autologistic", an autologistic model will be fit, and ‘f_col‘ and ‘f_ex‘ must both be NULL.
f_col	A brms-type model formula for colonization in colonization-extinction dynamic models. If provided, must begin with "~".
f_ex	A brms-type model formula for extinction probabilities in colonization-extinction dynamic models. If provided, must begin with "~".
multi_init	Must be NULL unless the model is a dynamic (multiseason) model, in which case must be either "explicit" or "equilibrium". If "explicit", then ‘f_occ‘ must be provided to model occupancy probabilities in the first timestep. If "equilibrium", then ‘f_occ‘ must be ‘NULL‘ and the initial occupancy probabilities are assumed to be the (possibly site-specific) equilibrium probabilities from the colonization- extinction dynamics.
f_auto	Relevant only for autologistic models. A brms-type model formula for the autologistic offset parameter (theta). If provided, must begin with "~".
augmented	Logical. Must be TRUE if data are formatted for a data-augmented multi-species model, and FALSE otherwise.

threads	NULL or positive integer. If integer, the number of threads to use per chain in within chain parallelization. Currently available only with single-season rep-constant models, and must be set to NULL otherwise.
...	additional arguments passed to `brms::brm()`

Value

a `brmsfit` containing the fitted occupancy model.

Examples

```
sfd <- simulate_flocker_data()
fd <- make_flocker_data(
 sfd$obs,
 sfd$unit_covs,
 sfd$event_covs
)
flock(
  f_occ = ~ s(uc1) + (1|species),
  f_det = ~ uc1 + ec1 + (1|species),
  flocker_data = fd,
  refresh = 50, chains = 1, warmup = 5, iter = 200,
  control = list(adapt_engaged = FALSE, stepsize = .05, max_treedepth = 5),
  seed = 123
  )
```

---

flocker_stancode *Generate stan code for an occupancy model*

Description

Generate stan code for an occupancy model

Usage

```
flocker_stancode(
  f_occ = NULL,
  f_det,
  flocker_data,
  data2 = NULL,
  multiseason = NULL,
  f_col = NULL,
  f_ex = NULL,
  multi_init = NULL,
  f_auto = NULL,
  augmented = FALSE,
  threads = NULL,
  ...
)
```

Arguments

f_occ	A brms-type model formula for occupancy. If provided, must begin with "~".
f_det	A brms-type model formula for detection. Must begin with "~". OR, a brmsformula object including formulas for all of the relevant distributional parameters in the desired model (det and at least one of occ, colo, ex, autologistic, and Omega). The $formula element of the brmsformula must be the detection formula, beginning with det ~. This latter option unadvisable except when necessary (e.g. when a nonlinear formula is desired), as input checking is less thorough.
flocker_data	data, generally the output of make_flocker_data().
data2	additional data (e.g. a covariance matrix for a phylogenetic effect)
multiseason	Must be NULL (the default) or one of "colex" or "autologistic". If NULL, data must be formatted for a single-season model. Otherwise, the data must be formatted for a multiseason model. If "colex", a colonization-extinction model will be fit, and 'f_col' and 'f_ex' must be specified. If "autologistic", an autologistic model will be fit, and 'f_col' and 'f_ex' must both be NULL.
f_col	A brms-type model formula for colonization in colonization-extinction dynamic models. If provided, must begin with "~".
f_ex	A brms-type model formula for extinction probabilities in colonization-extinction dynamic models. If provided, must begin with "~".
multi_init	Must be NULL unless the model is a dynamic (multiseason) model, in which case must be either "explicit" or "equilibrium". If "explicit", then 'f_occ' must be provided to model occupancy probabilities in the first timestep. If "equilibrium", then 'f_occ' must be 'NULL' and the initial occupancy probabilities are assumed to be the (possibly site-specific) equilibrium probabilities from the colonization- extinction dynamics.
f_auto	Relevant only for autologistic models. A brms-type model formula for the autologistic offset parameter (theta). If provided, must begin with "~".
augmented	Logical. Must be TRUE if data are formatted for a data-augmented multi-species model, and FALSE otherwise.
threads	NULL or positive integer. If integer, the number of threads to use per chain in within chain parallelization. Currently available only with single-season rep-constant models, and must be set to NULL otherwise.
...	additional arguments passed to brms::brm()

