

Package: kalmix (via r-universe)

October 10, 2026

Type Package

Title General State-Space Modelling, Filtering and Change-Point Detection

Version 0.6.0

Description A compact, dependency-light toolkit for linear-Gaussian and discrete-state state-space models. Provides a vector-state Kalman filter and Rauch-Tung-Striebel smoother in closed form, a native Gaussian-mixture (regime-switching) filter, hidden Markov model filtering and Viterbi decoding, a likelihood-ratio change-point detector, and a minimal single-case interrupted-time-series causal contrast built on the smoother. Mean-reverting spread estimation for pairs trading is included as one documented application of the general machinery, building on the scalar Ornstein-Uhlenbeck spread model of Elliott, van der Hoek and Malcolm (2005) [doi:10.1080/14697680500149370](https://doi.org/10.1080/14697680500149370). Methods follow Kalman (1960) [doi:10.1115/1.3662552](https://doi.org/10.1115/1.3662552), Rauch, Tung and Striebel (1965) [doi:10.2514/3.3166](https://doi.org/10.2514/3.3166) and Durbin and Koopman (2012) [doi:10.1093/acprof:oso/9780199641178.001.0001](https://doi.org/10.1093/acprof:oso/9780199641178.001.0001).

License MIT + file LICENSE

URL <https://github.com/max578/kalmix>

BugReports <https://github.com/max578/kalmix/issues>

Encoding UTF-8

Language en-AU

Depends R (>= 4.3)

Imports S7, stats, cli

Suggests acir, ggplot2, kernR, mvtnorm, testthat (>= 3.1.5), withr, knitr, rmarkdown

Remotes max578/kernR,
biometryhub/ACI/acir@dbec5a1ee5f3ec9b8863260d2e35e52f02702b6c

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Collate 'linalg-internal.R' 'ssm-internal.R' 'ssm-class.R'
 'student-t-internal.R' 'kalman.R' 'ssm-fit.R'
 'mixture-internal.R' 'mixture.R' 'mixture-smoother.R'
 'hmm-internal.R' 'hmm.R' 'changepoint.R' 'its-causal.R'
 'adequacy.R' 'student-t.R' 'aci.R' 'cir.R' 'kalmix-classes.R'
 'spread.R' 'filter.R' 'kalmix-package.R' 'zzz.R'

Config/roxygen2/version 8.1.0

Repository <https://max578.r-universe.dev>

Date/Publication 2026-10-09 22:30:22 UTC

RemoteUrl <https://github.com/max578/kalmix>

RemoteRef main

RemoteSha 784634e8e95f5d664657848a1a576a801d9cfe84

Contents

aci	3
aci_fit	6
causal_information_rate	7
changepoint_fit	10
detect_changepoint	11
estimate_obs_df	13
half_life	14
hmm	15
hmm_filter	16
hmm_fit	17
hmm_viterbi	18
innov_diag	19
innovation_diagnostics	20
its_causal	21
its_fit	23
kalman_filter	24
kalman_fit	25
kalmix_filter	26
mixture_filter	28
mixture_smoother	29
ou_fit	30
regime_fit	31
regime_smooth	32
rts_fit	33
rts_smoother	34
spread_model	35
spread_series	36
ssm	37

<i>aci</i>	3
ssm_fit	39
ssm_mle	41
Index	43

<i>aci</i>	<i>Assimilative Causal Inference read-out for a state-space model</i>
------------	---

Description

Runs the closed-form Kalman filter and Rauch-Tung-Striebel smoother of `kalman_filter()` and `rts_smoother()` over an `ssm`, then reads the time-resolved causal information series off the two passes following Andreou, Chen and Bollt (2026). The causal information at each step is the relative entropy of the smoothing distribution of the latent state from its filtering distribution,

$$I_t = D_{\text{KL}}(N(x_{t|n}, P_{t|n}) \parallel N(x_{t|t}, P_{t|t})),$$

the information the future of the series adds to the present estimate of the state (their eq. 7). The series is summarised by its time-average, the mean causal information.

