

Package: crywolf (via r-universe)

October 16, 2026

Title Bayesian False-Positive Occupancy Models with Calibration Checks

Version 1.2.0

Description Fits single-season occupancy models that correct for false positives from automated detectors, such as acoustic classifiers, using the multiple-methods (site-confirmation) design of Miller et al. (2011) <[doi:10.1890/10-1396.1](https://doi.org/10.1890/10-1396.1)>, in which a subset of detections is verified. Models are written in 'Stan' and fitted through 'cmdstanr'. The false-positive rate can be constant or depend on site covariates, and the true-detection process accepts observation covariates and a recorder random effect. The package also simulates data under homogeneous, unstructured, and structured false-positive regimes and runs simulation-based calibration checks, so users can ask whether the correction keeps nominal interval coverage for their own design and how much verification it needs.

License GPL (>= 3)

URL <https://github.com/diogoprov/crywolf>,
<https://provetelab.org/crywolf/>

BugReports <https://github.com/diogoprov/crywolf/issues>

Depends R (>= 4.2)

Imports cmdstanr (>= 0.8.0), loo, posterior, purrr (>= 1.0.0), stats, tibble, tools, utils

Suggests bayesplot, furrr, future, ggplot2, knitr, progressr, rmarkdown, testthat (>= 3.0.0), unmarked

VignetteBuilder knitr

Remotes stan-dev/cmdstanr

Additional_repositories <https://stan-dev.r-universe.dev>

SystemRequirements CmdStan
(<https://mc-stan.org/users/interfaces/cmdstan>)

Config/testthat/edition 3

Encoding UTF-8
Roxygen list(markdown = TRUE)
Config/roxygen2/version 8.1.1
LazyData true
Repository https://diogoprov.r-universe.dev
Date/Publication 2026-10-16 02:21:58 UTC
RemoteUrl https://github.com/diogoprov/crywolf
RemoteRef HEAD
RemoteSha 62633f489c6cf1cf533c71eee4f5408250ea7389

Contents

curiango	2
fp_calibrate	4
fp_calibrate_fit	5
fp_compile	7
fp_data	8
fp_data_wide	9
fp_occu	10
fp_occu-methods	13
fp_ppc	15
fp_priors	17
fp_stan_file	18
fp_verification_curve	18
plot.fp_calibration	19
simulate.fp_occu	20
simulate_fp	21
Index	24

curiango	<i>Passive acoustic monitoring of the Common Pauraque in the Pantanal</i>
----------	---

Description

Detections of the Common Pauraque (*Nyctidromus albicollis*) by two automated classifiers in one-minute recordings made at seven sites in the Pantanal wetlands and adjacent Cerrado, Mato Grosso do Sul, Brazil, between November 2023 and February 2025, with an independent set of recordings annotated by ear. The occupancy unit is the site-night, the occasion is the one-minute recording, and the annotated recordings are the verified method of the multiple-methods design.

Usage

curiango

Format

A list with three data frames:

minutes One row per one-minute recording (97,806 rows): `site_night` (site-night identifier, the occupancy unit), `site` (recorder site, one of seven), `time` (start of the recording), `conf_custom` and `conf_birdnet` (confidence scores of a custom-trained classifier and of the BirdNET global model v2.4, 0 when the classifier returned nothing), and the acoustic indices `bio` (Bioacoustic Index) and `aci` (Acoustic Complexity Index).

nights One row per site-night (1,753 rows): `site_night`, `site`, `night` (date of the evening), and the nightly means of temperature (`temp`, °C) and relative humidity (`rh`, %), and the fraction of the lunar disc illuminated (`moon`).

verified One row per annotated recording (1,199 rows): `site_night`, `site`, `time`, and `y`, 1 if any 3-s segment of the recording held the species.

Details

The annotated recordings are an independent ground-truth test set of about 1% of the recordings, chosen in proportion to the number of recordings at each site, and annotated without reference to the classifiers (Piatti et al., 2026). They can therefore serve as the verified method of `fp_data()`. A detection by a classifier is a confidence score at or above a threshold; Piatti et al. (2026) used 0.40 (custom) and 0.10 (BirdNET) to maximize F1, and 0.95 for both to maximize precision.

The source table has one to 20 rows (detection windows) per recording, and a classifier's score is non-zero only in the windows where it detected the species. This dataset keeps one row per recording, with the maximum score over its windows, so a recording is a detection when any of its windows reaches the threshold. The script that builds it from the public deposit is `data-raw/curiango.R` in the package repository.

Source

Piatti, L., de Sousa, D. L. H., Oliveira, B. S., de Melo, A. V., Sugai, L. S. M., & Provete, D. B. (2026). Data and code for: Methodological choices influence ecological inference in passive acoustic monitoring of a Neotropical nightjar. Zenodo. doi:10.5281/zenodo.21795898. Licence CC BY 4.0.

References

Piatti, L., de Sousa, D. L. H., Oliveira, B. S., de Melo, A. V., Sugai, L. S. M., & Provete, D. B. (2026). Methodological choices influence ecological inference in passive acoustic monitoring of a Neotropical nightjar. *Journal of Avian Biology*, in press.

