Package: recruitlag (via r-universe)

October 15, 2026

Type Package

Title Can Episodic Recruitment be a Delayed Consequence of Continuous Reproduction?

Version 0.1.0

Description Tests whether an episodic recruitment record can be the delayed consequence of a reproductive record, under any distributed lag. A delay is a convolution, and a convolution is a weighted average, so the concentration in time of expected recruitment can never exceed that of reproduction: reproduction's own concentration is a ceiling. The package computes that ceiling by searching families of lag weights (pure delays, windows, geometric decay and Dirichlet draws), calibrates the pooled comparison against negative-binomial counts to show what it can and cannot refute, and runs the host-level (replicate-unit) test in which the null is the lag hypothesis exactly as it is asserted: expected recruitment on each unit is that unit's own lagged reproduction, scaled to its own total. Uncertainty on the exceedance comes by three routes, a cluster bootstrap over units, a Bayesian bootstrap (Rubin 1981 <doi:10.1214/aos/1176345338>) and a negative-binomial model of the counts fitted with 'Stan' through 'cmdstanr'; recruit_triage() asks whether a first sighting is a recruit or a plant returning from dormancy. Plotting functions draw the ceiling strip, the lag profiles against the observed counts, the calibration and a forest plot of the intervals. The census data on the epiphytic orchid Lepanthes eltoroensis analysed in the companion paper are included.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 4.1)

Imports ggplot2, rlang, stats, tibble, tools, utils

VignetteBuilder knitr

Suggests testthat (>= 3.0.0), knitr, rmarkdown, dplyr, tidyr, cmdstanr (>= 0.6.0), posterior

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

Config/testthat/edition 3

URL https://github.com/RaymondLTremblay/recruitlag

BugReports https://github.com/RaymondLTremblay/recruitlag/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Raymond L. Tremblay [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8588-4372>)

Maintainer Raymond L. Tremblay <raymond.tremblay@gmail.com>

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

Date/Publication 2026-10-15 19:17:47 UTC

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

RemoteRef HEAD

RemoteSha f0704386703f9d419e6b009f8afd2be88df5482f

Contents

caladenia_dormancy . . . 3
ceiling_calibration . . . 4
ceiling_draws_table . . . 5
convolve_lag . . . 6
expected_recruits . . . 8
flat_kernel . . . 9
host_lag_test . . . 10
host_matrices . . . 13
lag_ceiling . . . 14
lag_ceiling_bayesboot . . . 16
lag_ceiling_boot . . . 18
lag_ceiling_by . . . 20
lag_ceiling_stan . . . 22
lag_kernels . . . 25
lepanthes . . . 27
phi_moment . . . 28
plot.ceiling_calibration . . . 30
rain_forest . . . 31
rain_gini . . . 32
rain_indices . . . 34
rain_intervals . . . 35
rain_profiles . . . 37

rain_series 38
rain_sightings 39
rain_strip 40
rain_strips 40
rain_units 41
recruit_triage 42
regimes 45
series_from 45

Index **47**

`caladenia_dormancy` *Published dormancy, survival and detection for nine* Caladenia *species*

Description

Posterior means and 95% credible intervals from the multi-state capture-recapture analysis of Tremblay et al. (2009), Table 2: the probability of surviving from one year to the next, the probability of being seen above ground given survival, and the probability of dormancy, which is `p_survival * (1 - p_resight)`. These are the `s` and `r` that `recruit_triage()` uses to compute how many first sightings a record's own dormancy would produce.

Usage

```
caladenia_dormancy
```

Format

A data frame with 9 rows: `species`, `p_resight`, `p_resight_lo`, `p_resight_hi`, `p_survival`, `p_survival_lo`, `p_survival_hi`, `p_dormancy`, `p_dormancy_lo`, `p_dormancy_hi`, `n_plants`, `years`.

Scale and how to read it

All are annual probabilities. `p_resight` near 1 means a living plant is almost always above ground; near 0.6 means a third of the living plants are unseen in any year. Dormancy of two or more consecutive years was rare in every species (below 3% except *C. macroclavia*, 12%), which is why the expected-returns series in `recruit_triage()` falls off within three periods. The values come from one or a few populations per species and are not the species' constants; use them as a calibration, and prefer a capture-recapture estimate from the record itself when it is long enough to give one.

Source

Tremblay, R. L., Perez, M.-E., Larcombe, M., Brown, A., Quarmby, J., Bickerton, D., French, G. and Bould, A. (2009) Dormancy in *Caladenia*: a Bayesian approach to evaluating latency. *Australian Journal of Botany* 57: 340-350. doi:10.1071/BT08163

Examples

```
caladenia_dormancy
```

`ceiling_calibration` *Calibrate the pooled ceiling comparison against realised counts*

Description

The ceiling bounds the expected recruitment series. Observed recruits are counts about that expectation, and small clumped counts look episodic whatever their expectation. `ceiling_calibration()` simulates negative-binomial counts about two means that both obey the ceiling, a flat mean (N0) and the best-kernel mean rescaled to the observed total (N1), and reports how often the simulated series is at least as concentrated as the observed one. In the companion paper this shows that the pooled comparison has no power on its own; the host-level test (`host_lag_test()`) is where the refutation is made.

Usage

```
ceiling_calibration(x, vmr = NULL, nsim = 20000L)
```

Arguments

`x`	A `lag_ceiling` object.
`vmr`	Variance-to-mean ratio of the counts, used to set the negative-binomial size parameter through `phi = mu / (vmr - 1)`. Defaults to the variance-to-mean ratio of the observed recruit series.
`nsim`	Number of simulated series per null.

Value

An object of class `ceiling_calibration`: a data frame with one row per null (`N0 flat mean`, `N1 best-kernel mean`) giving `phi`, the median and 5 to 95% range of the simulated Gini and CV, and the probabilities `p_gini`, `p_cv` and `p_zeros` that a simulated series is at least as concentrated, or has at least as many zero periods, as the observed one. The simulated draws are kept in `attr(, "sims")` for plotting.

Scale and how to read it

`vmr`**, the variance-to-mean ratio.** 1 is Poisson, above 1 is overdispersed, below 1 is underdispersed. The negative binomial exists only above 1, so a series with `vmr <= 1` raises an error rather than being silently forced. The observed recruit series is used by default.

`phi`**, the reported size parameter.** $\phi = \mu/(v-1)$, the same quantity as in `phi_moment()` and read the same way: variance $\mu + \mu^2/\phi$, so small `phi` is strong clumping and $\phi \to \infty$ is Poisson. It is derived here from the pooled series rather than from unit-level expectations, so it is not directly comparable with the `phi` of `host_lag_test()`.

`p_gini`, `p_cv`, `p_zeros`. Proportions in [0, 1]: the fraction of `nsim` simulated series that are at least as concentrated as, or have at least as many zero periods as, the observed one. 0.5 is the middle of the simulated distribution. The Monte Carlo resolution floor is $1/\mathrm{nsim}$.

How to read a large value, which is the usual result. A probability near 0.2 to 0.4, as in the *Lepanthes eltoroensis* record, does not support the lag hypothesis. It says that this comparison cannot separate the two, because counts that are small and clumped look episodic whatever their mean. That is a statement about the power of the pooled test and is the reason `host_lag_test()` exists. Reporting only the pooled comparison would understate what the data can do.

The two nulls. `N0 flat mean` spreads the expected total evenly; `N1 best-kernel mean` uses the shape of the most concentrated kernel found by `lag_ceiling()`, rescaled to the observed total. N1 is the more favourable of the two to a delay, so it is the one to read first. Both obey the ceiling by construction.

No threshold. No cutoff is applied to these probabilities, and none is taken from the literature (Wasserstein and Lazar 2016).

References

Bliss, C. I. and Fisher, R. A. (1953) Fitting the negative binomial distribution to biological data. *Biometrics* 9: 176-200. doi:10.2307/3001850

Gelman, A., Meng, X.-L. and Stern, H. (1996) Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica* 6: 733-760.

Wasserstein, R. L. and Lazar, N. A. (2016) The ASA statement on p-values: context, process, and purpose. *The American Statistician* 70: 129-133. doi:10.1080/00031305.2016.1154108

Examples

```
ce <- lag_ceiling(lepanthes_census, reproduction = "inflorescences", K = 4,
                  kernels = lag_kernels(4, bin = 6, unit = "mo"))
ceiling_calibration(ce, nsim = 2000)
```

`ceiling_draws_table` *One table for several records and routes*

Description

Stacks the interval tables of any number of `lag_ceiling_draws` objects (from `lag_ceiling_boot()`, `lag_ceiling_bayesboot()` or `lag_ceiling_stan()`) into one tidy data frame with a `record` column, so that a document can print one table and a reader can compare records and routes down a column. The probability statement each route makes is carried in its own column with its name beside it, because the bootstrap's one-sided p-value and the Bayesian routes' posterior probability are different things and must not share a heading.

Usage

```
ceiling_draws_table(..., .list = NULL, quantity = "exceedance")
```

Arguments

`...`	`lag_ceiling_draws` objects, named. The names become the `record` column, so name them by the record or the route as the table needs: `` `L. eltoroensis` = b `` or `` `L. eltoroensis, bootstrap` = b ``.
`.list`	Alternatively, a named list of such objects. `NULL` entries, for a route that was not run, are dropped.
`quantity`	Which rows to keep: `"exceedance"` (the default), `"all"`, or any of the six quantity labels (`"observed Gini"`, `"ceiling Gini"`, `"exceedance Gini"`, `"observed CV"`, `"ceiling CV"`, `"exceedance CV"`).

Value

A data frame with columns `record`, `method`, `estimand`, `quantity`, `estimate`, `lower`, `upper`, `level`, `interval`, `probability`, `probability_is` and `note`. `probability` is filled on the exceedance rows only.

Scale and how to read it

`estimate`, `lower` and `upper` are on the scale of the quantity: a concentration index for the observed and ceiling rows, a ratio centred on 1 for the exceedance rows. `interval` says which interval it is (BCa, percentile, equal-tailed, highest-density) and `method` which route made it, and the two together fix how the interval may be read: a confidence interval from the cluster bootstrap, a credible interval from the other two. `probability_is` names the statement in `probability`. Print the table with `knitr::kable(digits = 2)` in a document.

Examples

```
set.seed(1)
k <- lag_kernels(4, bin = 6, unit = "mo")
b  <- lag_ceiling_boot(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                       K = 4, kernels = k, R = 200)
bb <- lag_ceiling_bayesboot(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                            K = 4, kernels = k, draws = 200)
ceiling_draws_table(`23 host trees` = b, `23 host trees` = bb)
```

`convolve_lag` *Apply a lag profile to a reproductive record*

Description

Computes the expected recruitment series implied by a set of lag weights: at each period at which recruits are scored, the weighted sum of the reproductive record over the preceding bins.

Usage

```
convolve_lag(X, w, t_R = NULL, lag0 = 1L, missing = c("backfill", "drop"))
```

Arguments

`X`	Numeric vector: the reproductive record at periods `1, 2, ...`.
`w`	Numeric vector of `K + 1` non-negative lag weights (they are renormalised to sum to one).
`t_R`	Integer vector: the periods (indices into `X`) at which recruits are scored. Defaults to the last `length(X) - lag0 - K` periods, that is, every period whose whole window lies inside the record; an error is raised if there is none.
`lag0`	Offset of the first bin. With `lag0 = 1` (the default, and the convention of a projection matrix) weight `k + 1` is applied to `X[t - 1 - k]`, so the first bin is the period preceding the recruit census. With `lag0 = 0` the first bin is the recruit census itself.
`missing`	How to treat lags that fall before the first period of `X`. `"backfill"` (the default) reads them as `X[1]`, which asserts that reproduction before the record began was at the level first observed. `"drop"` returns `NA` for any period whose window is not fully within the record. Zero-filling is deliberately not offered: it asserts that no seed fell before the record began, which turns a long pure delay into a prediction of no recruits at the first censuses and is an artefact.