Usage

```
aci(
  model,
  y,
  lead_time = TRUE,
  dt = 1,
  eval_points = NULL,
  max_lag = NULL,
  n_lag = NULL,
  engine = c("online", "expanding"),
  functional = c("objective", "objective_exact"),
  mechanism = NULL,
  alpha = 0.05,
  transition = NULL,
  init_prob = NULL
)
```

Arguments

<code>model</code>	An <code>ssm</code> , or a list of two or more <code>ssm</code> regimes for a regime-switching read-out.
<code>y</code>	A numeric vector or n by d matrix of observations.
<code>lead_time</code>	Logical scalar – whether to compute the objective causal information rate and lead-time as well as the causal information series. Defaults to TRUE.
<code>dt</code>	Numeric scalar – the sampling interval, so the lead-time is in meaningful time units (for example years for an annual ENSO index). Defaults to 1, giving the lead-time in steps.

<code>eval_points</code>	An optional integer vector of step indices at which to anchor the lead-time computation; passed to <code>causal_information_rate()</code> . Defaults to NULL, which spreads a handful of anchors across the interior of the series.
<code>max_lag</code>	Integer scalar – the largest future-window lag, in steps, used for the lead-time. Defaults to one fifth of the series length.
<code>n_lag</code>	Integer scalar or NULL – the number of window lengths the divergence profile behind the lead-time is evaluated at; passed to <code>causal_information_rate()</code> , which also checks the quadrature has converged. Defaults to NULL (every lag, capped at 65 points).
<code>engine</code>	Character – the lead-time engine, "online" (default) or "expanding"; passed to <code>causal_information_rate()</code> .
<code>functional</code>	Character – "objective" (default) for the efficient integral form of the range, "objective_exact" for the threshold-averaged form; passed to <code>causal_information_rate()</code> .
<code>mechanism</code>	Optional declared provenance for the model as the data-generating mechanism: NULL (the default, an unverified mechanism), or a list carrying a <code>verified_on</code> Date on which the model was checked against the external authority that owns it. Only a verified mechanism can raise an adequate read-out to a "grounded" verdict.
<code>alpha</code>	Numeric scalar – the significance level the model-adequacy diagnostics abstain at. Defaults to 0.05.
<code>transition, init_prob</code>	The regime-transition matrix and initial regime distribution when model is a list of regimes (see <code>mixture_filter()</code>); ignored for an <code>ssm</code> . Default to NULL.

Details

By default the objective causal information rate and the decision lead-time are also computed (`lead_time = TRUE`), through `causal_information_rate()`: the divergence of the complete smoother from an expanding-future-window smoother is integrated into a single threshold-free lead-time in the time units of the series (their eqs. 8–9). Set `lead_time = FALSE` to return the causal information series alone, which is cheaper for long series.

Under a Student-t observation family (`obs_family = "student_t"`) the filter pass is the outlier-robust variational update, whose per-step posteriors are Gaussian approximations. The relative-entropy chain treats those approximations as exact Gaussians, so the causal information and lead-time inherit the variational approximation; the Gaussian family involves no approximation.

A regime-switching model is supplied as a list of `ssm` regimes (with the optional `transition` and `init_prob` of `mixture_filter()`): the read-out then compares the Kim smoother's collapsed posteriors with the GPB1 filter's. Both are the model class's own Gaussian collapses of the true mixtures, so the causal information inherits that approximation; the verdict is grounded through the mixture-PIT diagnostics, and the lead-time uses the expanding-window construction (the online recursion is linear-Gaussian only).

The recovered state is interpretable as a cause of the observed series only under the maintained assumption that the `ssm` is the data-generating mechanism. That assumption is not taken on trust: the read-out is grounded through `innovation_diagnostics()`, and a model whose standardised innovations are not white, not correctly scaled, or not Gaussian is declared inadequate, so the verdict abstains with an "ungrounded" grounding token and the returned `aci_fit` additionally carries the

`c("orchestra_refusal", "kalmix_abstention")` classes (as does a fit whose adequacy could not be judged at all, or whose lead-time is censored or unconverged – see censored/the quadrature convergence check), so a cross-member caller can recognise the read-out as unreliable without parsing the grounding string. Passing the diagnostics establishes self-consistency only; the verdict is labelled "grounded" solely when an adequate model is accompanied by declared, dated mechanism provenance, and is "ungrounded" otherwise – that ordinary, expected case is not itself abstained on.