Examples

```
str(curango, max.level = 1)

# detections by the custom classifier at the MaxF1 threshold
mean(curango$minutes$conf_custom >= 0.40)
```

Description

Simulates many datasets from a design with `simulate_fp()`, fits one or more false-positive models to each with `fp_occu()`, and scores how often the credible intervals contain the true values and how biased the posterior means are. Use it before collecting data, to ask whether the correction will be calibrated for the number of sites, occasions, and verified sites you can afford, and whether a constant false-positive rate is enough under the confusion you suspect.

Usage

```
fp_calibrate(
  design,
  models = list(constant = ~1),
  occ = ~x,
  det = ~1,
  reps = 50,
  prob = 0.95,
  base_seed = 1L,
  parallel = FALSE,
  fit_args = list(chains = 4, parallel_chains = 1, iter_warmup = 1000, iter_sampling =
    1000, adapt_delta = 0.95)
)
```

Arguments

design	A data frame (or list) whose columns are arguments of <code>simulate_fp()</code> ; each row is one cell of the design. Arguments not given take the defaults of <code>simulate_fp()</code> .
models	Named list of formulas for the false-positive rate, one model per element, as in the fp argument of <code>fp_occu()</code> . The default fits only the constant-rate model.
occ, det	Formulas for occupancy and true detection, shared by all models. Simulated data have the occupancy covariate x and the detection covariates bio and aci.
reps	Number of replicate datasets per cell. Each costs one Stan fit per model. Fifty replicates give a Monte Carlo standard error of about 0.03 for a coverage of 0.95.
prob	Width of the credible intervals scored.
base_seed	Integer. Replicate j (counting over cells and replicates) uses seed base_seed + j, so results are reproducible and every model sees exactly the same datasets.
parallel	Logical. Fit replicates in parallel with 'furr'. Set the plan first, e.g. <code>future::plan(future::multisession, workers = 8)</code> , and keep <code>parallel_chains = 1</code> in <code>fit_args</code> . In interactive sessions with 'progressr' installed, a progress bar shows the estimated time remaining, also when fitting in parallel.
fit_args	List of further arguments for <code>fp_occu()</code> . Unless set here, <code>random_recorder</code> is TRUE only when the design simulates a recorder effect (<code>sd_recorder > 0</code>): fitting one whose true standard deviation is zero produces divergent transitions.

Details

The scored quantities are the occupancy intercept and slope (`occ_(Intercept)`, `occ_x`), the false-positive rate averaged over sites (`p10_bar`, against the realized mean rate), the detection probability of the verified method (`p_verified`), and, for models that include the confusing gradient `w`, its effect on the false-positive rate (`fp_w`).

Value

A tibble of class `fp_calibration`, one row per replicate, model, and scored quantity, with the truth, the posterior mean and interval, whether the interval covered the truth, and sampler diagnostics. Use `base::summary()` for coverage, its Monte Carlo standard error, bias, and RMSE per cell and model.

See Also

`fp_verification_curve()`, `simulate_fp()`

Examples

```
## Not run:
design <- data.frame(p10 = 0.05, conf_effect = c(0, 1.5), conf_cor = 0.7)
cal <- fp_calibrate(design,
                    models = list(constant = ~ 1, fp_on_x = ~ x),
                               occ = ~ x, reps = 50)

summary(cal)

## End(Not run)
```

`fp_calibrate_fit`

Calibration of a fitted model on data like the real ones

Description

Asks whether the credible intervals of a model would cover the truth in data like the ones it was fitted to. The function simulates datasets from the fit with `simulate.fp_occu()`, so that they keep the design of the real data (sites, covariates, occasions, verification, recorders) and follow the fitted model, refits each candidate model to every dataset, and scores the coverage and bias of their intervals, as `fp_calibrate()` does for designed simulations.

Usage

```
fp_calibrate_fit(
  fit,
  models = list(fitted = fit$formulas$fp),
  reps = 50,
  truth = c("mean", "posterior"),
  prob = 0.95,
  base_seed = 1L,
  parallel = FALSE,
```

```
fit_args = list(chains = 4, parallel_chains = 1, iter_warmup = 1000, iter_sampling =
  1000, adapt_delta = 0.95)
)
```

Arguments

fit	An fp_occu object: the generating model.
models	Named list of false-positive formulas (or NULL for no false positives) to refit. The default refits the generating model. The occupancy and detection formulas, the recorder effect, and the priors are those of fit.
reps	Number of simulated datasets.
truth	"mean" (default) or "posterior"; see simulate.fp_occu() .
prob	Width of the credible intervals.
base_seed	Seeds are base_seed + 1, ..., base_seed + reps.
parallel, fit_args	As in fp_calibrate() .

Details

The scored quantities are the occupancy coefficients, the detection coefficients, the intercept of the verified method, the mean false-positive rate over sites (p_{10_bar} , when both the generating and the refitted model have false positives), and the slopes of the false-positive rate in the refitted model. A slope that is not in the generating model has a true value of zero.

The most useful comparison pits the model you fitted against a simpler one. For example, if a model with $fp = \sim temp + rh$ estimated clear effects of the covariates on the false-positive rate, refitting $fp = \sim 1$ to data simulated from it shows how much a constant rate would bias the occupancy effects at your sample size and verification effort. This answers the question for the fitted structure only: a dependence of the false-positive rate on something you did not measure is not part of the generating model.

Each replicate refits every model, so the cost is that of reps times the number of models fits of the size of the real data. Use parallel, and threads_per_chain in fit_args for large datasets.

Value

An object of class fp_calibration, as from [fp_calibrate\(\)](#), with a single design cell.