Value

generated stancode

Examples

```
sfd <- simulate_flocker_data()
fd <- make_flocker_data(
 sfd$obs,
 sfd$unit_covs,
 sfd$event_covs
```

```
)
flocker_stancode(
  f_occ = ~ s(uc1) + (1|species),
  f_det = ~ uc1 + ec1 + (1|species),
  flocker_data = fd,
  refresh = 50, chains = 1, warmup = 5, iter = 200,
  control = list(adapt_engaged = FALSE, stepsize = .05, max_treedepth = 5),
  seed = 123
  )
```

`flocker_standata` *Generate stan data for an occupancy model*

Description

Generate stan data for an occupancy model

Usage

```
flocker_standata(
  f_occ = NULL,
  f_det,
  flocker_data,
  data2 = NULL,
  multiseason = NULL,
  f_col = NULL,
  f_ex = NULL,
  multi_init = NULL,
  f_auto = NULL,
  augmented = FALSE,
  threads = NULL,
  ...
)
```

Arguments

`f_occ`	A brms-type model formula for occupancy. If provided, must begin with "~".
`f_det`	A brms-type model formula for detection. Must begin with "~". OR, a `brmsformula` object including formulas for all of the relevant distributional parameters in the desired model (det and at least one of occ, colo, ex, autologistic, and Omega). The `$formula` element of the `brmsformula` must be the detection formula, beginning with `det ~`. This latter option unadvisable except when necessary (e.g. when a nonlinear formula is desired), as input checking is less thorough.
`flocker_data`	data, generally the output of `make_flocker_data()`.
`data2`	additional data (e.g. a covariance matrix for a phylogenetic effect)

multiseason	Must be NULL (the default) or one of "colex" or "autologistic". If NULL, data must be formatted for a single-season model. Otherwise, the data must be formatted for a multiseason model. If "colex", a colonization-extinction model will be fit, and ‘f_col‘ and ‘f_ex‘ must be specified. If "autologistic", an autologistic model will be fit, and ‘f_col‘ and ‘f_ex‘ must both be NULL.
f_col	A brms-type model formula for colonization in colonization-extinction dynamic models. If provided, must begin with "~".
f_ex	A brms-type model formula for extinction probabilities in colonization-extinction dynamic models. If provided, must begin with "~".
multi_init	Must be NULL unless the model is a dynamic (multiseason) model, in which case must be either "explicit" or "equilibrium". If "explicit", then ‘f_occ‘ must be provided to model occupancy probabilities in the first timestep. If "equilibrium", then ‘f_occ‘ must be ‘NULL‘ and the initial occupancy probabilities are assumed to be the (possibly site-specific) equilibrium probabilities from the colonization- extinction dynamics.
f_auto	Relevant only for autologistic models. A brms-type model formula for the autologistic offset parameter (theta). If provided, must begin with "~".
augmented	Logical. Must be TRUE if data are formatted for a data-augmented multi-species model, and FALSE otherwise.
threads	NULL or positive integer. If integer, the number of threads to use per chain in within chain parallelization. Currently available only with single-season rep-constant models, and must be set to NULL otherwise.
...	additional arguments passed to `brms::brm()`

Value

generated stan data

Examples

```
sfd <- simulate_flocker_data()
fd <- make_flocker_data(
 sfd$obs,
 sfd$unit_covs,
 sfd$event_covs
)
flocker_standata(
  f_occ = ~ s(uc1) + (1|species),
  f_det = ~ uc1 + ec1 + (1|species),
  flocker_data = fd,
  refresh = 50, chains = 1, warmup = 5, iter = 200,
  control = list(adapt_engaged = FALSE, stepsize = .05, max_treedepth = 5),
  seed = 123
  )
```

`get_flocker_prior` *Get prior for occupancy model*

Description

Get prior for occupancy model

Usage

```
get_flocker_prior(
  f_occ = NULL,
  f_det,
  flocker_data,
  data2 = NULL,
  multiseason = NULL,
  f_col = NULL,
  f_ex = NULL,
  multi_init = NULL,
  f_auto = NULL,
  augmented = FALSE,
  threads = NULL,
  ...
)
```