Value

A numeric vector of the same length as `t_R`.

Scale and how to read it

The output is on the scale of `X` and in its units, because the weights are renormalised to sum to 1 and the result is a weighted average: every value lies between the smallest and the largest `X` in its own window. Nothing is multiplied by a rate or a recruitment probability, so the output is not a predicted number of recruits. It is the shape of the expected recruitment series, up to an unknown scale, which is all the ceiling argument needs and all that `lag_ceiling()` uses.

Two consequences are worth stating because they are where the intuition usually slips. Averaging cannot sharpen: the output can never be more concentrated in time than `X` itself, whatever the weights, and it equals `X`'s concentration only for a pure delay, which permutes the series rather than mixing it. And averaging a nearly constant `X` gives a nearly constant output for every set of weights, which is why a flat reproductive record cannot identify a lag profile. Slutzky (1937) is the classical statement of what a moving average does to a series.

References

Koyck, L. M. (1954) *Distributed Lags and Investment Analysis*. North-Holland, Amsterdam.

Slutzky, E. (1937) The summation of random causes as the source of cyclic processes. *Econometrica* 5: 105-146. doi:10.2307/1907241

Examples

```
X <- lepanthes_census$inflorescences
# a pure 24 to 30 month delay, five six-month bins, recruits scored at censuses 3 to 12
convolve_lag(X, c(0, 0, 0, 0, 1), t_R = 3:12)
```

`expected_recruits` *Expected recruitment per replicate unit under a lag profile*

Description

The lag hypothesis, stated at the level at which it is asserted: expected recruitment on each unit (host tree, plot, population) at each census is that unit's own lagged reproduction, weighted by the lag profile and scaled so that the unit's expected total equals its observed total. Nothing is fitted; the profile fixes the shape and the observed totals fix the scale.

Usage

```
expected_recruits(
  R,
  X = NULL,
  w,
  lag0 = 1L,
  missing = c("backfill", "renormalise", "zero"),
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits"
)
```

Arguments

`R`	Either a data frame with one row per unit and census holding the columns named in `unit`, `period`, `reproduction` and `recruits` (recruits `NA` at unscored censuses; see `host_matrices()`), or a numeric matrix of recruit counts, one row per unit and one column per census at which recruits are scored, with `colnames(R)` the census indices and `rownames(R)` the unit identifiers.
`X`	Numeric matrix of the reproductive record, one row per unit and one column per census from the first, with the same `rownames` as `R`. Ignored when `R` is a data frame.
`w`	Lag weights of length `K + 1`.
`lag0`	Offset of the first bin (see `convolve_lag()`).
`missing`	Treatment of lags before the first column of `X`: `"backfill"` reads them as the first census; `"renormalise"` drops them and renormalises the remaining weights; `"zero"` reads them as zero. The last two force a long delay to predict nothing at the first censuses and are offered only as sensitivity cases.
`unit`, `period`, `reproduction`, `recruits`	Column names used when `R` is a data frame.

Value

A matrix of expected counts with the dimensions of `R`. A unit whose lagged reproduction is zero throughout, but which recruited, is given a flat expectation.

Scale and how to read it

The returned values are expected counts on the scale of R, and each row sums to that unit's observed recruit total. That is the whole of the scaling: no rate, no offset and no fitted coefficient. Two consequences follow and are worth keeping in mind when reading the output.

First, the test built on this expectation is about **timing only**. How many recruits a unit received is taken from the data and is never predicted, so a unit that recruited heavily cannot make the lag hypothesis look good or bad by its total, only by when those recruits arrived.

Second, a unit that never recruited has an all-zero row and contributes nothing at all, so the effective replication is the number of units with at least one recruit and not nrow(R). In the *Lepanthes eltoroensis* host matrices that is 17 of the 23 trees, although host_lag_test() prints all 23. sum(rowSums(R) > 0) gives the number that actually carries the test, and it is the same set phi_moment() estimates clumping over.

A unit with recruits but no lagged reproduction on record is given a flat row rather than being dropped, because dropping it would quietly remove the units least favourable to the lag hypothesis.

Examples

```
mu <- expected_recruits(lepanthes_hosts, unit = "host",
                        reproduction = "inflorescences", w = flat_kernel(4))
round(mu[1:3, ], 2)
# each unit's expected total is its observed total, by construction
all.equal(rowSums(mu), rowSums(host_matrices(lepanthes_hosts, unit = "host",
                                             reproduction = "inflorescences")$R))
```

---

flat_kernel *The flat reference: every bin weighted equally*

Description

A convenience for the limiting case in which reproduction contributes equally at every lag, which for a near-constant reproductive record gives a near-constant expected recruitment series.

Usage

```
flat_kernel(K)
```

Arguments

K	Lag horizon, as in lag_kernels().

Value

A numeric vector of K + 1 equal weights.

Scale and how to read it

Each weight is $1/(K+1)$, so `flat_kernel(4)` is five weights of 0.2 and the horizon is covered evenly. It is one end of the range that `lag_kernels()` searches: the flat kernel gives the least concentrated expected series available, a pure delay the most concentrated, and every other profile lies between.

It is not a null of "no lag". The flat kernel is the lag hypothesis with memory spread evenly over the whole horizon, which is a strong biological claim in its own right (seed from every one of the last `K + 1` bins contributing equally). The projection-matrix null, recruitment set by the immediately preceding interval alone, is the opposite extreme, `c(1, 0, 0, 0, 0)`.

Do not confuse it with the flat reference of `host_lag_test()`. The `flat = TRUE` row of that function spreads each unit's recruit total evenly over the censuses, which is a statement about recruitment. The flat kernel spreads the weights evenly over lags, which is a statement about reproduction's memory. Where reproduction is close to constant the two give nearly the same expected series, and that near-coincidence is itself the second diagnosis of the companion paper.

Examples

```
flat_kernel(4)
sum(flat_kernel(4))
# the two extremes of the searched family, on the same record
X <- lepanthes_census$inflorescences
convolve_lag(X, flat_kernel(4), t_R = 3:12)      # least concentrated
convolve_lag(X, c(0, 0, 0, 0, 1), t_R = 3:12)    # a pure delay, most concentrated
```

---

host_lag_test — *The host-level test of the lag hypothesis*

Description

Tests whether the recruitment record, scored across replicate units, could have arisen under a delayed consequence of each unit's own reproduction. For each lag profile the expected counts are those of `expected_recruits()`, the clumping is `phi_moment()` (or a value you supply), and negative-binomial counts are simulated for every unit and summed over units to give a system-wide census series. The test reports, for each profile, the probability that a simulated series is at least as extreme as the observed one on each statistic. The verdict is the maximum of that probability over every profile, that is, the most favourable case the lag hypothesis can make for itself.

Usage

```
host_lag_test(
  R,
  X = NULL,
  K,
  lag0 = 1L,
  kernels = lag_kernels(K, bin = NULL),
  missing = c("backfill", "renormalise", "zero"),
```

```
  flat = TRUE,
  phi = "moment",
  nsim = 10000L,
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits"
)
```

Arguments

`R` Either a data frame with one row per unit and census holding the columns named in `unit`, `period`, `reproduction` and `recruits` (recruits `NA` at unscored censuses; see `host_matrices()`), or a numeric matrix of recruit counts, one row per unit and one column per census at which recruits are scored, with `colnames(R)` the census indices and `rownames(R)` the unit identifiers.

`X` Numeric matrix of the reproductive record, one row per unit and one column per census from the first, with the same `rownames` as `R`. Ignored when `R` is a data frame.

`K` Lag horizon (see `lag_kernels()`).

`lag0` Offset of the first bin (see `convolve_lag()`).

`kernels` A `lag_kernels` object; defaults to the named families (pure delays, windows at every start, geometric decay) with no random draws. Add `n_dirichlet` draws for a fuller search, at a cost in time.

`missing` Treatment of lags before the first column of `X`: `"backfill"` reads them as the first census; `"renormalise"` drops them and renormalises the remaining weights; `"zero"` reads them as zero. The last two force a long delay to predict nothing at the first censuses and are offered only as sensitivity cases.

`flat` Whether to add the flat-rate reference, in which each unit's total is spread evenly over the censuses.

`phi` `"moment"` for the moment estimate under each profile, or a numeric vector of fixed values at which to evaluate every profile as well (a stress test with more clumping than the counts support).

`nsim` Simulated series per profile.

`unit`, `period`, `reproduction`, `recruits` Column names used when `R` is a data frame.

Details

The statistics are: `silent`, the number of censuses at which no unit recruited; `gini` and `cv`, the concentration of the census totals; `max`, the largest census total. The Gini and CV are the pre-specified statistics of the companion paper; the silent-census count is the most interpretable and the largest census the least sensitive.

Value

An object of class `host_lag_test` with `obs` (the observed statistics of the census totals), `table` (one row per profile and phi setting: `label`, `family`, `phi_source`, `phi`, `gini_expected` (the Gini of the expected census totals), `p_silent`, `p_gini`, `p_cv`, `p_max`, `median_silent`), `verdict` (the maximum of each probability over profiles, with the profile attaining it), and the inputs.

Scale and how to read it

The probabilities. `p_silent`, `p_gini`, `p_cv` and `p_max` are proportions in [0, 1]: the fraction of `nsim` simulated records, drawn under the profile in that row, that are at least as extreme as the observed one on that statistic. They are predictive probabilities in the sense of Gelman, Meng and Stern (1996), computed under a fully specified null with nothing fitted. A value near 0.5 means the observed record is an ordinary draw under that profile. A value near 0 means it is not.

The resolution floor. A probability cannot be resolved below $1/\mathrm{nsim}$. With the default `nsim = 10000` a reported 0.0000 means "smaller than 1e-4", not zero, and the difference between 0.0000 and 0.0002 is one simulated record. Raise `nsim` before reading anything into a very small value, and note that `plot.host_lag_test()` floors the axis at $1/\mathrm{nsim}$ for the same reason.

The verdict is a maximum, not an average. `verdict` takes, for each statistic and each `phi` setting separately, the largest probability over every profile, together with the profile attaining it. It is therefore the best case the lag hypothesis can make for itself over the whole family searched. A small verdict says that no profile in the family, not merely the fitted one, reproduces the record.

No threshold. The package applies no cutoff to these probabilities and reports them as numbers. The 0.05 line drawn by `plot.host_lag_test()` is a conventional reference mark and not a decision rule; a bright-line threshold is exactly what the ASA statement on p-values advises against (Wasserstein and Lazar 2016). What the companion paper reports is the magnitude (a verdict of 0.0007 at the moment `phi`, 0.048 at a `phi` fixed three times lower) and leaves the reader the judgement.

Where the test is weak. The probabilities rise, correctly, when reproduction is seasonal, because a concentrated input can produce a concentrated output and the ceiling is then high. An ordinary verdict is therefore not evidence for a lag; it is the absence of evidence against one. See the `invented-regimes` vignette, which shows both outcomes on constructed records whose mechanism is known.