Value

An [aci_fit](#).

References

- Andreou, M., Chen, N. and Bollt, E. (2026). Assimilative causal inference. *Nature Communications*, 17, 1854.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45.
- Rauch, H. E., Tung, F. and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8), 1445–1450.

See Also

[causal_information_rate\(\)](#) for the lead-time alone; [kalman_filter\(\)](#) and [rts_smoother\(\)](#) for the underlying passes.

Other causal: [aci_fit\(\)](#), [causal_information_rate\(\)](#), [estimate_obs_df\(\)](#), [innov_diag\(\)](#), [innovation_diagnostics\(\)](#), [its_causal\(\)](#), [its_fit\(\)](#)

Examples

```
## A latent oscillation seen through a slow, noisy observation channel: the
## future of the series is informative about the hidden state, so the causal
## information is positive.
model <- ssm(
  transition = matrix(c(0.6, -0.5, 0.5, 0.6), nrow = 2),
  observation = matrix(c(1, 0), nrow = 1),
  state_cov = diag(c(0.2, 0.2)),
  obs_cov = 1,
  init_state = c(0, 0),
  init_cov = diag(c(5, 5))
)
set.seed(1)
x <- numeric(200)
s <- c(0, 0)
for (t in seq_len(200)) {
  s <- as.numeric(model@transition[[1]] %*% s) + rnorm(2, sd = 0.45)
  x[t] <- s[1] + rnorm(1)
}
fit <- aci(model, x, lead_time = FALSE)
fit@mean_causal_information
```

aci_fit

*An Assimilative Causal Inference read-out***Description**

An S7 object holding the output of `aci()`: the per-step causal information series, its time aggregate (the mean causal information), the objective causal information rate and the implied decision lead-time, alongside the filter and smoother passes the read-out was computed from.

Usage

```
aci_fit(
  causal_information = numeric(0),
  mean_causal_information = numeric(0),
  causal_information_rate = NA_real_,
  lead_time = NA_real_,
  dt = 1,
  filter = NULL,
  smoother = NULL,
  grounding = "ungrounded",
  grounding_reason = NA_character_,
  adequacy = NULL,
  censored = FALSE,
  monotone = NA
)
```

Arguments

<code>causal_information</code>	A numeric vector of length n – the per-step causal information series I_t , the smoother-from-filter relative entropy at each step.
<code>mean_causal_information</code>	Numeric scalar – the time-average of <code>causal_information</code> , a single summary of how much the future informs the state over the series.
<code>causal_information_rate</code>	Numeric scalar – the objective, threshold-free causal information rate (a lead-time in the time units of the series), or <code>NA_real_</code> if it was not requested.
<code>lead_time</code>	Numeric scalar – the decision lead-time, an alias of the causal information rate in time units, or <code>NA_real_</code> if not requested.
<code>dt</code>	Numeric scalar – the sampling interval the lead-time is expressed in. Defaults to 1, so the lead-time is in steps.
<code>filter</code>	The forward pass the read-out was computed from: a <code>kalman_fit</code> , or a <code>regime_fit</code> for a regime-switching model.
<code>smoother</code>	The smoothing pass: an <code>rts_fit</code> , or a <code>regime_smooth</code> for a regime-switching model.

grounding	Character scalar – the grounding token of the causal verdict, "grounded" or "ungrounded"; see the token definitions in innovation_diagnostics() .
grounding_reason	Character scalar – a short label for why the verdict carries that token ("grounded", "mechanism_unverified", "model_inadequate" or "insufficient_data").
adequacy	The innov_diag model-adequacy diagnostics the grounding was read from.
censored	Logical scalar – whether any lead-time anchor was right-censored, that is whether its future window ran past the end of the record or its divergence profile had not decayed by the last lag it could be evaluated at. A censored lead-time is a lower bound, not an estimate, and degrades the grounding token.
monotone	Logical scalar – whether every anchor's divergence profile decreases with lag, the condition under which the efficient and exact range functionals are the same functional.