Arguments

`f_occ`	A brms-type model formula for occupancy. If provided, must begin with "~".
`f_det`	A brms-type model formula for detection. Must begin with "~". OR, a `brmsformula` object including formulas for all of the relevant distributional parameters in the desired model (det and at least one of occ, colo, ex, autologistic, and Omega). The `$formula` element of the `brmsformula` must be the detection formula, beginning with `det ~`. This latter option unadvisable except when necessary (e.g. when a nonlinear formula is desired), as input checking is less thorough.
`flocker_data`	data, generally the output of `make_flocker_data()`.
`data2`	additional data (e.g. a covariance matrix for a phylogenetic effect)
`multiseason`	Must be NULL (the default) or one of "colex" or "autologistic". If NULL, data must be formatted for a single-season model. Otherwise, the data must be formatted for a multiseason model. If "colex", a colonization-extinction model will be fit, and ‘f_col‘ and ‘f_ex‘ must be specified. If "autologistic", an autologistic model will be fit, and ‘f_col‘ and ‘f_ex‘ must both be NULL.
`f_col`	A brms-type model formula for colonization in colonization-extinction dynamic models. If provided, must begin with "~".
`f_ex`	A brms-type model formula for extinction probabilities in colonization-extinction dynamic models. If provided, must begin with "~".

multi_init	Must be NULL unless the model is a dynamic (multiseason) model, in which case must be either "explicit" or "equilibrium". If "explicit", then ‘f_occ‘ must be provided to model occupancy probabilities in the first timestep. If "equilibrium", then ‘f_occ‘ must be ‘NULL‘ and the initial occupancy probabilities are assumed to be the (possibly site-specific) equilibrium probabilities from the colonization- extinction dynamics.
f_auto	Relevant only for autologistic models. A brms-type model formula for the autologistic offset parameter (theta). If provided, must begin with "~".
augmented	Logical. Must be TRUE if data are formatted for a data-augmented multi-species model, and FALSE otherwise.
threads	NULL or positive integer. If integer, the number of threads to use per chain in within chain parallelization. Currently available only with single-season rep-constant models, and must be set to NULL otherwise.
...	additional arguments passed to `brms::brm()`

Value

A dataframe summarizing the parameters on which priors can be specified and giving the default priors for those parameters. See `?brms::get_prior` for further details.

Examples

```
sfd <- simulate_flocker_data()
fd <- make_flocker_data(
 sfd$obs,
 sfd$unit_covs,
 sfd$event_covs
)
get_flocker_prior(
 f_occ = ~ s(uc1) + + (1|species),
 f_det = ~ uc1 + ec1 + (1|species),
 flocker_data = fd
)
```

---

get_Z *Get posterior distribution of the Z matrix*

Description

Get posterior distribution of the Z matrix

Usage

```
get_Z(
  flocker_fit,
  draw_ids = NULL,
  history_condition = TRUE,
```

```
  sample = FALSE,
  new_data = NULL,
  allow_new_levels = FALSE,
  sample_new_levels = "uncertainty"
)
```

Arguments

flocker_fit	A flocker_fit object
draw_ids	Vector of indices of the posterior draws to be used. If 'NULL' (the default) all draws are used in their native order.
history_condition	Should the posterior distribution for Z directly condition on the observed detection history ('TRUE') or not ('FALSE')? For example, at sites with at least one detection, the true occupancy state conditioned on the history is one with absolute certainty. Without directly conditioning on the history, the occupancy state is controlled by the posterior distribution for the occupancy probability psi.
sample	Should the return be posterior probabilities of occupancy (FALSE), or bernoulli samples from those probabilities (TRUE)
new_data	Optional new data at which to predict the Z matrix. Can be the output of 'make_flocker_data' or the 'unit_covs' input to 'make_flocker_data' provided that 'history_condition' is 'FALSE' and the occupancy model is a single-season, non-augmented model.
allow_new_levels	allow new levels for random effect terms in 'new_data'? Will error if set to 'FALSE' and new levels are provided in 'new_data'.
sample_new_levels	If 'new_data' is provided and contains random effect levels not present in the original data, how should predictions be handled? Passed directly to 'brms::prepare_predictions', which see.