References

Gelman, A., Meng, X.-L. and Stern, H. (1996) Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica* 6: 733-760.

Wasserstein, R. L. and Lazar, N. A. (2016) The ASA statement on p-values: context, process, and purpose. *The American Statistician* 70: 129-133. doi:10.1080/00031305.2016.1154108

Examples

```
ht <- host_lag_test(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                    K = 4, nsim = 500, kernels = lag_kernels(4, bin = 6, unit = "mo"))
ht
```

host_matrices *Build the unit-by-period matrices from a long data frame*

Description

The host-level functions work on two matrices, recruits (units by scored periods) and reproduction (units by all periods). Most census data live in a long data frame with one row per unit and period; this function makes the matrices from it. Periods at which `recruits` is `NA` for every unit are taken as unscored.

Usage

```
host_matrices(
  data,
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits"
)
```

Arguments

`data` A data frame with one row per unit and period.

`unit`, `period`, `reproduction`, `recruits`
Names of the columns holding the unit identifier, the period index (integer, from 1), the reproductive count and the recruit count.

Value

A list with `R` (recruits, units by scored periods, `colnames` the period indices) and `X` (reproduction, units by periods 1 to the last period in `data`). Missing reproduction is treated as zero. A unit with `NA` recruits at a period that was scored for other units is treated as having zero recruits there, with a warning.

Examples

```
m <- host_matrices(lepanthes_hosts, unit = "host", reproduction = "inflorescences")
dim(m$R); dim(m$X)
```

`lag_ceiling` *The ceiling that any delay places on the concentration of recruitment*

Description

If recruitment is the delayed consequence of reproduction, expected recruitment is a convolution of the reproductive record, and a convolution is a weighted average. A weighted average is never more concentrated in time than the series it averages, so reproduction's own concentration is a ceiling on the concentration of expected recruitment, attained only by a pure delay. `lag_ceiling()` states that bound, searches a family of lag weights to confirm that none exceeds it on the periods at which recruits are scored, and compares the observed recruitment series with it.

Usage

```
lag_ceiling(
  X,
  R = NULL,
  t_R = NULL,
  K,
  lag0 = 1L,
  kernels = lag_kernels(K, n_dirichlet = 20000L),
  missing = c("backfill", "drop"),
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits"
)
```

Arguments

`X`	Either a numeric vector, the reproductive record at periods `1, 2, ...`, or a data frame with one row per period (or per unit and period, which is summed) holding the columns named in `period`, `reproduction` and `recruits`; recruits are `NA` at unscored periods.
`R`	Numeric vector: recruits scored at the periods `t_R`. Ignored when `X` is a data frame.
`t_R`	Periods of `X` at which `R` was scored; defaults to the last `length(R)` periods of `X`. Ignored when `X` is a data frame.
`K`	Lag horizon (see `lag_kernels()`).
`lag0`	Offset of the first bin (see `convolve_lag()`).
`kernels`	A `lag_kernels` object. Defaults to every family with 20,000 Dirichlet draws, as in the companion paper.
`missing`	Treatment of pre-record lags (see `convolve_lag()`).
`period, reproduction, recruits`	Column names used when `X` is a data frame.

Details

The searched ceiling can differ from the theorem bound by a small amount in either direction, because the kernels are scored on the recruitment periods only and pre-record lags are backfilled. The exceedance bounds the expected series, not the counts: whether a given exceedance refutes the delay depends on the sampling variation of the counts, which `ceiling_calibration()` evaluates for the pooled series and `host_lag_test()` evaluates across replicate units.

Value

An object of class `lag_ceiling` with elements `obs` (the concentration of `R`), `theorem` (the concentration of `X`, which is the bound), `ceiling` (the largest Gini and CV any searched kernel produced on the periods `t_R`), `exceedance` (observed over ceiling; `NA` when the reproductive record is flat, since the ceiling is then 0 and the ratio undefined, with `flat = TRUE`), `best` (the kernel attaining the ceiling), `search` (a data frame with one row per kernel) and the inputs. With `missing = "drop"`, scored periods whose lag window reaches before the record are left out of `R` and `t_R`, so that the observed and the expected series are compared on the same periods; `n_dropped` counts them.

Scale and how to read it

`theorem`**,** `ceiling` **and** `obs` are concentration indices, so they carry the scale of `rain_gini()` and `rain_cv()`: 0 for an even series, rising with concentration, with a maximum of $(n-1)/n$ and $\sqrt{n}$ respectively that depends on the number of scored periods. `theorem` is the bound derived from the reproductive series itself; `ceiling` is the largest value actually attained by any kernel searched, on exactly the periods at which recruits were scored.

`exceedance` **is a ratio, observed over searched ceiling.** It is therefore centred on 1, not on 0:

- below 1: the observed recruitment record is no more concentrated than some delay predicts. Nothing is refuted and nothing needs to be.
- at 1: the record sits exactly at the bound, which is what a pure delay of a perfectly observed series would give.
- above 1: the record is more concentrated than the expectation under any delay searched. In the *Lepanthes eltoroensis* data the ratios are 2.56 and 2.51 for the six-monthly record and 9.0 and 15.9 for the monthly one.

An exceedance above 1 is not by itself a refutation, and no threshold is supplied for it. The bound applies to the expected series. A count series is more concentrated than its own mean, so some exceedance is expected even when the lag hypothesis is true, and how much depends on the counts rather than on the biology. Read the exceedance as a description of the gap to be explained, then take it to `ceiling_calibration()`, which shows what the pooled comparison can refute (in this record, nothing), and to `host_lag_test()`, which is where the refutation is made.

A high ceiling means a weak test. The bound is reproduction's own concentration, so a seasonal reproductive record gives a high ceiling and little to detect. The test bites where reproduction is close to continuous. The closest empirical precedent for comparing the variability of the two series is Wright et al. (2005), who measured both for a neotropical forest but derived no bound from them.

References

Hardy, G. H., Littlewood, J. E. and Polya, G. (1952) *Inequalities*, 2nd edn. Cambridge University Press, Cambridge.

Marshall, A. W., Olkin, I. and Arnold, B. C. (2011) *Inequalities: Theory of Majorization and Its Applications*. Springer, New York. doi:10.1007/9780387682761

Wright, S. J., Muller-Landau, H. C., Calderon, O. and Hernandez, A. (2005) Annual and spatial variation in seedfall and seedling recruitment in a neotropical forest. *Ecology* 86: 848-860. doi:10.1890/030750

Examples

```
ce <- lag_ceiling(lepanthes_census, reproduction = "inflorescences", K = 4,
                  kernels = lag_kernels(4, n_dirichlet = 200, bin = 6, unit = "mo"))
ce
# the same from vectors
d <- lepanthes_census
lag_ceiling(d$inflorescences, d$recruits[3:12], t_R = 3:12, K = 4,
            kernels = lag_kernels(4, bin = 6, unit = "mo"))
```

---

lag_ceiling_bayesboot *A posterior for the exceedance, by the Bayesian bootstrap over units*

Description

The same quantity as `lag_ceiling_boot()`, the exceedance of the realised counts over the ceiling, but constructed as a posterior. Rubin's (1981) Bayesian bootstrap places a flat Dirichlet prior on the unknown distribution of units and draws, at each iteration, a set of weights `Dirichlet(1, ..., 1)` over the units; both series are rebuilt as the weighted sums and the ceiling and exceedance are recomputed. The result is a posterior distribution for each statistic, from which a credible interval and `P(exceedance > 1)` are read directly.

Usage

```
lag_ceiling_bayesboot(
  data,
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits",
  K,
  lag0 = 1L,
  kernels = lag_kernels(K),
  missing = c("backfill", "drop"),
  draws = 4000L,
  level = 0.9,
  interval = c("eti", "hdi")
)
```

Arguments

`data`	A data frame with one row per unit and period.
`unit`, `period`, `reproduction`, `recruits`	Column names.
`K`	Lag horizon (see `lag_kernels()`).
`lag0`	Offset of the first bin (see `convolve_lag()`).
`kernels`	A `lag_kernels` object, used in every replicate. The default is the named families with no random draws: the pure delays attain the bound, so random kernels can only add time.
`missing`	Treatment of pre-record lags (see `convolve_lag()`).
`draws`	Number of posterior draws.
`level`	Coverage of the interval. The default 0.9 matches the 90% credible intervals of the companion paper.
`interval`	`"eti"` (the default) for the equal-tailed interval, the central `level` of the posterior; `"hdi"` for the highest-density interval, the shortest interval holding `level` of the posterior.

Value

An object of class `lag_ceiling_draws` with `method = "bayesian bootstrap"`: `point` (the statistics on the full data), `draws` (one row per posterior draw), `table` (estimate and credible interval for each statistic), `prob` (`P(exceedance > 1)` on each index), `n_units` and the inputs.

Scale and how to read it

The interval is a credible interval and `prob` is a posterior probability: under the stated prior, that is the probability that the exceedance exceeds 1. This is the reading the ordinary bootstrap's `p_boot` is often given and does not have. The prior is the flat Dirichlet on the distribution of units, which is noninformative in Rubin's sense and is the same prior for every data set; nothing else is assumed, and no likelihood for the counts is written down.

Numerically the two bootstraps usually agree closely, because the Bayesian bootstrap is a smoothed version of the ordinary one. Where they differ is in what the sentence built on them may say.

Equal-tailed or highest density. The equal-tailed interval is invariant to a change of scale: the 5th percentile of the log exceedance is the log of the 5th percentile of the exceedance. The highest-density interval is not: the shortest interval on the ratio and the shortest interval on its log, back-transformed, are different intervals, and for a skewed ratio the difference is visible. `interval = "hdi"` is offered so that a user who prefers it can have it, and the help page says this so that the choice is made knowingly.

What it estimates. The exceedance of the *realised* counts, as in `lag_ceiling()` and `lag_ceiling_boot()`. A count series is more concentrated than its own expectation, so the exceedance is inflated by sampling variation in a way that no interval on it removes; `lag_ceiling_stan()` estimates the concentration of the expected series instead, which is the object the bound actually applies to.

References

Rubin, D. B. (1981) The Bayesian bootstrap. *The Annals of Statistics* 9: 130-134. doi:10.1214/aos/1176345338

Examples

```
set.seed(1)
bb <- lag_ceiling_bayesboot(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                            K = 4, draws = 200, kernels = lag_kernels(4, bin = 6, unit = "mo"))
bb
```

`lag_ceiling_boot` *A confidence interval for the exceedance, by resampling units*

Description

The ceiling is arithmetic and carries no uncertainty, but the two concentration indices it is compared with are estimated from a short record, and so is their ratio, the exceedance. `lag_ceiling_boot()` puts a confidence interval on them by resampling the replicate units (host trees, plots, populations) with replacement, rebuilding **both** the reproductive series and the recruit series from the resampled units, and recomputing the ceiling and the exceedance in every replicate. Numerator and denominator therefore move together, as they should, since the same units generate both series.

Usage

```
lag_ceiling_boot(
  data,
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits",
  K,
  lag0 = 1L,
  kernels = lag_kernels(K),
  missing = c("backfill", "drop"),
  R = 2000L,
  level = 0.9,
  type = c("bca", "percentile")
)
```

Arguments

`data`	A data frame with one row per unit and period.
`unit`, `period`, `reproduction`, `recruits`	Column names.
`K`	Lag horizon (see `lag_kernels()`).
`lag0`	Offset of the first bin (see `convolve_lag()`).
`kernels`	A `lag_kernels` object, used in every replicate. The default is the named families with no random draws: the pure delays attain the bound, so random kernels can only add time.
`missing`	Treatment of pre-record lags (see `convolve_lag()`).