See Also

[fp_calibrate\(\)](#), [simulate.fp_occu\(\)](#)

Examples

```
## Not run:
sim <- simulate_fp(nsite = 300, nocc = 20, conf_effect = 1.5,
  conf_cor = 0.7, seed = 1)
m_x <- fp_occu(sim$data, occ = ~ x, fp = ~ x, random_recorder = FALSE)
cal <- fp_calibrate_fit(m_x, models = list(constant = ~ 1, fp_on_x = ~ x),
  reps = 20)
```

```
summary(cal)
plot(cal, parameter = "occ_x")

## End(Not run)
```

fp_compile*Compile (or load) the Stan model*

Description

Compiles the false-positive occupancy model the first time it is needed and keeps the executable in the user cache directory (`tools::R_user_dir("crywolf", "cache")`), so later sessions load it without recompiling. `fp_occu()` calls this function itself; call it directly to compile ahead of time, for example before launching parallel workers in `fp_calibrate()`.

Usage

```
fp_compile(force = FALSE, quiet = TRUE)
```

Arguments

<code>force</code>	Logical. Recompile even if a current executable exists.
<code>quiet</code>	Logical. Suppress compiler output.

Details

The model is compiled with threading enabled (`stan_threads`), so that `fp_occu()` can split the likelihood of each chain among several threads (argument `threads_per_chain`).

Value

A `CmdStanModel` object (invisibly).

Examples

```
## Not run:
fp_compile()

## End(Not run)
```

fp_data

*Assemble data for a false-positive occupancy model***Description**

Collects the automated detections, the verified detections, and the site table into one object that `fp_occu()` can fit. All tables are in long format, one row per occasion, which accommodates unequal numbers of occasions per site without padding. Use `fp_data_wide()` if your data are detection matrices, as in 'unmarked'.

Usage

```
fp_data(
  auto,
  sites,
  verified = NULL,
  site = "site",
  y = "y",
  recorder = NULL,
  confirmed = NULL
)
```

Arguments

auto	Data frame of the automated method (the one that can return false positives), one row per site and occasion. It needs a site column and a 0/1 detection column, and may hold observation covariates for the true-detection process and a recorder (or observer) identifier.
sites	Data frame with one row per site: the site identifier plus any site covariates used in the occupancy (occ) or false-positive (fp) formulas of <code>fp_occu()</code> . Sites with no automated occasions are allowed and contribute only through their verified data, if any.
verified	Optional data frame of the verified method (detections checked by an expert and therefore free of false positives), one row per site and occasion, with the same site and detection columns. Without it the false-positive rate is identified only by the priors, and <code>fp_occu()</code> warns about it.
site, y	Names of the site and detection columns, shared by all tables.
recorder	Optional name of a column of auto that identifies the recorder or observer, used for a random intercept on true detection.
confirmed	Optional names of two columns of auto holding, for the occasions whose observations were checked by an expert, the number of observations confirmed as the target species and the number confirmed as something else, in that order (the observation-confirmation design of Chambert et al. 2015). Use NA in both for occasions that were not checked. All observations of a checked occasion must have been checked, so an occasion with a detection has at least one confirmed observation, and one without has none. See Details of <code>fp_occu()</code> .

Value

An object of class `fp_data`: a list with the three tables (rows with a missing detection dropped), the column names, and the vector `known_present`, which flags sites with at least one verified detection or confirmed true observation.

References

Chambert, T., Miller, D. A. W., & Nichols, J. D. (2015). Modeling false positive detections in species occurrence data under different study designs. *Ecology*, 96, 332–339. doi:10.1890/14-1507.1

See Also

`fp_data_wide()`, `fp_occu()`, `simulate_fp()`

Examples

```
sim <- simulate_fp(nsite = 50, nocc = 5, seed = 1)
sim$data

# the same structure built by hand
d <- fp_data(auto = sim$data$auto, sites = sim$data$sites,
             verified = sim$data$verified, recorder = "recorder")
d
```

fp_data_wide

Assemble data from detection matrices

Description

Convenience wrapper around `fp_data()` for data stored as site-by-occasion matrices, the layout used by 'unmarked'. Missing values (NA) mark occasions that were not sampled, so sites may differ in effort.

Usage

```
fp_data_wide(
  y_auto,
  y_verified = NULL,
  site_covs = NULL,
  obs_covs = NULL,
  recorder = NULL
)
```

Arguments

y_auto	Matrix of automated detections (sites in rows, occasions in columns), coded 0/1/NA.
y_verified	Optional matrix of verified detections with the same number of rows; unverified sites are rows of NA.
site_covs	Optional data frame of site covariates, one row per site, in the same order as the rows of y_auto.
obs_covs	Optional named list of matrices, each the same size as y_auto, holding observation covariates of the automated method.
recorder	Optional vector, one value per site, identifying the recorder or observer.

Value

An object of class fp_data.

See Also

[fp_data\(\)](#)

Examples

```
set.seed(1)
ya <- matrix(rbinom(40, 1, 0.3), 10, 4)
yv <- matrix(NA, 10, 2); yv[1:3, ] <- rbinom(6, 1, 0.5)
fp_data_wide(ya, yv, site_covs = data.frame(x = rnorm(10)))
```

fp_occu

Fit a Bayesian false-positive occupancy model

Description

Fits the multiple-methods false-positive occupancy model of Miller et al. (2011) in Stan. Detections from an automated method may be false positives, whereas detections from a verified subset are certain. The occupancy probability depends on site covariates, true detection on observation (or site) covariates and an optional recorder random intercept, and the false-positive rate is either constant or a function of site covariates.