Value

The posterior Z matrix in the shape of the first visit in 'obs' as passed to make_flocker_data, with posterior iterations stacked along the final dimension

Examples

```
get_Z(example_flocker_model_single)
```

log1m_inv_logit *Numerically stable log one-minus inverse logit*

Description

Numerically stable log one-minus inverse logit

Usage

```
log1m_inv_logit(x)
```

Arguments

x	real number or vector of reals

Value

the logarithm of one minus the inverse logit of x

Examples

```
log1m_inv_logit(0)
```

log_inv_logit *Numerically stable log inverse logit*

Description

Numerically stable log inverse logit

Usage

```
log_inv_logit(x)
```

Arguments

x	real number or vector of reals

Value

the logarithm of the inverse logit of x

Examples

```
log_inv_logit(0)
```

log_lik_flocker *Compute unit-wise or series-wise log-likelihood matrix for a flocker_fit object*

Description

Compute unit-wise or series-wise log-likelihood matrix for a flocker_fit object

Usage

```
log_lik_flocker(flocker_fit, draw_ids = NULL)
```

Arguments

flocker_fit	A flocker_fit object
draw_ids	the draw ids to compute log-likelihoods for. Defaults to using the full posterior.

Details

In single-season models, rows are units (e.g. points or species-points; suitable for leave-one-unit-out CV). In multiseason models, rows are series (i.e. points or species-points, suitable for leave-one-series-out CV). In augmented models, rows are species (suitable for leave-one-species-out CV).

Value

A posterior log-likelihood matrix, where iterations are rows and units, series, or species are columns.

Examples

```
log_lik_flocker(example_flocker_model_single)
```

loo_compare_flocker *LOO comparisons for flocker models.*

Description

LOO comparisons for flocker models.

Usage

```
loo_compare_flocker(model_list, model_names = NULL, thin = NULL)
```

Arguments

model_list	a list of flocker_fit objects.
model_names	An optional vector of names for the models.
thin	specify the amount of thinning required. 1 or NULL results in no thinning, 2 retains every other value, 3 every third, etc.

Value

a ‘compare.loo‘ matrix

Examples

```
ml <- rep(list(example_flocker_model_single), 3)
loo_compare_flocker(ml)
```

---

loo_flocker *Compute loo for flocker_fit objects*

Description

Compute loo for flocker_fit objects

Usage

```
loo_flocker(x, thin = NULL)
```

Arguments

x	a flocker_fit object or a list of flocker_fit objects
thin	specify the amount of thinning required. 1 or NULL implies no thinning, 2 implies every other value, 3 every third, etc.

Value

a loo object or a list of loo objects

Examples

```
## Not run:
loo_flocker(example_flocker_model_single)

## End(Not run)
```

`make_flocker_data` *Format data for occupancy model with* `flock()`.

Description

Format data for occupancy model with `flock()`.

Usage

```
make_flocker_data(
  obs,
  unit_covs = NULL,
  event_covs = NULL,
  type = "single",
  n_aug = NULL,
  quiet = FALSE
)
```

Arguments

`obs`	If `type = "single"`, an I x J matrix-like object where closure is assumed across rows and columns are repeated sampling events. If `type = "multi"`, an I x J x K array where rows are sites or species-sites, columns are repeated sampling events, and slices along the third dimension are seasons. Allowable values are 1 (detection), 0 (no detection), and NA (no sampling event). If `type = "augmented"`, an L x J x K array where rows L are sites, columns J are repeat sampling events, and slices K are species. The data must be packed so that, for a given unit (site, site-species, site-timestep, site-species-timestep) all realized visits come before any missing visits (NAs are trailing within their rows).
`unit_covs`	If `type = "single"` a dataframe of covariates for each closure-unit that are constant across repeated sampling events within units. If `type = "multi"`, a list of such dataframes, one per timestep. All dataframes must have identical column names and types, and all dataframes must have I rows. If `type = "augmented"`, a dataframe of covariates for each site that are constant across repeated sampling events within sites (no dependence on species is allowed).
`event_covs`	If `type = "single"`, a named list of I x J matrices, each one corresponding to a covariate that varies across repeated sampling events within closure-units. If `type = "multi"`, a named list of I x J x K arrays, each one corresponding to a covariate that varies across repeated sampling events within closure-units. If `type = "augmented"`, a named list of L x J matrices, each one corresponding to a covariate that varies across repeated sampling events within sites (no dependence on species is allowed).
`type`	The type of occupancy model desired. Options are: `"single"` for a single_season model, `"multi"` for a multi-season (dynamic) model, or `"augmented"` for a single-season multi-species model with data-augmentation for never-observed pseudospecies.

n_aug	Number of pseudo-species to augment. Only applicable if type = "augmented".
quiet	Hide progress bars and informational messages?