R	Number of bootstrap replicates.
level	Coverage of the interval. The default 0.9 matches the 90% credible intervals of the companion paper.
type	"bca" (the default) for the bias-corrected and accelerated interval of Efron (1987), with the acceleration from the leave-one-unit- out estimates; "percentile" for the plain percentile interval.

Value

An object of class lag_ceiling_draws with method = "bootstrap": point (the six statistics on the full data, identical to lag_ceiling() on the same kernels), draws (one row per replicate), table (estimate and interval for each statistic), p_boot (the one-sided bootstrap p-value for exceedance at most 1, on each index), n_units, R, n_failed (replicates in which a statistic could not be computed, usually because no resampled unit recruited) and the inputs.

Scale and how to read it

The interval is a confidence interval and is read in the usual way: the procedure covers the true value in level of repeated samples of units from the same population of units. It is not a probability that the exceedance lies in the interval. The Bayesian reading is available from lag_ceiling_bayesboot(), which is built to give it.

p_boot **is a one-sided bootstrap p-value**, the fraction of replicates in which the exceedance was at most 1: the achieved significance level against the hypothesis that the record is no more concentrated than a delay predicts. It is not the probability that the exceedance is at most 1, although it is easily mistaken for one, and it cannot be resolved below 1/R. A value printed as < 1/R means no replicate reached 1.

Why units and not periods. The Gini and the CV are permutation invariant, so resampling periods would not disturb them by reordering, but periods within a unit are not independent and the units are the design's replicates. Resampling units is the cluster bootstrap of Davison and Hinkley (1997, ch. 3). With few units (the *Lepanthes eltoroensis* record has 23) the BCa interval is preferred to the percentile one, since it corrects for the skew a ratio carries and for the bias between the bootstrap distribution and the estimate. Below ten units the function warns that the bootstrap distribution has few distinct values; ten is a working convention of this package for when to say so, not a published minimum, and the interval is still computed.

What the interval does not say. A wide interval does not weaken the ceiling, which is a bound and not an estimate. It says how firmly the record itself sits above the bound. Whether an exceedance above 1 refutes the delay is still the question for host_lag_test(), because the bound applies to the expected series and not to the counts.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and their Application*. Cambridge University Press, Cambridge.

DiCiccio, T. J. and Efron, B. (1996) Bootstrap confidence intervals. *Statistical Science* 11: 189-228. doi:10.1214/ss/1032280214

Efron, B. (1987) Better bootstrap confidence intervals. *Journal of the American Statistical Association* 82: 171-185. doi:10.1080/01621459.1987.10478410

Examples

```
set.seed(1)
b <- lag_ceiling_boot(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                      K = 4, R = 200, kernels = lag_kernels(4, bin = 6, unit = "mo"))
b
as.data.frame(b)
```

---

lag_ceiling_by *The ceiling for every unit in a long data frame*

Description

Runs `lag_ceiling()` once per replicate unit (population, plot, host tree) and returns the results together, so that a figure of one strip per unit, or a table of one row per unit, is a single call. This is the comparison the ceiling is usually wanted for: not whether one record exceeds its bound, but how far above the bound each unit sits and how much that varies within a study.

Usage

```
lag_ceiling_by(
  data,
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits",
  K,
  lag0 = 1L,
  kernels = lag_kernels(K),
  missing = c("backfill", "drop"),
  min_periods = K + 3L,
  min_recruits = 2
)
```

Arguments

`data`	A data frame with one row per unit and period.
`unit`, `period`, `reproduction`, `recruits`	Column names.
`K`	Lag horizon (see `lag_kernels()`).
`lag0`	Offset of the first bin (see `convolve_lag()`).
`kernels`	A `lag_kernels` object, used for every unit so that the units are compared on the same family.
`missing`	Treatment of pre-record lags (see `convolve_lag()`).
`min_periods`	Units with fewer scored periods (periods at which recruits were counted) than this are skipped. The default `K + 3` is the shortest record on which a horizon of `K + 1` bins leaves anything to compare.

`min_recruits` Units with fewer recruits than this are skipped, since a unit with one recruit has the maximum concentration, $(n-1)/n$, by arithmetic rather than by biology.

Details

Rows in which both reproduction and recruits are missing are taken as periods in which that unit was not watched, and are dropped before its periods are re-indexed to start at 1. A unit followed from 2015 to 2021 is therefore a seven-period record, not a 22-period record with fifteen silent years in front of it. Keeping them would read those years as periods with no reproduction, which raises the unit's ceiling and makes the test look weaker than it is. Units are otherwise independent: nothing is pooled, and the kernels are the same for all of them.

Value

A named list of `lag_ceiling` objects, of class `lag_ceiling_list`, with the skipped units and the reason in `attr(, "skipped")`. It can be passed straight to `rain_strip()`, and `as.data.frame()` gives one row per unit.

Scale and how to read it

The columns of `as.data.frame()` carry the scales of `lag_ceiling()`: `ceiling_gini` and `observed_gini` are concentration indices bounded above by $(n-1)/n$, where `n` is that unit's own number of scored periods, and `exceedance_gini` and `exceedance_cv` are the ratios, centred on 1. A unit whose reproductive record is flat has no ceiling to exceed, and its exceedance is `NA` with the reason in `best_kernel`.

Because `n` differs between units, two cautions apply to reading the table or the figure down a column. A unit with few periods has a lower attainable maximum, so its Gini is not on the same footing as a longer unit's, and `n` is reported for that reason. And the exceedance is a ratio of two numbers computed on the same `n`, so it is the safer quantity to compare across units of different lengths.

What varies between units is usually the ceiling rather than the observation: the ceiling is a property of that unit's reproductive record, so a unit with seasonal or patchy reproduction sits higher and has less to detect. A column of exceedances close to 1 is not a weak result, it is a statement that those units' reproduction is itself episodic.

Examples

```
ces <- lag_ceiling_by(lepanthes_hosts, unit = "host",
                      reproduction = "inflorescences", K = 4,
                      kernels = lag_kernels(4, bin = 6, unit = "mo"))
ces
as.data.frame(ces)
```

`lag_ceiling_stan` *A posterior for the exceedance of the expected series, from a model of the counts*

Description

The bound applies to the **expected** recruitment series: under any delay, `E[R_t]` is a weighted average of the reproductive record and cannot be more concentrated than it. `lag_ceiling()` compares the bound with the realised counts, which are more concentrated than their own expectation whenever counts are small, and `ceiling_calibration()` shows how much that inflates the exceedance. `lag_ceiling_stan()` estimates the object the bound is about. It fits a negative-binomial model to the recruit counts on every unit at every scored period, with a unit effect and a period effect on the log scale, and hands the posterior of the expected system totals to the same ceiling arithmetic as everywhere else in the package. The reproductive record enters as observed: it is the driver the hypothesis names, and the ceiling is computed from it exactly as in `lag_ceiling()`. The result is a posterior for the concentration of the expected recruitment series and for its ratio to the ceiling.

Usage

```
lag_ceiling_stan(
  data,
  unit = "unit",
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits",
  K,
  lag0 = 1L,
  kernels = lag_kernels(K),
  missing = c("backfill", "drop"),
  chains = 4,
  parallel_chains = chains,
  threads_per_chain = 1,
  iter_warmup = 1500,
  iter_sampling = 2000,
  adapt_delta = 0.95,
  max_treedepth = 12,
  seed = NULL,
  sigma_scale = 1,
  phi_prior = c(2, 0.1),
  level = 0.9,
  interval = c("eti", "hdi"),
  refresh = 0,
  ...
)
```

Arguments

`data`	A data frame with one row per unit and period.

`unit`, `period`, `reproduction`, `recruits`
: Column names.

`K`
: Lag horizon (see `lag_kernels()`).

`lag0`
: Offset of the first bin (see `convolve_lag()`).

`kernels`
: A `lag_kernels` object, used in every replicate. The default is the named families with no random draws: the pure delays attain the bound, so random kernels can only add time.

`missing`
: Treatment of pre-record lags (see `convolve_lag()`).

`chains`, `iter_warmup`, `iter_sampling`, `seed`, `adapt_delta`, `max_treedepth`
: Passed to CmdStan's sampler. The defaults are those of the companion paper's hierarchical recruitment model: a hierarchical model with a period effect that has to reach very low values at silent periods needs the higher target acceptance and the deeper trees.

`parallel_chains`
: Chains run at once; defaults to `chains`.

`threads_per_chain`
: Threads within each chain. Above 1 the likelihood is split across threads with Stan's `reduce_sum`, and the model is compiled with threading on (once, cached separately). Worth it for records with many units, such as a thousand patches; on a small record it only adds overhead. `chains * threads_per_chain` should not exceed the cores available.

`sigma_scale`
: Scale of the half-Student-t(3) priors on the two effect standard deviations.

`phi_prior`
: Shape and rate of the gamma prior on the negative-binomial clumping parameter. The default `c(2, 0.1)` is the working prior of the companion paper, a placeholder pending elicitation; it keeps `phi` away from both 0 and infinity, which is what lets the sampler separate clumping from the period effect.

`level`
: Coverage of the interval. The default 0.9 matches the 90% credible intervals of the companion paper.

`interval`
: `"eti"` or `"hdi"`, as in `lag_ceiling_bayesboot()`.

`refresh`
: How often CmdStan reports progress; 0 is silent.

`...`
: Further arguments to the `$sample()` method of a `CmdStanModel`.

Value

An object of class `lag_ceiling_draws` with `method = "stan"`: `point` (the posterior median of each statistic), `draws` (one row per posterior draw), `table`, `prob` (`P(exceedance > 1)`), `diagnostics` (divergent transitions, transitions that hit the maximum tree depth, largest Rhat and smallest bulk ESS over the model's parameters), `fit` (the `CmdStanMCMC` object, for anything the summary does not cover) and the inputs.

Requirements

The model is compiled and run through `cmdstanr`, which is not on CRAN, and CmdStan itself. Install with `install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev", getOption("repos")))` and then `cmdstanr::install_cmdstan()`. The compiled model is cached in `tools::R_user_dir("recruitlag", "cache")` after the first call. Without them the function stops with that message and nothing else in the package is affected.

Scale and how to read it

A different estimand from the two bootstraps. They put an interval on the exceedance of the realised counts; this puts one on the exceedance of the expected series. The second is usually lower, because the negative- binomial model attributes part of the concentration of the counts to sampling variation about a smoother mean, and the difference between the two is itself informative: it is the share of the raw exceedance that the counts alone could have produced.

What the model assumes. The recruit count on a unit at a period is negative binomial with a unit effect and a period effect, additive on the log scale, partially pooled, and one clumping parameter. No lag is fitted; the ceiling is imposed afterwards, from the observed reproductive record, over the same kernels as `lag_ceiling()`. The period effects carry the concentration, so with few units per period they are shrunk toward each other and the estimated concentration is conservative, that is, it errs toward a lower exceedance.

Why the reproductive record is not modelled too. Modelling it and taking the ceiling from its expected series looks symmetrical, but the ceiling is then a modelled quantity that can shrink to zero when the reproductive period effects do, and the exceedance, a ratio, explodes. The hypothesis names the observed record as the driver, so that is what the ceiling is computed from.