Usage

```
fp_occu(
  data,
  occ = ~1,
  det = ~1,
  fp = ~1,
  random_recorder = !is.null(data$recorder),
  random_fp = FALSE,
```

```

    priors = fp_priors(),
    chains = 4,
    parallel_chains = chains,
    threads_per_chain = 1,
    iter_warmup = 1000,
    iter_sampling = 1000,
    adapt_delta = 0.95,
    seed = NULL,
    refresh = NULL,
    ...
  )

```

Arguments

<code>data</code>	An <code>fp_data</code> object from <code>fp_data()</code> , <code>fp_data_wide()</code> , or <code>simulate_fp()</code> .
<code>occ</code>	One-sided formula for the logit of occupancy, using columns of <code>data\$sites</code> .
<code>det</code>	One-sided formula for the logit of the true-detection probability of the automated method. It may use columns of <code>data\$auto</code> and of <code>data\$sites</code> . It must keep the intercept.
<code>fp</code>	One-sided formula for the logit of the false-positive rate, using columns of <code>data\$sites</code> . ~ 1 (the default) fits a constant rate; $\sim w$ lets the rate change along w . Use <code>NULL</code> to fit a standard occupancy model without false positives. The intercept is fixed by the model (see Details) and cannot be removed.
<code>random_recorder</code>	Logical. Add a recorder random intercept to true detection? Defaults to <code>TRUE</code> when data has a recorder column. Data from <code>simulate_fp()</code> always have one; with its default <code>sd_recorder = 0</code> set this to <code>FALSE</code> , because a random effect whose true variance is zero causes divergent transitions.
<code>random_fp</code>	Logical. Add a site random intercept to the logit of the false-positive rate, for rates that vary among sites in ways the covariates in <code>fp</code> do not capture. <code>FALSE</code> by default; see the section "Site random effect on the false-positive rate" before turning it on.
<code>priors</code>	Prior settings from <code>fp_priors()</code> .
<code>chains</code> , <code>iter_warmup</code> , <code>iter_sampling</code> , <code>adapt_delta</code> , <code>seed</code>	Passed to the <code>\$sample()</code> method of 'cmdstanr'.
<code>parallel_chains</code>	Number of chains run in parallel.
<code>threads_per_chain</code>	Number of threads each chain uses to compute the likelihood, which is summed over sites in parallel with Stan's <code>reduce_sum()</code> . The default, 1, runs each chain on one core. Larger values speed up fits with many sites or occasions; keep <code>parallel_chains * threads_per_chain</code> at or below the number of cores. The results do not depend on it beyond floating-point rounding.
<code>refresh</code>	How often each chain reports its progress, in iterations. The default, <code>NULL</code> , reports every 10% of the iterations in interactive sessions and nothing otherwise; <code>0</code> silences it. Warm-up iterations are usually slower than sampling ones, so the first half of the run takes more than half of the time.

... Further arguments passed to `$sample()`.

Details

Identifiability. The intercept of the false-positive rate equals the intercept of true detection minus a positive gap, $\gamma_0 = \alpha_1 - \Delta$, so that at the reference values of the covariates (all zero) a true detection is more likely than a false positive. This removes the label switching between the occupied and unoccupied states. Centre the covariates, so that the reference is an average site; `simulate_fp()` already returns them standardized.

Verified data. The verified detections anchor the false-positive rate. With none, the model still runs, but the rate is informed only by the priors and the gap, and the fit should not be trusted.

Observation-confirmation design. When data has confirmed occasions (argument `confirmed` of `fp_data()`), the model follows the observation-confirmation design of Chambert et al. (2015): an expert checks every observation of some automated occasions and finds only false ones, only true ones, both, or none. False observations can then occur at occupied sites too, and `det` and `fp` model s_1 and s_0 , the probabilities of at least one true and at least one false observation on an occasion; a site-level detection at an occupied site has probability $p_{11} = 1 - (1 - s_1)(1 - s_0)$, and `p11_ref` reports it at the reference. The confirmed occasions identify s_0 directly, so verified data are optional; they can be combined with this design. The gap still applies, so that $s_1 > s_0$ at the reference.

Site random effect on the false-positive rate. With `random_fp = TRUE` the logit of the false-positive rate of site i gains a term $e_i \sim \text{Normal}(0, \sigma_{fp})$, reported as `sd_fp_site`, with a half-Normal prior (`fp_priors(fp_site_sd = )`). In simulations with 300 sites, 20 automated occasions, and 30% of sites verified, it restored the coverage of the mean false-positive rate (`p10_bar`) when the rate varied among sites at random (from 0.00 and 0.55 with a constant rate to 0.95) and cost nothing when the rate was homogeneous. Use it with care:

- It adds one parameter per site, and fits took 6 to 9 times longer.
- It mixes less well: in those simulations the largest R-hat exceeded 1.01 in a quarter to three quarters of the fits. Check the diagnostics, and run longer chains (`iter_warmup`, `iter_sampling`) when they are poor.
- Each site has a single latent state, so a site effect on false positives is told apart from occupancy mainly by the verified data; with few occasions or little verification it may be weakly identified.
- It does not fix false positives that follow an environmental gradient. Those bias the occupancy effects, and the remedy is to put the occupancy covariates, and any measured correlate of the confusion, in `fp` (see below). Prefer covariates, and add the random effect for the variation that remains.