Details

The causal information at step t is the relative entropy of the smoothing distribution from the filtering distribution of the latent state,

$$I_t = D_{\text{KL}}(N(x_{t|n}, P_{t|n}) \parallel N(x_{t|t}, P_{t|t})),$$

the information the future of the series adds to the present estimate of the state. It is non-negative and zero only where the future is uninformative about the state at t .

Value

An S7 object of class `aci_fit`.

See Also

Other causal: [aci\(\)](#), [causal_information_rate\(\)](#), [estimate_obs_df\(\)](#), [innov_diag\(\)](#), [innovation_diagnostics\(\)](#), [its_causal\(\)](#), [its_fit\(\)](#)

causal_information_rate

Objective causal influence range and decision lead-time

Description

Computes the objective, threshold-free causal influence range of Andreou, Chen and Bollt (2026, eqs. 8–9) for a state-space model: the effective horizon over which the future of the observed series keeps informing the estimate of the latent state, returned as a decision lead-time in the time units of the series.

Usage

```
causal_information_rate(
  model,
  y,
  dt = 1,
  eval_points = NULL,
  max_lag = NULL,
  n_lag = NULL,
  engine = c("online", "expanding"),
  functional = c("objective", "objective_exact"),
  epsilon = .cir_epsilon_grid(),
  margin = 0.1,
  tol = 0.02,
  transition = NULL,
  init_prob = NULL
)
```

Arguments

<code>model</code>	An ssm .
<code>y</code>	A numeric vector or matrix of observations.
<code>dt</code>	Numeric scalar – the sampling interval; the returned lead-time is in these time units. Defaults to 1.
<code>eval_points</code>	An optional integer vector of anchor step indices. Defaults to NULL, which places anchors across the interior of the series (avoiding the ends, where the future window is short).
<code>max_lag</code>	Integer scalar – the largest future-window lag in steps. Defaults to one fifth of the series length, capped so an anchor still has future to integrate over.
<code>n_lag</code>	Integer scalar or NULL – the number of window lengths the divergence profile is evaluated at, including zero. Defaults to NULL, which evaluates every lag in the window, the finest grid the record supports, capped at 401 points. A thinned grid over-estimates the integral of a sharply decaying profile, so whenever the grid is thinned the quadrature is compared with the same reduction on every other point and a warning is issued when the two disagree by more than <code>tol</code> in relative terms.
<code>engine</code>	Character – "online" (default) computes the lead-time with a fixed-point smoother recursion that avoids re-smoothing and handles time-varying models; "expanding" reruns the smoother on each truncated series and requires a time-invariant model. The two agree to numerical precision and cross-check the smoother recursion. A regime-switching model always uses the expanding construction (the online recursion is linear-Gaussian only) and engine is ignored.
<code>functional</code>	Character – "objective" (default) for the efficient integral form, "objective_exact" for the threshold-averaged form. See the details.
<code>epsilon</code>	Numeric vector – the tolerance grid the exact functional averages the subjective range over, in nats. Defaults to the 129-point logarithmic grid $10^{\text{seq}(-6, 0.5, \text{length.out} = 129)}$, matching <code>acir</code> 's former default. Ignored by functional = "objective".

margin	Numeric scalar in $(0, 1)$ – the fraction of its peak the profile must have decayed below by its last evaluated lag for the anchor to count as resolved. Defaults to 0.1.
tol	Numeric scalar – the relative disagreement between the quadrature and the same quadrature on half the grid points above which a warning is issued. Defaults to 0.02. Set to Inf to silence the check.
transition, init_prob	The regime-transition matrix and initial regime distribution when model is a list of regimes; passed to <code>mixture_filter()</code> and <code>mixture_smoother()</code> . Ignored for an <code>ssm</code> .

Details

At an anchor step t_0 the smoother is recomputed on an expanding future window: the relative entropy $D(L)$ of the complete smoother from the smoother that has seen the series only up to $t_0 + L$ (their eq. 8, with the complete smoother as the integrating density) falls from its filter value at $L = 0$ – the per-step causal information of eq. 7 – towards zero as the window grows.