The recruits must be counts. A rate has no negative-binomial likelihood, and the function refuses it with a message.

Cells that were not censused are left out, not read as zero. A unit with no row, or an `NA` recruit count, at a scored period contributes nothing to the likelihood for that period, whereas `lag_ceiling()` and the bootstraps read it as zero. The expected series is then the total the full set of units would have produced at every period, so on a record where the units visited varied between periods (the *Lepanthes* metapopulation, for instance) it is an effort-corrected series and can sit above the bootstraps, which count what was seen.

Read the sampler diagnostics before the interval. Divergent transitions, an Rhat above about 1.01 or a bulk ESS below about 100 mean the posterior was not explored and the interval is not to be trusted. The function warns at those values, and the printed summary carries the numbers. The two cutoffs are working conventions of this package for deciding when to warn, not published thresholds; the interval itself is never altered by them. Transitions that hit the maximum tree depth are an efficiency warning, not a validity one, but many of them with a poor Rhat mean the chains sat in different regions.

Equal-tailed or highest density. As in `lag_ceiling_bayesboot()`: the equal-tailed interval is invariant to a change of scale and the highest-density interval is not, which matters for a ratio.

References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. and Rubin, D. B. (2013) *Bayesian Data Analysis*, 3rd edn. Chapman and Hall, Boca Raton.

Stan Development Team (2024) *Stan Modeling Language Users Guide and Reference Manual*. https://mc-stan.org

Examples

```
## Not run:
st <- lag_ceiling_stan(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                       K = 4, kernels = lag_kernels(4, bin = 6, unit = "mo"))
```

```
st

## End(Not run)
```

---

lag_kernels *Families of lag weights*

Description

Builds the set of lag profiles (kernels) over which the ceiling is searched and the host-level null is evaluated. Each kernel is a vector of K + 1 non-negative weights summing to one; weight k + 1 applies to the reproductive record k bins before the first lagged bin (see convolve_lag() for the census convention).

Usage

```
lag_kernels(
  K,
  families = c("delay", "window", "geometric", "dirichlet"),
  windows = c("all", "anchored"),
  rho = c(0.2, 0.4, 0.6, 0.8, 0.95),
  n_dirichlet = 0L,
  alpha = 0.3,
  bin = NULL,
  unit = "",
  extra = NULL
)
```

Arguments

K	Lag horizon: the number of bins beyond the first, so that a kernel has K + 1 weights. The paper uses K = 6 for monthly data (lags 0 to 6 months) and K = 4 for six-monthly data (five six-month bins, labelled 0-6 to 24-30 months, counted from the first lagged census; see lag0 in convolve_lag()).
families	Which families to include. "delay": pure delays, all weight on one bin (these attain the ceiling). "window": uniform windows of every width from 2 to K + 1, at every start (windows = "all") or anchored at the first bin only (windows = "anchored"). "geometric": geometric decay at the rates in rho. "dirichlet": n_dirichlet random draws from a symmetric Dirichlet distribution with shape alpha.
windows	"all" or "anchored"; see families.
rho	Decay rates for the geometric family.
n_dirichlet	Number of random Dirichlet kernels.
alpha	Shape of the Dirichlet distribution (0.3 gives sparse kernels; 1 is uniform on the simplex).

`bin`	Width of one lag bin in the time unit of `unit`, used only to label kernels (for example `bin = 6, unit = "mo"` labels the six-monthly bins "0-6 mo", "6-12 mo", ...). `NULL` labels bins by index.
`unit`	Time unit for labels.
`extra`	A named list of additional weight vectors to append, for example a posterior-mean lag profile from a fitted model.

Value

An object of class `lag_kernels`: a list with one element per kernel, each a list with `label`, `family` and `w`. `as.matrix()` returns the weights as a matrix with one row per kernel.

Scale and how to read it

The weights. Every kernel is a probability vector: `K + 1` non-negative weights summing to 1. The weights say how memory is distributed over the horizon, never how much recruitment there is; the scale is carried elsewhere and never enters the ceiling.

`K`**, the horizon, in bins and in time.** A kernel covers `K + 1` bins, so with `bin` months per bin the horizon reaches `bin * (K + 1)` months back from the first lagged bin. The labels count from that first lagged bin, not from the recruit census: with `lag0 = 1` the first bin is the census before the recruit census, and the label "0-6 mo" means the first six months of the lag window. The paper's `K = 4` with `bin = 6` searches a 30-month window, labelled 0-6 to 24-30 months. Nothing outside the horizon is tested, so a delay longer than `bin * (K + 1)` is not refuted by a small verdict: state the horizon whenever the result is reported.

`alpha`**, the Dirichlet shape.** This is the only parameter here with a scale that is easy to misread. It is the shape of a symmetric Dirichlet on the simplex of weights:

- `alpha < 1` (the default 0.3) pushes draws toward the vertices, so most random kernels are sparse and close to pure delays. This is the useful setting, because concentrated kernels are the ones that approach the ceiling.
- `alpha = 1` is uniform on the simplex.
- `alpha > 1` pushes draws toward the centre, giving flat profiles, which searches the least concentrated corner of the family and will understate the ceiling.

The construction of a Dirichlet simplex over lag weights follows Ogle et al. (2015), where it is a prior; here the same object is a search grid and nothing is fitted.

`rho`**, the geometric decay rates.** Weight `k` is proportional to ρ^k, so `rho` near 0 concentrates on the first bin and `rho` near 1 approaches equal weights. This is the Koyck (1954) form, the shape assumed whenever memory is taken to fade at a constant rate per period.

How many kernels you get. The delay family contributes `K + 1`, the window family every contiguous width and start (`windows = "all"`) or only those anchored at the first bin, the geometric family one per `rho`, and the Dirichlet family `n_dirichlet`. Print the object to see the total. More kernels can only raise the searched ceiling and so can only make the test more conservative.

References

Gasparrini, A. (2014) Modeling exposure-lag-response associations with distributed lag non-linear models. *Statistics in Medicine* 33: 881-899. doi:10.1002/sim.5963

Koyck, L. M. (1954) *Distributed Lags and Investment Analysis*. North-Holland, Amsterdam.

Ogle, K., Barber, J. J., Barron-Gafford, G. A., Bentley, L. P., Young, J. M., Huxman, T. E., Loik, M. E. and Tissue, D. T. (2015) Quantifying ecological memory in plant and ecosystem processes. *Ecology Letters* 18: 221-235. doi:10.1111/ele.12399

van de Pol, M. and Cockburn, A. (2011) Identifying the critical climatic time window that affects trait expression. *The American Naturalist* 177: 698-707. doi:10.1086/659101

Examples

```
k <- lag_kernels(4, bin = 6, unit = "mo")
k
as.matrix(k)[1:3, ]
```

`lepanthes` *Census records of the epiphytic orchid* Lepanthes eltoroensis

---

Description

Four tibbles from the two surveys analysed in the companion paper, in which reproduction is continuous in time and recruitment is episodic. Luquillo Mountains, Puerto Rico.

Format

Four tibbles. `lepanthes_monthly`: 23 rows, columns `period`, `reproductive_adults`, `recruits`. `lepanthes_census`: 12 rows, columns `period`, `census`, `years_since_georges`, `adults`, `juveniles`, `seedlings`, `inflorescences`, `flowers`, `fruits`, `recruits`. `lepanthes_hosts`: 276 rows, columns `host`, `period`, `adults`, `inflorescences`, `recruits`. `lepanthes_profile`: 5 rows, columns `bin`, `months`, `weight`. Each is described below.

Details

`lepanthes_monthly` (23 rows): the monthly survey of four populations on the El Toro trail, September 1994 to November 1996. `period` is the census index, `reproductive_adults` the number of adults bearing an inflorescence (summed over populations) and `recruits` the new seedlings first seen at that census (`NA` at census 1, where first appearance is undefined).

`lepanthes_census` (12 rows): system totals of the six-monthly survey of 23 host trees begun after Hurricane Georges, March 1999 to September 2004: `period`, `census` (label), `years_since_georges`, stage counts (`adults`, `juveniles`, `seedlings`), reproductive totals (`inflorescences`, `flowers`, `fruits`) and `recruits`, scored at censuses 3 to 12 (`NA` at censuses 1 and 2, which are the initial enumeration and relocation additions).

`lepanthes_hosts` (276 rows): the same survey with one row per host tree and census: `host` (identifier), `period`, `adults`, `inflorescences` and `recruits` (`NA` at censuses 1 and 2). This is the input of `host_lag_test()`, through `unit = "host"` and `reproduction = "inflorescences"`.

lepanthes_profile (5 rows): the posterior-mean lag profile of the distributed-lag model fitted in the companion paper, one row per six-month bin (bin, months, weight), weights summing to one. An example of a fitted profile to pass to lag_kernels() through extra.

Source

Tremblay, R. L. Sex every month, seedlings once in a while: can episodic recruitment be a delayed consequence of continuous reproduction? (companion paper); Tremblay & Ackerman (2001) Biological Journal of the Linnean Society 72: 47-62 for the monthly survey. The census data are released under the Creative Commons Attribution 4.0 licence (CC BY 4.0), the same terms as the paper's data archive; the package code is under the MIT licence.

Examples

```
lepanthes_census
head(lepanthes_hosts)
```

---

phi_moment — *Moment estimate of negative-binomial clumping under a heterogeneous mean*

Description

With expected counts that vary by unit and census, the clumping parameter is identified by the identity E[(R - mu)^2] = E[mu] + E[mu^2] / phi, so phi = mean(mu^2) / (mean((R - mu)^2) - mean(mu)), over the units that recruited at least once. Estimating phi about a single pooled mean instead attributes the between-unit spread of the means to clumping and roughly halves phi.

Usage

```
phi_moment(R, mu, cap = 1e+06)
```

Arguments

R	Numeric matrix of recruit counts, units by scored censuses.
mu	Matrix of expected counts, as returned by expected_recruits().
cap	Upper limit returned when the counts show no overdispersion, that is, when the observed mean squared deviation does not exceed the mean and the moment estimator would be negative or undefined. The default 1e6 is an effectively Poisson value and is a sentinel, not an estimate: treat a returned cap as "no overdispersion detected".

Value

A single number on the scale described below.

Scale and how to read it

phi is the size (or clumping) parameter of the negative binomial, k in the ecological literature, entering the variance as $\mathrm{Var}(R) = \mu + \mu^2/\phi$. Read $1/\phi$ rather than ϕ: it is the excess variance per unit of squared mean.

- $\phi \to \infty$ ($1/\phi = 0$): no clumping, the counts are Poisson about their mean. This is the value cap stands in for.
- $\phi = 1$: variance $\mu + \mu^2$, strong clumping.
- $\phi < 1$: heavier still. The *Lepanthes eltoroensis* host matrices give $\phi = 0.19$ under the heterogeneous mean, so at a mean of 1 recruit the variance is about 6.
- $\phi \leq 0$ is not returned: the estimator is capped instead.

Smaller phi makes the null more permissive, because clumped counts are themselves episodic. A test run at a small phi is therefore the conservative one, which is why `host_lag_test()` accepts a vector of fixed values as a stress test.

No threshold, and no comparison across data sets. The package supplies no value of phi above which a record counts as aggregated. phi is also not a fixed constant of a species or a system: it is density dependent (Taylor, Woiwod and Perry 1979), so two estimates are comparable only at comparable means. Report it with the mean it was estimated at, as `host_lag_test()` does.