Predictions with `predict(process = "fp")` are for a site with $e_i = 0$, a typical site.

When to model the false-positive rate. In the simulations behind this package, a constant rate gave calibrated intervals for the occupancy slope when false positives were homogeneous; random variation of the rate among sites made the estimate of the rate unreliable but largely spared the slope. When the rate tracked an environmental gradient correlated with the occupancy covariate and the confusion was strong, the slope was biased and verifying more sites did not fix it. Placing the occupancy covariate in `fp` recovered part of the calibration, and placing the gradient itself recovered all of it. Use `fp_calibrate()` to check your own design.

Parameter names. Coefficients are labelled by process and column of the design matrix: `occ_*` (occupancy), `det_*` (true detection), `fp_*` (false positive, where `fp_(Intercept)` is γ_0), plus `fp_gap` (Δ), `sd_recorder`, `p_verified` (detection probability of the verified method), `p11_ref` and `p10_ref` (true- and false-positive probabilities at the reference), and `p10_bar` (false-positive probability averaged over the sites).

Value

An object of class `fp_occu`, with methods for `base::print()`, `base::summary()`, `stats::coef()`, `stats::predict()`, `loo::loo()`, and `posterior::as_draws_df()`. The 'cmdstanr' fit is in element `fit`.

References

- Chambert, T., Miller, D. A. W., & Nichols, J. D. (2015). Modeling false positive detections in species occurrence data under different study designs. *Ecology*, 96, 332–339. doi:10.1890/14-1507.1
- Miller, D. A., Nichols, J. D., McClintock, B. T., Grant, E. H. C., Bailey, L. L., & Weir, L. A. (2011). Improving occupancy estimation when two types of observational error occur: non-detection and species misidentification. *Ecology*, 92, 1422–1428. doi:10.1890/101396.1

See Also

`fp_data()`, `fp_priors()`, `fp_calibrate()`

Examples

```
## Not run:
sim <- simulate_fp(nsite = 300, nocc = 20, conf_effect = 1.5,
                  conf_cor = 0.7, seed = 42)

# constant false-positive rate (the simulated recorders have no effect,
# so the recorder term is left out)
m0 <- fp_occu(sim$data, occ = ~ x, fp = ~ 1, random_recorder = FALSE)
# false-positive rate that tracks the occupancy covariate
m1 <- fp_occu(sim$data, occ = ~ x, fp = ~ x, random_recorder = FALSE)

coef(m0)["occ_x"]; coef(m1)["occ_x"] # truth: -0.8
loo::loo_compare(loo(m0), loo(m1))

## End(Not run)
```

Description

Summaries, point estimates, predictions, and model comparison for objects returned by `fp_occu()`.

Usage

```
## S3 method for class 'fp_occu'
print(x, digits = 3, ...)

## S3 method for class 'fp_occu'
summary(object, prob = 0.95, ...)

## S3 method for class 'summary.fp_occu'
print(x, digits = 3, ...)

## S3 method for class 'fp_occu'
coef(object, type = c("all", "occ", "det", "fp"), ...)

## S3 method for class 'fp_occu'
predict(
  object,
  newdata,
  process = c("occ", "fp"),
  prob = 0.95,
  summary = TRUE,
  ...
)

## S3 method for class 'fp_occu'
loo(x, ...)

## S3 method for class 'fp_occu'
as_draws_df(x, ...)
```

Arguments

x, object	An fp_occu object.
digits	Number of digits to print.
...	Unused, or passed on (see each method).
prob	Width of the equal-tailed credible interval.
type	Which coefficients to return: "all" (default), "occ", "det", or "fp".
newdata	Data frame with the covariates of the requested process. If missing, the sites used in the fit.
process	"occ" for the occupancy probability or "fp" for the false-positive rate, both at the site level.
summary	Logical. Return a summary per row (TRUE) or the matrix of posterior draws (rows are draws, columns are rows of newdata)?

Value

summary() returns an object of class summary.fp_occu whose element table is a tibble with the

posterior mean, standard deviation, credible interval, R-hat, and effective sample sizes of every parameter.

coef() returns a named vector of posterior means.

predict() returns a data frame with the posterior mean and credible interval of the probability for each row of newdata, or a matrix of draws.

loo() returns a psis_loo object from 'loo'. The pointwise log-likelihood is computed per site, with the latent state marginalized, so the comparison is leave-one-site-out.

as_draws_df() returns the posterior draws with the readable parameter names.

Examples

```
## Not run:
sim <- simulate_fp(nsite = 200, nocc = 10, seed = 1)
m <- fp_occu(sim$data, occ = ~ x, fp = ~ 1, random_recorder = FALSE)
m                                     # posterior summary and diagnostics
summary(m, prob = 0.9)$table
coef(m, "occ")
predict(m, newdata = data.frame(x = c(-1, 0, 1)), process = "occ")
loo(m)
draws <- as_draws_df(m)

## End(Not run)
```

fp_ppc

Posterior predictive check for a false-positive occupancy model

Description

Replicates the detection data from the posterior of an `fp_occu()` fit and compares observed and replicated data with a discrepancy statistic, giving a Bayesian p-value. Detections are summed per site, so the check asks whether the model reproduces how many detections each site gets.