Value

A flocker_data list that can be passed as data to flock().

Examples

```
sfd <- simulate_flocker_data()
make_flocker_data(
 sfd$obs,
 sfd$unit_covs,
 sfd$event_covs
)
```

---

make_flocker_data_augmented

#' Format data for data-augmented occupancy model, to be passed to flock().

Description

#' Format data for data-augmented occupancy model, to be passed to flock().

Usage

```
make_flocker_data_augmented(
  obs,
  n_aug,
  site_covs = NULL,
  event_covs = NULL,
  quiet = FALSE
)
```

Arguments

obs	An I x J x K array where rows I are sites, columns J are repeat sampling events, and slices K are species. Allowable values are 1 (detection), 0 (no detection), and NA (no sampling event). The data must be formatted so that all NAs are trailing within their rows.
n_aug	Number of pseudospecies to augment
site_covs	A dataframe of covariates for each site that are constant across repeated sampling events.
event_covs	A named list of I x J matrices, each one corresponding to a covariate that varies across repeated sampling events within sites
quiet	Hide progress bars and informational messages?

Value

A flocker_data list that can be passed as data to `flocker()`.

`make_flocker_data_dynamic` *Format data for dynamic (multi-season) occupancy model, to be passed to* `flock()`.

Description

Format data for dynamic (multi-season) occupancy model, to be passed to `flock()`.

Usage

```
make_flocker_data_dynamic(
  obs,
  unit_covs = NULL,
  event_covs = NULL,
  quiet = FALSE
)
```

Arguments

`obs`	An I x J x K array where closure is assumed across rows, columns are repeated sampling events, and slices along the third dimension are seasons. Allowable values are 1 (detection), 0 (no detection), and NA (no sampling event). The data must be formatted so that all NAs are trailing within their rows across repeat visits, but not necessarily across seasons.
`unit_covs`	A list of dataframes (one per season) of covariates for each closure-unit that are constant across repeated sampling events within units. All dataframes must have identical column names and types, and all dataframes must have I rows.
`event_covs`	A named list of I x J x K arrays, each one corresponding to a covariate that varies across repeated sampling events within closure-units
`quiet`	Hide progress bars and informational messages?

Value

A flocker_data list that can be passed as data to `flock()`.

make_flocker_data_static *Format data for single-season occupancy model, to be passed to* `flock()`.

Description

Format data for single-season occupancy model, to be passed to `flock()`.

Usage

```
make_flocker_data_static(
  obs,
  unit_covs = NULL,
  event_covs = NULL,
  quiet = FALSE
)
```

Arguments

`obs`	An I x J matrix-like object where closure is assumed across rows and columns are repeated sampling events. Allowable values are 1 (detection), 0 (no detection), and NA (no sampling event).
`unit_covs`	A dataframe of covariates for each unit that are constant across repeated sampling events within closure-units.
`event_covs`	A named list of I x J matrices, each one corresponding to a covariate that varies across repeated sampling events within closure-units
`quiet`	Hide progress bars and informational messages?

Value

A flocker_data list that can be passed as data to `flock()`.

Examples

```
sfd <- simulate_flocker_data()
make_flocker_data_static(
 sfd$obs,
 sfd$unit_covs,
 sfd$event_covs
)
```

predict_flocker *Get posterior predictions from a flocker model*

Description

Get posterior predictions from a flocker model

Usage

```
predict_flocker(
  flocker_fit,
  draw_ids = NULL,
  history_condition = FALSE,
  new_data = NULL,
  mixed = FALSE,
  allow_new_levels = FALSE,
  sample_new_levels = "uncertainty"
)
```