Which mean it is estimated about matters. Estimating phi about a single pooled mean, rather than about the unit-by-census means of `expected_recruits()`, attributes the between-unit spread of the means to clumping and roughly halves the estimate. In the companion paper the single-mean value was 0.089 against the moment value 0.19 under the heterogeneous mean.

References

Anscombe, F. J. (1949) The statistical analysis of insect counts based on the negative binomial distribution. *Biometrics* 5: 165-173. doi:10.2307/3001918

Bliss, C. I. and Fisher, R. A. (1953) Fitting the negative binomial distribution to biological data. *Biometrics* 9: 176-200. doi:10.2307/3001850

Taylor, L. R., Woiwod, I. P. and Perry, J. N. (1979) The negative binomial as a dynamic ecological model for aggregation, and the density dependence of k. *Journal of Animal Ecology* 48: 289-304. doi:10.2307/4114

Examples

```
m <- host_matrices(lepanthes_hosts, unit = "host", reproduction = "inflorescences")
mu <- expected_recruits(m$R, m$X, w = flat_kernel(4))
phi_moment(m$R, mu)
```

`plot.ceiling_calibration`

Plot a calibration or a host-level test

Description

For a `ceiling_calibration`, the distribution of the simulated Gini under each null with the observed value and the ceiling marked. For a `host_lag_test`, the probability of a record at least as extreme as the observed one, by kernel family, on a log scale.

Usage

```
## S3 method for class 'ceiling_calibration'
plot(x, ...)

## S3 method for class 'host_lag_test'
plot(x, statistic = c("silent", "gini", "cv", "max"), ...)
```

Arguments

`x`	The object.
`...`	Ignored.
`statistic`	For `host_lag_test`: which probability to plot.

Value

A ggplot.

Scale and how to read it

For a `host_lag_test` the vertical axis is a probability on a log scale, floored at $1/\mathrm{nsim}$ because no simulation can resolve below that. Points sitting on the floor mean "smaller than $1/\mathrm{nsim}$", not zero. Two horizontal lines are drawn: the dashed line is the flat reference, which is the comparison that matters, and the dotted line at 0.05 is a reference mark for the eye, the level a reader will look for, and not a decision rule (Wasserstein and Lazar 2016); nothing in the package tests against it. Read the plot as the spread of the whole kernel family, since the verdict is the highest point on it.

For a `ceiling_calibration` the axis is the Gini of simulated count series, with the observed value and the ceiling on the mean marked. The distance between those two marks is the question the plot answers: how much of the gap between the ceiling and the observation is explained by counting alone.

References

Wasserstein, R. L. and Lazar, N. A. (2016) The ASA statement on p-values: context, process, and purpose. *The American Statistician* 70: 129-133. doi:10.1080/00031305.2016.1154108

Examples

```
ce <- lag_ceiling(lepanthes_census, reproduction = "inflorescences", K = 4,
                  kernels = lag_kernels(4, bin = 6, unit = "mo"))
plot(ceiling_calibration(ce, nsim = 2000))

ht <- host_lag_test(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                    K = 4, nsim = 500, kernels = lag_kernels(4, bin = 6, unit = "mo"))
plot(ht, statistic = "silent")
```

---

rain_forest — *The exceedance and its interval, for several records and routes on one axis*

Description

A forest plot: one row per record, one point and interval per route, and a vertical line at 1, which is the bound. Because the exceedance is a ratio the axis is logarithmic, so that a record twice above the bound and one twice below it sit at equal distances from the line.

Usage

```
rain_forest(
  ...,
  .list = NULL,
  index = c("gini", "cv", "both"),
  order = c("estimate", "name")
)
```

Arguments

`...`	`lag_ceiling_draws` objects, named. The names become the `record` column, so name them by the record or the route as the table needs: `` `L. eltoroensis` = b `` or `` `L. eltoroensis, bootstrap` = b ``.
`.list`	Alternatively, a named list of such objects. `NULL` entries, for a route that was not run, are dropped.
`index`	`"gini"` (the default), `"cv"` or `"both"` (two panels).
`order`	`"estimate"` (the default, largest exceedance at the top) or `"name"` (the order the records were given).

Value

A ggplot.

Scale and how to read it

The horizontal axis is the exceedance, observed concentration over the ceiling, on a log scale with 1 marked. An interval wholly to the right of 1 is a record more concentrated than any delay of its reproductive series predicts, under that route's reading of "interval"; one that crosses 1 is a record on which the check cannot decide. Routes are distinguished by shape and colour and are read as their help pages say: a confidence interval for the cluster bootstrap, a credible interval for the Bayesian bootstrap and the negative-binomial model, and the last of these estimates the expected series rather than the counts, so it usually sits to the left of the other two. A row whose routes disagree is worth a sentence; a row whose routes agree and cross 1 is a record on which the reproductive series is itself episodic, or the recruits too few, and the figure cannot tell which. The rows are ordered by the first route's estimate unless `order = "name"`.

Examples

```
set.seed(1)
k <- lag_kernels(4, bin = 6, unit = "mo")
b  <- lag_ceiling_boot(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                       K = 4, kernels = k, R = 200)
bb <- lag_ceiling_bayesboot(lepanthes_hosts, unit = "host", reproduction = "inflorescences",
                            K = 4, kernels = k, draws = 200)
rain_forest(`23 host trees` = b, `23 host trees` = bb, index = "both")
```

---

rain_gini — *Concentration of a series in time*

Description

Indices of how unevenly a non-negative series is spread across its periods. The Gini coefficient is zero when every period carries the same count and approaches one when the whole total falls in a single period. The coefficient of variation is the standard deviation divided by the mean. Both are Schur-convex, which is what makes them usable with the ceiling argument: a weighted average of a series can never be more concentrated, by either index, than the series itself.

Usage

```
rain_gini(x)

rain_cv(x)

concentration(x)
```

Arguments

`x`	A numeric vector of non-negative counts or rates, one per period. `NA` values are dropped.

Details

The Gini coefficient uses the plug-in definition $G = 2\sum_i i x_{(i)}/(n\sum_i x_i) - (n+1)/n$, with $x_{(i)}$ the values in increasing order. It is biased downward in short series (Deltas 2003), but the bias applies equally to the observed series and to the ceiling, so a short series cannot manufacture an exceedance.

Value

`rain_gini()` and `rain_cv()` return a single number (`NA` when the series sums to zero). `concentration()` returns a named numeric vector with `n`, `mean`, `gini`, `cv`, `zeros` (number of periods with a zero count) and `max_share` (the largest single period's share of the total).

Scale and how to read it

Lower end. Both indices are 0 when every period carries the same amount, and both rise as the total is moved into fewer periods. Neither can be negative.

Upper end, which depends on the number of periods and not on the data. The largest value the plug-in Gini can take on `n` periods is $(n-1)/n$, reached when one period carries the whole total: 0.900 on the 10 scored six-monthly censuses, 0.955 on the 22 scored monthly periods. The ceiling of 1 is a limit as `n` grows and is never attained by a finite series (Damgaard and Weiner 2000). `rain_cv()` uses the `n - 1` divisor of `stats::sd()`, so its maximum on `n` periods is $\sqrt{n}$: 3.16 and 4.69 for the same two series. A Gini of 0.65 therefore means something different on 10 periods than on 100, and values are comparable only between series of the same length. Every comparison this package makes, observed against ceiling and observed against simulated, is between series of the same length for that reason.

Bias correction is deliberately not applied. Damgaard and Weiner (2000) recommend multiplying the sample Gini by $n/(n-1)$ for an unbiased estimate. `rain_gini()` returns the uncorrected value because every use of it here is a ratio or a rank between series of equal length, in which a common factor cancels. Multiply by $n/(n-1)$ yourself if you want an estimate of the population value.

No threshold. The package supplies no value of either index above which a series counts as episodic, and takes none from the literature. The level of the index carries no verdict on its own. What carries the inference is the comparison: the index of the observed series against the ceiling (`lag_ceiling()`), and against series simulated under a mean that obeys the ceiling (`ceiling_calibration()`, `host_lag_test()`).

Why these two indices and not others. Both are scale invariant, so the unknown scale of expected recruitment never enters, and both are Schur-convex: moving a quantity from a fuller period to an emptier one lowers them, which is the principle of transfers (Atkinson 1970; Marshall, Olkin and Arnold 2011, ch. 2). Schur-convexity is what makes the ceiling argument go through (Hardy, Littlewood and Polya 1952). They are not interchangeable: two series with the same Gini can have differently shaped Lorenz curves, and Damgaard and Weiner (2000) propose the Lorenz asymmetry coefficient to separate them. `concentration()` therefore reports `zeros` and `max_share` alongside, which say plainly how much of the record is empty and how much of the total falls in its largest period. `max_share` runs from $1/n$ (even) to 1 (everything in one period).

References

Atkinson, A. B. (1970) On the measurement of inequality. *Journal of Economic Theory* 2: 244-263. doi:10.1016/00220531(70)900396

Damgaard, C. and Weiner, J. (2000) Describing inequality in plant size or fecundity. *Ecology* 81: 1139-1142. doi:10.1890/00129658(2000)081[1139:DIIPSO]2.0.CO;2

Deltas, G. (2003) The small sample bias of the Gini coefficient: results and implications for empirical research. *The Review of Economics and Statistics* 85: 226-234. doi:10.1162/rest.2003.85.1.226

Hardy, G. H., Littlewood, J. E. and Polya, G. (1952) *Inequalities*, 2nd edn. Cambridge University Press, Cambridge.

Marshall, A. W., Olkin, I. and Arnold, B. C. (2011) *Inequalities: Theory of Majorization and Its Applications*. Springer, New York. doi:10.1007/9780387682761

Examples

```
rain_gini(c(5, 5, 5, 5))          # 0: perfectly even
rain_gini(c(0, 0, 0, 20))         # 0.75, which is (n - 1) / n: the maximum on 4 periods
rain_gini(c(0, 0, 0, 0, 0, 20))  # 0.833: the same series shape, more periods, higher maximum
rain_cv(c(0, 0, 0, 20))           # 2: the maximum on 4 periods is sqrt(4)

# the two Lepanthes records, each read against its own n
concentration(lepanthes_census$recruits)   # n = 10, so Gini can reach 0.900
concentration(lepanthes_monthly$recruits)  # n = 22, so Gini can reach 0.955
```

`rain_indices` *Compare the ceiling across candidate reproductive measures*

Description

A census usually offers several measures of reproduction: adults, reproductive plants, inflorescences, flowers, fruits, and rates derived from them. They do not give the same ceiling, and the choice can change the exceedance several fold, so it is a decision to report rather than to make quietly. `rain_indices()` runs `lag_ceiling()` once per candidate column and tabulates the result.

Usage

```
rain_indices(
  data,
  reproduction,
  period = "period",
  recruits = "recruits",
  K,
  lag0 = 1L,
  kernels = lag_kernels(K),
  missing = c("backfill", "drop")
)
```

Arguments

`data` A data frame with one row per period, or per unit and period, which is summed.

`reproduction` Character vector of candidate reproductive columns.

`period, recruits` Column names.

`K, lag0, kernels, missing` As in `lag_ceiling()`. The same kernels are used for every candidate, so the comparison is like for like.

Value

A data frame with one row per candidate: the concentration of that reproductive series, the searched ceiling, the exceedance on both indices, the kernel that attained the ceiling, and how many values of that column were missing (and so read as zero).