Usage

```
fp_ppc(
  object,
  method = c("automated", "verified"),
  fit_stat = c("freeman-tukey", "chi-squared"),
  ndraws = 1000,
  seed = NULL
)
```

Arguments

object	An fp_occu object.
method	Which data to check: "automated" (default; every site) or "verified" (verified sites only).

fit_stat	Discrepancy statistic: "freeman-tukey" (default), $\sum_i (\sqrt{y_i} - \sqrt{E_i})^2$, or "chi-squared", $\sum_i (y_i - E_i)^2 / E_i$, where y_i is the number of detections at site i and E_i its expected value given the parameters and the latent state.
ndraws	Number of posterior draws to use, sampled without replacement. NULL uses all of them.
seed	Optional integer seed for the subsampling and replication.

Details

The latent occupancy state is marginalized in the Stan model, so for each posterior draw z_i is first drawn from its conditional posterior given the data and the parameters,

$$\Pr(z_i = 1 \mid \mathbf{y}_i, \theta) = \frac{\psi_i f(\mathbf{y}_i \mid z_i = 1, \theta)}{\psi_i f(\mathbf{y}_i \mid z_i = 1, \theta) + (1 - \psi_i) f(\mathbf{y}_i \mid z_i = 0, \theta)},$$

which is 1 at sites with a verified detection. Given z_i , automated detections are replicated with probability $z_i p_{11} + (1 - z_i) p_{10}$ and verified detections with probability $z_i r_{11}$. The discrepancy is computed for the observed and the replicated data with the same z and parameters, and the Bayesian p-value is the proportion of draws in which the replicated discrepancy exceeds the observed one. Values near 0 or 1 indicate lack of fit; values near 0.5 indicate that the observed data are typical of the fitted model.

Under the observation-confirmation design, the conditional probability of z_i also uses the confirmed occasions, and "automated" checks the occasions that were not confirmed.

A p-value near 0.5 does not show that the false-positive process is correctly specified: a check of detection counts can have little power against a false-positive rate that tracks an unmeasured gradient. Use `fp_calibrate()` to ask whether the intervals of a given model are calibrated for a design.

The elements `y` and `yrep` follow the conventions of the 'bayesplot' package, so its `ppc_*()` functions can be used directly, for example `bayesplot::ppc_bars(ppc$y, ppc$yrep)`.

Value

An object of class `fp_ppc`, a list with `y` (observed detections per site), `yrep` (matrix of replicated detections, draws in rows and sites in columns), `fit_obs` and `fit_rep` (discrepancy of the observed and replicated data, one value per draw), `p_value`, `site` (the site identifiers of the columns), `method`, and `fit_stat`.

See Also

`fp_occu()`, `fp_calibrate()`

Examples

```
## Not run:
sim <- simulate_fp(nsite = 200, nocc = 10, seed = 1)
m <- fp_occu(sim$data, occ = ~ x, fp = ~ 1, random_recorder = FALSE)
ppc <- fp_ppc(m, ndraws = 500, seed = 1)
ppc
bayesplot::ppc_bars(ppc$y, ppc$yrep)
```

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

fp_priors	<i>Prior settings for fp_occu()</i>
-----------	-------------------------------------

Description

Normal priors on the logit scale. The defaults are weakly informative for standardized covariates.

Usage

```
fp_priors(
  intercept_sd = 1.5,
  slope_sd = 2.5,
  gap_mean = 2,
  gap_sd = 1.5,
  recorder_sd = 2,
  fp_site_sd = 1
)
```

Arguments

intercept_sd	Standard deviation of the Normal(0, sd) priors on the intercepts of occupancy, true detection, and verified detection.
slope_sd	Standard deviation of the Normal(0, sd) priors on all slopes, including those of the false-positive rate.
gap_mean, gap_sd	Mean and standard deviation of the Normal prior, truncated at zero, on the gap between the true-detection and false-positive intercepts. The default, Normal(2, 1.5), puts the false-positive rate at the reference roughly an order of magnitude below true detection, while leaving room for the data.
recorder_sd	Scale of the half-Normal(0, sd) prior on the standard deviation of the recorder random intercept.
fp_site_sd	Scale of the half-Normal(0, sd) prior on the standard deviation of the site random intercept on the false-positive rate (fp_occu(random_fp = TRUE)).

Value

A list of class fp_priors.

Examples

```
fp_priors()
fp_priors(slope_sd = 1)
```

fp_stan_file	<i>Path to the Stan program</i>
--------------	---------------------------------

Description

Path to the Stan program

Usage

```
fp_stan_file()
```

Value

The path to the Stan file shipped with the package, for users who want to read the model or modify a copy of it.

Examples

```
fp_stan_file()
```

fp_verification_curve	<i>Coverage as a function of verification effort</i>
-----------------------	--

Description

Runs `fp_calibrate()` over a grid of verification fractions and prints the coverage of the occupancy slope against the fraction of sites verified. If the constant-rate model does not approach nominal coverage as verification grows, the false positives are probably structured, and the remedy is to model the false-positive rate (for instance on the occupancy covariate), not to verify more.