Two functionals reduce that profile to a range, and they are different functionals rather than two quadratures of one. The subjective range at tolerance ε is the *last* lag at which the profile still exceeds ε , $\tau_\varepsilon = \sup\{L : D(L) > \varepsilon\}$, and functional = "objective_exact" averages it over the tolerance, $M^{-1} \int_0^M \tau_\varepsilon d\varepsilon$ with $M = \max_L D(L)$. The default functional = "objective" is the computationally efficient underestimate the source paper gives,

$$\tau(t_0) = \frac{1}{M} \int_0^{L_{\max}} D(L) dL.$$

The two coincide when the profile decreases with lag, by the layer-cake identity, and the integral form is strictly the smaller as soon as it does not; a profile that rises before it decays is the ordinary case near the start of a record. Monotonicity is assumed for convenience in the source paper and is not required by the theory, so "objective" is reported as a lower bound and `attr(x, "monotone")` records whether the two agree. The range is averaged over the anchors and scaled to time units by `dt`.

Two engines compute the same divergence profile. The default "online" engine grows the future window through a fixed-point smoother recursion: the smoothed estimate of the anchor state is updated in place as each later observation is assimilated, by accumulating the Rauch-Tung-Striebel gains, so the whole profile costs one filter-smoother pass plus `length(eval_points)` light forward recursions and no re-smoothing. It is exact and applies to time-varying models. The "expanding" engine instead reruns the filter and smoother on each truncated series – the direct construction of the published equations, retained as a cross-check on the smoother recursion and restricted to time-invariant models. The two share the quadrature and the reduction, so their agreement grades the recursion and not the range functional; the independent grading of the functional is the conformance suite against `acir`.

An anchor whose future window runs past the end of the record cannot be resolved. The profile is then integrated over the lags the record does support and no further – the value is a lower bound, not an estimate – and the anchor is marked censored. So is an anchor whose profile has not decayed to within `margin` of its peak by the last lag it could be evaluated at, since the window was too short to see the influence expire. `attr(x, "censored")` is TRUE when any anchor is censored and `attr(x, "censored_fraction")` gives the proportion that are.

Value

Numeric scalar – the objective causal influence range (a lead-time in the time units set by `dt`), 0 where no future information is recoverable, carrying the attributes `censored` (logical), `censored_fraction` (numeric), `monotone` (logical, whether every anchor's profile decreases with lag), `n_lag` (the grid size used) and `converged` (logical, whether a thinned grid agreed with this one to within `tol`; NA when the check does not apply, for example a fully resolved grid).

References

Andreou, M., Chen, N. and Bollt, E. (2026). Assimilative causal inference. *Nature Communications*, 17, 1854.

See Also

`aci()`.

Other causal: `aci()`, `aci_fit()`, `estimate_obs_df()`, `innov_diag()`, `innovation_diagnostics()`, `its_causal()`, `its_fit()`

Examples

```
model <- ssm(
  transition = matrix(c(0.6, -0.5, 0.5, 0.6), nrow = 2),
  observation = matrix(c(1, 0), nrow = 1),
  state_cov = diag(c(0.2, 0.2)),
  obs_cov = 1,
  init_state = c(0, 0),
  init_cov = diag(c(5, 5))
)
set.seed(1)
x <- numeric(200)
s <- c(0, 0)
for (t in seq_len(200)) {
  s <- as.numeric(model@transition[[1]] %*% s) + rnorm(2, sd = 0.45)
  x[t] <- s[1] + rnorm(1)
}
causal_information_rate(model, x)
```

changepoint_fit

A change-point detection result

Description

An S7 object holding the output of `detect_changepoint()`: the detected change-point locations, the per-segment means, and the penalised criterion the search used.

Usage

```

changepoint_fit(
  changepoints = integer(0),
  n_segments = integer(0),
  segment_means = numeric(0),
  penalty = numeric(0),
  method = character(0)
)

```

Arguments

changepoints	An integer vector of the detected change-point indices (the last index of each segment but the last); empty if no change is supported.
n_segments	Integer scalar — the number of segments, <code>length(changepoints) + 1</code> .
segment_means	A numeric vector of the fitted mean of each segment.
penalty	Numeric scalar — the per-split penalty that was applied.
method	Character scalar — the detection method used.