Arguments

flocker_fit	A 'flocker_fit' object
draw_ids	Vector of indices of the posterior draws to be used. If 'NULL' (the default) all draws are used in their native order.
history_condition	Logical indicator of whether to directly condition the posterior for the occupancy state on the observed detection histories. For example, at sites with at least one detection, the true occupancy state conditioned on the history is one with absolute certainty. Without directly conditioning on the history, the occupancy state is controlled exclusively by the posterior distribution for the occupancy probability psi.
new_data	Optional new data at which to predict. If 'NULL', predictions are given at the data points used for model fitting. Otherwise, must be a flocker_data object produced by 'make_flocker_data'.
mixed	When 'new_data' is not provided, should random effect levels be drawn from their posteriors ('FALSE', the default) or re-sampled from their fitted hyperparameters ('TRUE'). The latter can be useful for mixed predictive checking. Note that setting to TRUE is not available for grouping terms involved in phylogenetic random effects or other random effects with specified covariance structures.
allow_new_levels	Should new_data be allowed to contain new levels for random effects?
sample_new_levels	If new_data is provided and contains random effect levels not present in the original data, how should predictions be handled? Passed directly to brms::prepare_predictions, which see.

Value

An array of posterior predictions in the same shape as the observations passed to ‘make_flocker_data()‘ with posterior iterations stacked along the final dimension.

Examples

```
unconditioned_preds <- predict_flocker(example_flocker_model_single)
conditioned_preds <- predict_flocker(
 example_flocker_model_single,
 history_condition = TRUE
)
```

---

simulate_flocker_data *Simulate data for use with* make_flocker_data() *and downstream functions.*

Description

Data will be simulated with one unit covariate that affects all relevant terms, one event covariate that affects detection (unless ‘rep_constant‘ is ‘TRUE‘), and one grouping factor representing species with correlated effects on all terms.

Usage

```
simulate_flocker_data(
  n_rep = 4,
  n_pt = 50,
  n_sp = 30,
  n_season = 1,
  multiseason = NULL,
  multi_init = NULL,
  augmented = FALSE,
  rep_constant = FALSE,
  params = NULL,
  covariates = NULL,
  seed = 123,
  ragged_rep = FALSE,
  missing_seasons = FALSE
)
```

Arguments

n_rep	number of replicate visits to simulate per closure unit
n_pt	number of points to simulate. The number of units for single- season models will be ‘n_pt*n_sp‘. The number of units for multi-season models will be ‘n_pt*n_sp*n_season‘.

n_sp	number of levels to include in random effect. For compatibility with multi-species models where the random effect represents species, the data get expanded such that there's a row (closure-unit) for each combination of sampling point and effect level (i.e. species)
n_season	Number of seasons desired. 1 yields data for a single-season model; all other positive integers yield data for multiseason models.
multiseason	if n_season is NULL, must be NULL. Otherwise, one of "colex" or "autologistic".
multi_init	if n_season is NULL, must be NULL. Otherwise, one of "explicit" or "equilibrium".
augmented	logical. If 'TRUE' data will be formatted for an augmented model, which requires that 'n_season == 1'. All never-observed species will be trimmed out of the data, and the default parameters will be modified to increase random effect variances for the detection and occupancy, intercepts and to decrease random effect variances for detection slopes, thus encouraging the existence of never-observed species. Furthermore, the data will be simulated without any covariate influence on occupancy.
rep_constant	logical: create data with unit covariates only (TRUE) or data that includes event covariates (FALSE)
params	a named list containing of parameter values to use in simulation. Any required parameters whose names are not in this list will be assigned their default values. To see the parameter names and structures required, run with 'params = NULL' (the default) and examine the '$params' element of the output.
covariates	a dataframe of covariate values to use in simulation, or NULL to simulate values. To see the covariate names and structures required, run with 'covariates = NULL' (the default) and examine the '$covariates' element of the output.
seed	random seed. NULL uses (and updates) the existing RNG state. Other values do not update the global RNG state.
ragged_rep	logical: create data with variable (TRUE) or constant (FALSE) numbers of visits per unit. If TRUE, approximately half of units will be missing approximately half of 'n_rep' visits. Intended primarily for development purposes (bug-checking models with missing data).
missing_seasons	logical; relevant only if n_season is greater than 1. create data with variable (TRUE) or constant (FALSE) numbers of seasons per series (TRUE). If TRUE, approximately half of series will be missing their even-numbered seasons.

Value

A named list with the observation matrix/array ($obs), the unit covariate dataframe(s) ($unit_covs), the event covariate list ($event_covs), the parameters used in simulation ($params) and the covariate list used in simulation ($covariates). If rep_constant is TRUE, then $event_covs will be NULL.

Examples

```
simulate_flocker_data()
```

Index