Usage

```
fp_verification_curve(
  ...,
  frac_verified = c(0.1, 0.3, 0.5, 0.7, 0.9),
  models = list(constant = ~1),
  occ = ~x,
  det = ~1,
  reps = 50,
  base_seed = 1L,
  parallel = FALSE,
  fit_args = list(chains = 4, parallel_chains = 1, iter_warmup = 1000, iter_sampling =
    1000, adapt_delta = 0.95)
)
```

Arguments

... Arguments of `simulate_fp()` that define the scenario (held fixed), e.g. `p10 = 0.05`, `conf_effect = 1.5`.
`frac_verified` Fractions of sites verified to sweep.
`models, occ, det, reps, base_seed, parallel, fit_args`
 As in `fp_calibrate()`.

Value

The `fp_calibration` object (invisibly), after printing the curve.

Examples

```
## Not run:
fp_verification_curve(p10 = 0.05, conf_effect = 1.5, conf_cor = 0.7,
                     models = list(constant = ~ 1, fp_on_x = ~ x),
                     reps = 50)

## End(Not run)
```

plot.fp_calibration	<i>Plot the results of a calibration check</i>
---------------------	--

Description

Plots the coverage (or bias) of one scored quantity against the column of the design that varies, one colour per model. Coverage is shown with an interval of two Monte Carlo standard errors and a dashed line at the nominal level; bias, with a dashed line at zero.

Usage

```
## S3 method for class 'fp_calibration'
plot(x, parameter = "occ_x", type = c("coverage", "bias"), by = NULL, ...)
```

Arguments

`x` An `fp_calibration` object from `fp_calibrate()` or `fp_verification_curve()`.
`parameter` Scored quantity to plot, by default the occupancy slope ("occ_x"). See `fp_calibrate()` for the others.
`type` "coverage" (default) or "bias".
`by` Name of the design column for the horizontal axis. By default, the first design column that varies among cells; further varying columns become panels. With no varying column, the models are on the axis.
 ... Unused.

Value

A 'ggplot2' object, which can be modified with the usual + syntax.

Examples

```
## Not run:
cal <- fp_verification_curve(p10 = 0.05, conf_effect = 1.5, conf_cor = 0.7,
                             models = list(constant = ~ 1, fp_on_x = ~ x),
                             reps = 50)

plot(cal)
plot(cal, type = "bias")

## End(Not run)
```

simulate.fp_occu	<i>Simulate data from a fitted model</i>
------------------	--

Description

Generates new detections from an fp_occu fit, keeping everything else of the original data: the sites and their covariates, the automated occasions with their observation covariates and recorders, which sites and occasions were verified, and which occasions were checked under the observation-confirmation design. Only the latent occupancy states and the detections are drawn anew, from the fitted model.

Usage

```
## S3 method for class 'fp_occu'
simulate(object, nsim = 1, seed = NULL, truth = c("mean", "posterior"), ...)
```

Arguments

object	An fp_occu object.
nsim	Number of datasets.
seed	Optional integer seed.
truth	Parameter values to simulate from: "mean" (default) uses the posterior means for every dataset; "posterior" uses a different random posterior draw for each dataset, which carries the uncertainty of the fit into the simulations.
...	Unused.

Details

The recorder effects, when the model has them, are kept at their values in the chosen draw (or their posterior means), so the simulated data have the same recorders with the same effects. Site effects on the false-positive rate (fp_occu(random_fp = TRUE)) are kept at their values in the draw with truth = "posterior"; with truth = "mean", whose site effects would be shrunk towards zero, new ones are drawn from a normal distribution with the posterior mean of sd_fp_site as standard deviation. Under the multiple-methods design, an automated occasion at an occupied site is a detection with probability p_{11} and at an unoccupied site with probability p_{10} ; verified occasions are detections with probability r_{11} at occupied sites only. Under the observation-confirmation design, true and false observations are drawn independently, as in [simulate_fp\(\)](#).

Value

A list of `nsim` elements, each a list with data (an `fp_data` object with the structure of the original data) and truth (the generating values on the scale of the output of `fp_occu()`, the latent states z , the false-positive rate of each site `p10_site`, and its mean `p10_bar`).

See Also

`fp_calibrate_fit()`, `simulate_fp()`

Examples

```
## Not run:
sim <- simulate_fp(nsite = 200, nocc = 10, seed = 1)
m <- fp_occu(sim$data, occ = ~ x, fp = ~ 1, random_recorder = FALSE)
new <- simulate(m, nsim = 2, seed = 1)
new[[1]]$data
new[[1]]$truth$p10_bar

## End(Not run)
```

simulate_fp

Simulate data for a false-positive occupancy model

Description

Generates one dataset from known parameters under the multiple-methods design: an automated method that returns true detections and false positives at every site, and a verified method, free of false positives, at a random subset of sites. The false-positive rate can be homogeneous, vary at random among sites (unstructured), or track a latent gradient correlated with the occupancy covariate (structured confusion, as when a soundalike species becomes commoner along the same gradient as the target).