Scale and how to read it

Read the table down the `exceedance` column, and read it as a sensitivity analysis rather than as a menu. Two things drive the differences.

A stock gives a lower ceiling than a flux. Adults change only by recruitment minus death, so the adult series is smooth whatever the plants are doing, and the exceedance computed from it is the largest available. It is the most favourable index to the paper's own argument and therefore the one to trust least.

Measurement error runs the other way, and protects the lag hypothesis. A reproductive measure that is caught only sometimes, such as a fruit that persists a month against a six-month census, looks more erratic than the truth. That raises its concentration, raises the ceiling, and lowers the exceedance. So a badly observed index makes the test conservative, and a low exceedance from such a column is not evidence that the lag hypothesis survives.

The consequence for reporting: give the whole table, say which columns are well observed in that particular study, and let the reader see the range. No index is the right one in general.

Examples

```
rain_indices(lepanthes_census, c("adults", "inflorescences", "flowers", "fruits"),
             K = 4, kernels = lag_kernels(4, bin = 6, unit = "mo"))
```

`rain_intervals` *Check that the census intervals are what the analysis assumes*

Description

Everything in this package is indexed by period, and a period is assumed to be a fixed length of time: bin 6 months, and lag bin 3 means 18 to 24 months back. Nothing in a table of counts reveals whether that assumption holds, because a data frame of periods 1 to 9 looks the same whether the visits were six months apart or scattered between two months and eighteen. If the study recorded the dates of its visits, this function reads them and says.

Usage

```
rain_intervals(
  data,
  date = "date",
  period = NULL,
  persistence = NULL,
  unit = c("mo", "yr", "day"),
  tolerance = 0.2
)
```

Arguments

`data`	A data frame, or a vector of dates.
`date`	Name of the date column, when `data` is a data frame. The column may be `Date`, `POSIXct`, or a character vector in ISO order (`"2001-06-01"`).
`period`	Name of the period column. If given, one date is expected per period and the function complains if a period carries more than one.
`persistence`	How long the thing being counted stays countable, in the same units as `unit`: about 1.5 months for a *Lepanthes* fruit, 8 for an *Encyclia* fruit, indefinitely for an adult plant. Optional. When given, the ratio of persistence to interval is reported for each interval, which is what decides whether that column was well observed.
`unit`	Time unit for the report, `"mo"` (default), `"yr"` or `"day"`.
`tolerance`	An interval is called irregular when it differs from the median interval by more than this fraction of the median. The default 0.2 allows a visit a fortnight early in a six-monthly schedule. It is a working convention of this package, not a published cutoff, and it decides only which intervals are flagged in the report.

Details

Run it before `lag_ceiling()` on any record whose visits were not on a schedule, and always on a record you did not collect yourself.

Value

An object of class `rain_intervals`, a data frame with one row per interval: the two periods, the two dates, the gap, the ratio to the median gap, and, when `persistence` is given, the persistence to interval ratio. `print()` gives the report.

Scale and how to read it

Three numbers matter, and each answers a different question.

The ratio to the median interval says whether the period index is honest. At 1 the visit fell where the schedule says. Intervals that differ from the median by more than `tolerance` (one fifth of the median by default) are flagged: the period index is then not a time axis, and both the reproduction and the recruit series are then partly a record of how long the crew waited rather than of what the plants did. A short interval catches fewer recruits and fewer fruits than a long one for reasons that have nothing to do with biology, and no kernel can undo that: the convolution is defined on equally

spaced bins. There is no correction inside this package for unequal intervals, and dividing the counts by the interval length is not one, because a count divided by time is a rate and the ceiling argument is about counts.

A negative or zero gap is a data error, not a short interval: the visits are out of order, or a year has been mistyped. It is reported as an error rather than a warning because every downstream number would be meaningless.

The persistence to interval ratio says whether the column was well observed, and it is the criterion for choosing between candidate reproductive measures in `rain_indices()`. At 1 or above, nothing appears and disappears between visits and the count is complete. At 0.25, a fruit that persists six weeks against a six-monthly visit, roughly three in four are never seen, and the series is erratic through being badly seen rather than through being episodic. That inflates its concentration, inflates the ceiling, and lowers the exceedance, so a badly observed index makes the check conservative: a low exceedance computed from one is not evidence that the lag hypothesis survives. Note the direction. Poor observation protects the hypothesis, so it is not a reason to distrust a refutation.

A record whose intervals are unequal is not useless. It can still support the reproductive comparison of `rain_indices()` within a single interval, and it can support anything asked of the whole record at once. What it cannot support is a lag, because a lag is a statement about a number of periods and the periods are not the same length.

Examples

```
# A six-monthly record, as the census intended
rain_intervals(data.frame(period = 1:5,
                          date = as.Date(c("2000-01-01", "2000-07-01", "2001-01-01",
                                           "2001-07-01", "2002-01-01"))),
               persistence = 1.5)
```

`rain_profiles` *Lag profiles, and what they predict on the real record*

Description

`rain_profiles()` draws a set of lag profiles as lollipops. `rain_expected()` applies them to the reproductive record of a `lag_ceiling` object, scales each expected series to the observed total, and draws them over the observed recruits. The point of the pair is that profiles which could not differ more give expected series that all but coincide when reproduction is near-constant, and none resembles an episodic record.

Usage

```
rain_profiles(profiles, bin = NULL, unit = "", lag0 = 1L)

rain_expected(x, profiles, period = "census")
```

Arguments

`profiles`	A named list of weight vectors, or a `lag_kernels` object.
`bin`, `unit`	Bin width and time unit for the lag axis labels; with `bin = NULL` bins are labelled by index from the first lagged bin.
`lag0`	Offset of the first bin, used for the axis labels.
`x`	A `lag_ceiling` object supplying the reproductive record, the recruit series and the lag convention.
`period`	Label for the time axis.

Value

A ggplot.

Examples

```
pr <- list(`one interval only (projection-matrix null)` = c(1, 0, 0, 0, 0),
           `concentrated early` = c(.7, .2, .07, .03, 0),
           `evenly spread` = rep(.2, 5),
           `concentrated late` = c(0, .03, .07, .2, .7))
rain_profiles(pr, bin = 6, unit = "months")
ce <- lag_ceiling(lepanthes_census, reproduction = "inflorescences", K = 4,
                  kernels = lag_kernels(4, bin = 6, unit = "mo"))
rain_expected(ce, pr)
```

`rain_series` *Plot the two series on a common time axis*

Description

Reproduction (blue) and recruitment (vermillion) as bars, one panel each, with the concentration of each printed in the panel title.

Usage

```
rain_series(x, unit = "period")
```

Arguments

`x`	A `lag_ceiling` object.
`unit`	Label for the time axis.

Value

A ggplot.

Examples

```
ce <- lag_ceiling(lepanthes_census, reproduction = "inflorescences", K = 4,
                  kernels = lag_kernels(4, bin = 6, unit = "mo"))
rain_series(ce, unit = "census")
```

---

rain_sightings *First sightings by period, against what dormancy alone would produce*

Description

Bars of the plants first seen at each period, stacked by their state at first sighting, with the expected number of returns of plants present but unseen at the first period drawn over them when `s` and `r` were given.

Usage

```
rain_sightings(x, unit = "period")
```

Arguments

`x`	A `recruit_triage` object.
`unit`	Label for the period axis.

Value

A ggplot.

Scale and how to read it

The vertical axis is a count of plants. Bars far above the line, in the first periods, are the plot being searched or enlarged; bars far above it after several settled periods are the closest thing to recruitment this record offers; bars at or below it are dormancy returns and nothing more. The colour split says whether the plants arrived flowering, which in most orchids means they were established already.

Examples

```
set.seed(7)
M <- matrix(NA_character_, 30, 8, dimnames = list(paste0("p", 1:30), 2000:2007))
for (i in 1:30) { start <- sample(1:5, 1)
  for (j in start:8) M[i, j] <- sample(c("F", "V", NA), 1, prob = c(.4, .35, .25)) }
rain_sightings(recruit_triage(M, species = "C. valida"))
```

rain_strip — *The ceiling strip: observed recruitment against the bound*

Description

Each strip is the concentration axis from 0 to 1. The blue tick is the ceiling (reproduction's own concentration); the region beyond it is unreachable by any delay; the vermillion point is observed recruitment.

Usage

```
rain_strip(..., index = c("gini", "cv"))
```

Arguments

`...`	One or more `lag_ceiling` objects, named to label the strips.
`index`	`"gini"` or `"cv"` (for `"cv"` the axis runs to the largest value shown).

Value

A ggplot.

Examples

```
k6 <- lag_kernels(4, bin = 6, unit = "mo"); k1 <- lag_kernels(6, bin = 1, unit = "mo")
c6 <- lag_ceiling(lepanthes_census, reproduction = "inflorescences", K = 4, kernels = k6)
c1 <- lag_ceiling(lepanthes_monthly, reproduction = "reproductive_adults", K = 6, lag0 = 0,
                  kernels = k1)
rain_strip(`monthly 1994` = c1, `six-monthly 1999-2004` = c6)
```

rain_strips — *Several strip figures instead of one unreadable one*

Description

`rain_strip()` is readable up to about a dozen series. With more units than that, `rain_strips()` splits them and returns one figure per group, either by a grouping you supply (tree species, transect, site) or in chunks of `size`. A group that is still too large is split further, and the figures are named so the caption can say which is which.

Usage

```
rain_strips(
  x,
  group = NULL,
  size = 10,
  index = c("gini", "cv"),
  order_by = c("exceedance", "ceiling", "observed", "name")
)
```

Arguments

x	A lag_ceiling_list from lag_ceiling_by(), or a named list of lag_ceiling objects.
group	Optional grouping for the units: either a vector as long as x, or a named vector or one-to-one lookup whose names are unit names. Units whose group is missing are collected as "ungrouped".
size	Largest number of strips in one figure. Groups larger than this are split into "... (1 of 3)" and so on.
index	"gini" or "cv", passed to rain_strip().
order_by	"exceedance" (default), "ceiling", "observed" or "name": how units are ordered within each figure.

Value

A named list of ggplots. Print them in a loop, one per figure.

Examples

```
ces <- lag_ceiling_by(lepanthes_hosts, unit = "host",
                      reproduction = "inflorescences", K = 4,
                      kernels = lag_kernels(4, bin = 6, unit = "mo"))
figs <- rain_strips(ces, size = 8)
names(figs)
figs[[1]]
```

---

rain_units *Every unit at once: its ceiling against what it recorded*

Description

The companion figure to rain_strip(), for when there are more units than strips can show. Each point is one unit: its own ceiling on the horizontal axis, the concentration of its observed recruitment on the vertical. The diagonal is equality, so a unit above the line recorded recruitment more concentrated than any delay of its own reproduction can predict, and the vertical distance from the line is the size of that gap.

Usage

```
rain_units(x, index = c("gini", "cv"), label = NULL)
```

Arguments

x	A lag_ceiling_list, as returned by lag_ceiling_by(), or a named list of lag_ceiling objects.
index	"gini" or "cv".
label	Units to label, as a character vector of names, or NULL for none. Labelling more than a handful defeats the purpose of this figure.

Value

A ggplot.

Scale and how to read it

Both axes are concentration indices on the scale of rain_gini(), so both run from 0 to a maximum set by the number of periods, and the diagonal is the only line that matters: it is the ceiling. The spread along the horizontal axis is the part people do not expect. It shows how much the bound itself varies between units, because the ceiling is a property of each unit's own reproductive record, and a unit whose reproduction is patchy sits far to the right and can never be refuted.