Usage

```
simulate_fp(
  nsite = 300,
  nocc = 20,
  psi = 0.5,
  b1 = -0.8,
  p11 = 0.6,
  p10 = 0.05,
  sigma_p10 = 0,
  p10_conc = NULL,
  conf_effect = 0,
  conf_cor = 0,
  r11 = 0.6,
  frac_verified = 0.3,
```

```

nver = 3,
a_bio = 0,
a_aci = 0,
sd_recorder = 0,
nrec = 5,
frac_confirmed = 0,
seed = NULL
)

```

Arguments

nsite	Number of sites.
nocc	Number of automated occasions per site.
psi	Occupancy probability at the mean of the covariate.
b1	Occupancy slope on the standardized covariate x .
p11	True-detection probability of the automated method at the mean of the detection covariates.
p10	False-positive probability of the automated method (its base value; the realized mean over sites is returned as <code>truth\$p10_marginal</code>).
sigma_p10	Among-site standard deviation of the logit of the false-positive rate (unstructured, logit-normal heterogeneity). 0 for none.
p10_conc	Optional concentration of a Beta distribution for the site-level false-positive rate, with mean <code>p10</code> (unstructured, aggregated heterogeneity; small values push rates towards 0 or 1). Overrides <code>sigma_p10</code> .
conf_effect	Slope of the logit of the false-positive rate on the confusing gradient w (structured confusion). 0 for none.
conf_cor	Correlation between w and the occupancy covariate x .
r11	Detection probability of the verified method.
frac_verified	Probability that a site is verified.
nver	Number of verified occasions per verified site.
a_bio, a_aci	Effects of two standardized observation covariates (<code>bio</code> , <code>aci</code>) on true detection. 0 by default.
sd_recorder	Standard deviation of a recorder random intercept on true detection. 0 by default.
nrec	Number of recorders.
frac_confirmed	Probability that an automated occasion is checked under the observation-confirmation design of Chambert et al. (2015). 0 (default) for none. See Details.
seed	Optional integer seed.

Details

The occupancy covariate x is standardized, so `b1` is the true slope exactly. The confusing gradient is $w = \rho x + \sqrt{1 - \rho^2} \epsilon$, with ϵ standard normal, so it has unit variance and correlation ρ with x . In practice w is not observed; it is returned in `sites` so that a model given the true structure can serve as a positive control. With the same seed, the function returns the same datasets

as `simulate_fp()` of the `fpcalib` research compendium (v0.1.0, [doi:10.5281/zenodo.23379100](https://doi.org/10.5281/zenodo.23379100)), with `nocc`, `frac_verified`, `nver`, `p10_conc`, `sd_recorder`, and `nrec` corresponding to its `K`, `verified_coverage`, `n_verified`, `p10_beta_conc`, `sig_u`, and `nobser` (checked by `dev/compare-simulate-fp.R` in the package repository).

With `frac_confirmed > 0`, false observations occur at occupied sites too: on each occasion, a true observation occurs with probability `p11` (and its covariate effects) at occupied sites, and a false one with the site's false-positive rate at any site, independently. The occasion is a detection if either occurs. Checked occasions get the columns `n_true` and `n_false` of `auto` (0 or 1; NA elsewhere). These draws come after all others, so the default design reproduces the same random numbers.

Value

A list of class `fp_sim` with elements `data` (an `fp_data` object whose `sites` table holds `x` and `w`, and whose `auto` table holds `bio`, `aci`, and `recorder`) and `truth` (the generating values, the realized mean false-positive rate `p10_marginal`, the realized correlation between `w` and `x`, and the latent states `z`).

See Also

`fp_occu()`, `fp_calibrate()`

Examples

```
# homogeneous false positives
s0 <- simulate_fp(nsite = 100, nocc = 10, seed = 1)
s0

# structured confusion
s1 <- simulate_fp(nsite = 100, nocc = 10, conf_effect = 1.5,
                  conf_cor = 0.7, seed = 1)
s1$truth$p10_marginal
```

Index

- * **datasets**
 - curiango, [2](#)
- as_draws_df.fp_occu (fp_occu-methods), [13](#)
- base::print(), [13](#)
- base::summary(), [5](#), [13](#)
- coef.fp_occu (fp_occu-methods), [13](#)
- curiango, [2](#)
- fp_calibrate, [4](#)
- fp_calibrate(), [5–7](#), [12](#), [13](#), [16](#), [18](#), [19](#), [23](#)
- fp_calibrate_fit, [5](#)
- fp_calibrate_fit(), [21](#)
- fp_compile, [7](#)
- fp_data, [8](#)
- fp_data(), [3](#), [9–13](#)
- fp_data_wide, [9](#)
- fp_data_wide(), [8](#), [9](#), [11](#)
- fp_occu, [10](#)
- fp_occu(), [4](#), [7–9](#), [13](#), [15](#), [16](#), [21](#), [23](#)
- fp_occu-methods, [13](#)
- fp_ppc, [15](#)
- fp_priors, [17](#)
- fp_priors(), [11](#), [13](#)
- fp_stan_file, [18](#)
- fp_verification_curve, [18](#)
- fp_verification_curve(), [5](#), [19](#)
- loo.fp_occu (fp_occu-methods), [13](#)
- loo::loo(), [13](#)
- plot.fp_calibration, [19](#)
- posterior::as_draws_df(), [13](#)
- predict.fp_occu (fp_occu-methods), [13](#)
- print.fp_occu (fp_occu-methods), [13](#)
- print.summary.fp_occu (fp_occu-methods), [13](#)
- simulate.fp_occu, [20](#)
- simulate.fp_occu(), [5](#), [6](#)
- simulate_fp, [21](#)
- simulate_fp(), [4](#), [5](#), [9](#), [11](#), [12](#), [19–21](#)
- stats::coef(), [13](#)
- stats::predict(), [13](#)
- summary.fp_occu (fp_occu-methods), [13](#)