A unit at ceiling 0, whose reproduction is perfectly flat, sits on the vertical axis. Any recruitment concentration at all is infinitely above its ceiling, which is correct and is why the exceedance ratio is not used as the vertical axis here.

Examples

```
ces <- lag_ceiling_by(lepanthes_hosts, unit = "host",
                      reproduction = "inflorescences", K = 4,
                      kernels = lag_kernels(4, bin = 6, unit = "mo"))
rain_units(ces)
```

---

recruit_triage — *Are the first sightings recruits? Dormancy, detection and the discovery series*

Description

A tagged-plant census records, for each plant, the state it was in at each period: flowering, vegetative, not seen. The recruit series that such a record offers is the number of plants seen for the first time at each period, and that series is a recruitment series only if a first sighting is a new plant. Two things break the equivalence. Dormancy: a plant that was there all along can stay underground, or be missed, for a year or more and then be "discovered". Search: a plot that is enlarged, or searched harder, discovers established plants in bulk. This function puts the numbers a reader needs to judge a first-sighting series beside it.

Usage

```
recruit_triage(
  x,
  plant = "plant",
  period = "period",
  state = "state",
  seen = c("F", "V", "1", "2"),
  flowering = c("F", "2"),
  s = NULL,
  r = NULL,
  species = NULL
)
```

Arguments

`x`	A matrix with one row per plant and one column per period, or a long data frame with one row per plant and period.
`plant`, `period`, `state`	Column names when `x` is a long data frame.
`seen`	The state codes that mean the plant was above ground. The default covers `F`/`V` and `1`/`2` codings. Anything else (`NA`, `"NA"`, `"D"`, `0`, blank) is read as not seen.
`flowering`	The codes, within `seen`, that mean the plant was flowering.
`s`, `r`	Annual survival and the probability of being seen given survival, used to compute how many first sightings the record's own dormancy would produce from plants present at the first period. Supply them from a capture-recapture study of the species (for nine *Caladenia* species, caladenia_dormancy carries the published values) or leave `r` as `NULL` to use the record's own estimate (below) with the `s` you supply; leave both `NULL` and that table is not computed.
`species`	A row name of caladenia_dormancy, as a shortcut for `s` and `r`.

Value

An object of class `recruit_triage` with: `first` (one row per period: plants first seen, split by state at first sighting, the number of periods each had been watched for, the expected number of first sightings that are returns of plants present but unseen at the first period, and the excess over that), `dormancy` (the run-length table of unseen periods between sightings, as in Coates et al. 2006 Table 2, the share of plants with at least one such run, and the longest), `detection` (the record's own estimate of `r`: the share of periods between a plant's first and last sighting at which it was seen), `watched` (how many periods the plot had been watched before each first sighting) and the inputs.

Scale and how to read it

State at first sighting. A plant that is flowering when first seen has, in most orchids, been established for years. A first-sighting series in which most plants arrive flowering is a discovery series, not a recruitment series, and the share flowering is the first number to read. In *Lepanthes rubripetala* it was 29 of 43; in the Victorian terrestrials it ran from 57% to 96%.

The dormancy runs. The table counts, for every plant, every run of consecutive unseen periods that falls between two sightings, so each run is a plant known to have been alive and present while unseen. The share of plants with at least one run, and the longest run, say how much a first sighting after a few years of watching can be trusted. Runs at the end of a plant's record are not counted: a plant not seen again may be dead, and the record cannot tell.

The record's own detection estimate. `detection` is the share of periods, between each plant's first and last sighting, at which it was seen: an estimate of `r`, the probability of being seen given survival, by the conventional method (Kery et al. 2005; Tremblay et al. 2009). It is biased upward, because a plant is only known to be alive while it is being resighted, and it says nothing about survival.

Expected returns. With `s` and `r` given, a plant alive at period 1 was unseen there with probability `1 - r`, survives and stays unseen for a further period with probability `q = s (1 - r)`, and is seen at period `k` with probability `s r` after `k - 2` such periods. If `N1` plants were seen at period 1, then `N1 / r` were alive, `N1 (1 - r) / r` of them were unseen, and the expected number of those discovered at period `k` is `N1 (1 - r) s q^(k - 2)`. That is the number of first sightings that the record's own dormancy accounts for, and it falls fast: for *Caladenia valida* (survival 0.82, resighting 0.73) it is 22% of `N1` at the second period, 5% at the third and 1% at the fourth. The `excess` column is what is left after subtracting it, and is the most that recruitment plus search effort can have contributed. A large excess in a period in which the plot was enlarged is search; a large excess after several settled years of watching is the nearest thing to a recruit count this kind of record can give. Returns of plants that arrived after period 1 are not modelled: those are new plants whenever they are first seen, only late. The rates `s` and `r` are per period of the record: the `caladenia_dormancy` rates are annual, so they apply to an annual census and not to one made twice a year.

Improbable runs. With `s` and `r` given, a run of `k` unseen periods between two sightings has probability `q^k` for a living plant. Runs long enough that `q^k` falls below 0.01 are reported in `dormancy` as `improbable_runs`, from the run length `improbable_from` on: more likely a tag reused for a new plant than one plant dormant that long. The 0.01 is a working convention of this package for which runs to point at, not a published cutoff, and the runs are counted whatever it is set to.

What this does not do. It does not make a first-sighting series into a recruitment series. It says how far the two are apart, and where. A record with a seedling or protocorm class does not need it.

References

Coates, F., Lunt, I. D. and Tremblay, R. L. (2006) Effects of disturbance on population dynamics of the threatened orchid *Prasophyllum correctum* D.L. Jones and implications for grassland management in south-eastern Australia. *Biological Conservation* 129: 59-69. doi:10.1016/j.biocon.2005.06.037

Kery, M., Gregg, K. B. and Schaub, M. (2005) Demographic estimation methods for plants with unobservable life-states. *Oikos* 108: 307-320. doi:10.1111/j.00301299.2005.13589.x

Tremblay, R. L., Perez, M.-E., Larcombe, M., Brown, A., Quarmby, J., Bickerton, D., French, G. and Bould, A. (2009) Dormancy in *Caladenia*: a Bayesian approach to evaluating latency. *Australian Journal of Botany* 57: 340-350. doi:10.1071/BT08163

Examples

```
# a small invented record: 30 plants, 8 years, F/V/NA
set.seed(7)
M <- matrix(NA_character_, 30, 8, dimnames = list(paste0("p", 1:30), 2000:2007))
for (i in 1:30) { start <- sample(1:5, 1)
```

```
  for (j in start:8) M[i, j] <- sample(c("F", "V", NA), 1, prob = c(.4, .35, .25)) }
tr <- recruit_triage(M, species = "C. valida")
tr
rain_sightings(tr)
```

`regimes` *Six invented census records spanning the cases the test separates*

Description

Constructed records, not data, in one long tibble. Three reproductive regimes (seasonal, intermediate, aseasonal) crossed with two recruitment mechanisms. Under `"delayed"`, recruits on each unit are a true delayed consequence of that unit's own reproduction: a fixed lag profile over five bins (weights 0.10, 0.25, 0.35, 0.20, 0.10 at one to five periods back) with negative-binomial counts. Under `"gated"`, recruits arrive only at three of the 24 periods, when an external gate is open, at a rate proportional to each unit's average reproduction, so the record is episodic and does not track reproduction. Each record has 20 units and 24 periods; recruits are scored at periods 6 to 24 (`K = 4`, `lag0 = 1`) and are `NA` elsewhere. The generating script is `data-raw/regimes.R` in the source repository (https://github.com/RaymondLTremblay/recruitlag).

Format

`regimes` has one row per record, unit and period (2,880 rows): `record` (`seasonal_delayed`, `seasonal_gated`, `intermediate_delayed`, `intermediate_gated`, `aseasonal_delayed`, `aseasonal_gated`), `label`, `seasonality` (0, 0.5 or 1), `mechanism`, `unit`, `period`, `reproduction` and `recruits`. `regimes_truth` holds the generating rules: the five weights of the true lag profile and the 24 gate values.

Examples

```
r <- regimes[regimes$record == "aseasonal_gated", ]
ce <- lag_ceiling(r, K = 4, kernels = lag_kernels(4))
ce
```

`series_from` *Pull the two system-wide series from a data frame*

Description

Sums (or takes, if there is one row per period) reproduction and recruits by period. Periods at which `recruits` is `NA` in every row are unscored; a period at which some rows are `NA` and others carry a count is summed over the rows with a count, with a warning.

Usage

```
series_from(
  data,
  period = "period",
  reproduction = "reproduction",
  recruits = "recruits"
)
```

Arguments

`data` A data frame with one row per period, or per unit and period.

`period`, `reproduction`, `recruits`
Names of the columns holding the period index (integer, from 1), the reproductive count and the recruit count.

Value

A list with `X` (reproduction at periods 1 to the last), `R` (recruits at the scored periods) and `t_R` (those periods).

Examples

```
series_from(lepanthes_census, reproduction = "inflorescences")
```

Index

∗ **datasets**
caladenia_dormancy, 3

caladenia_dormancy, 3, *43*
ceiling_calibration, 4
ceiling_calibration(), *15*, *22*, *33*
ceiling_draws_table, 5
concentration *(*rain_gini*)*, 32
convolve_lag, 6
convolve_lag(), *8*, *11*, *14*, *17, 18*, *20*, *23*, *25*

expected_recruits, 8
expected_recruits(), *10*, *28, 29*

flat_kernel, 9

host_lag_test, 10
host_lag_test(), *4, 5*, *9, 10*, *15*, *19*, *27*, *29*, *33*
host_matrices, 13
host_matrices(), *8*, *11*

lag_ceiling, 14
lag_ceiling(), *5*, *7*, *17*, *19–22*, *24*, *33–36*
lag_ceiling_bayesboot, 16
lag_ceiling_bayesboot(), *5*, *19*, *23, 24*
lag_ceiling_boot, 18
lag_ceiling_boot(), *5*, *16, 17*
lag_ceiling_by, 20
lag_ceiling_by(), *41, 42*
lag_ceiling_stan, 22
lag_ceiling_stan(), *5*, *17*
lag_kernels, 25
lag_kernels(), *9–11*, *14*, *17, 18*, *20*, *23*, *28*
lepanthes, 27
lepanthes_census *(*lepanthes*)*, 27
lepanthes_hosts *(*lepanthes*)*, 27
lepanthes_monthly *(*lepanthes*)*, 27
lepanthes_profile *(*lepanthes*)*, 27

phi_moment, 28
phi_moment(), *4*, *9, 10*
plot.ceiling_calibration, 30
plot.host_lag_test *(*plot.ceiling_calibration*)*, 30
plot.host_lag_test(), *12*

rain_cv *(*rain_gini*)*, 32
rain_cv(), *15*
rain_expected *(*rain_profiles*)*, 37
rain_forest, 31
rain_gini, 32
rain_gini(), *15*, *42*
rain_indices, 34
rain_indices(), *37*
rain_intervals, 35
rain_profiles, 37
rain_series, 38
rain_sightings, 39
rain_strip, 40
rain_strip(), *21*, *40, 41*
rain_strips, 40
rain_units, 41
recruit_triage, 42
recruit_triage(), *3*
regimes, 45
regimes_truth *(*regimes*)*, 45

series_from, 45
stats::sd(), *33*