Value

An S7 object of class `changepoint_fit`.

See Also

Other change-point: [detect_changepoint\(\)](#)

<code>detect_changepoint</code>	<i>Detect change-points in the mean of a sequence</i>
---------------------------------	---

Description

Finds shifts in the mean of a Gaussian sequence by binary segmentation on the single-change-point likelihood-ratio statistic. At each stage the segment is split at the location whose two-mean fit most improves the Gaussian fit over a single-mean fit; the split is kept only when the improvement clears a penalty, otherwise the segment is declared homogeneous.

Usage

```
detect_changepoint(x, penalty = "mbic", min_segment = 5L)
```

Arguments

x	A numeric vector — the sequence to scan.
penalty	Numeric scalar, "mbic", "sic", or "bic". A number is used directly as the per-split penalty on the statistic; "mbic" (the default) uses $3 \log(n)$ and "sic" (alias "bic") uses $\log(n)$. Larger penalties yield fewer change-points.
min_segment	Integer scalar — the shortest admissible segment length, so a single outlier is not declared its own segment. Defaults to 5.

Details

The single-change statistic at split τ is the drop in the Gaussian twice-negative-profile-log-likelihood, computed in closed form from the segment sums and sums of squares,

$$n \log \hat{\sigma}_{1:n}^2 - \tau \log \hat{\sigma}_{1:\tau}^2 - (n - \tau) \log \hat{\sigma}_{\tau+1:n}^2,$$

the variance being estimated within each candidate segment.

The penalty matters more than the statistic. Under the null of one homogeneous segment the best-split statistic is far from zero – on a length-100 series its null mean is near 7 and its 95th percentile near 13 – so the plain Schwarz penalty $\log(n)$ (about 4.6 at that length) over-segments badly. The default "mbic" penalty here is an empirically calibrated $3 \log(n)$, strengthened in the spirit of the modified Bayesian information criterion of Zhang and Siegmund (2007) to account for the change location being estimated (their criterion is a different functional; the name is kept for the shared motivation). On homogeneous Gaussian series it was calibrated to a false-detection rate at or below 6% from $n = 100$ upward while retaining full power against a one-standard-deviation mean shift. The weaker "sic" ($\log(n)$, alias "bic") penalty and any numeric value are offered for callers who want to tune sensitivity deliberately, accepting more false positives.

Value

A [changepoint_fit](#).

References

Scott, A. J. and Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3), 507–512. (Binary segmentation.) Killick, R., Fearnhead, P. and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500), 1590–1598.

Zhang, N. R. and Siegmund, D. O. (2007). A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1), 22–32. (The motivation for strengthening the per-split penalty; the "mbic" value here is an empirically calibrated $3 \log(n)$, not their functional.)

See Also

Other change-point: [changepoint_fit\(\)](#)

Examples

```
set.seed(1)
x <- c(rnorm(50, 0), rnorm(50, 4), rnorm(50, 1))
fit <- detect_changepoint(x)
fit@changepoints
```

estimate_obs_df

Choose the Student-t degrees of freedom by profile likelihood

Description

Profiles the one-step-ahead predictive log-likelihood of a state-space model over a grid of Student-t degrees of freedom and returns the maximiser. The grid includes Inf (the Gaussian model), so a series whose innovations are not heavy-tailed selects Inf and the model collapses to the linear-Gaussian filter – the estimator never imposes heavy tails that the data do not show.

Usage

```
estimate_obs_df(
  model,
  y,
  df_grid = c(3, 4, 5, 7, 10, 15, 30, Inf),
  lrt_margin = stats::qchisq(0.95, 1)/2
)
```

Arguments

<code>model</code>	An ssm giving the dynamics and the observation scale. Its own <code>obs_family</code> is ignored; each grid value is evaluated on a clone.
<code>y</code>	A numeric vector or n by d matrix of observations.
<code>df_grid</code>	A numeric vector of candidate degrees of freedom to profile over. Every value must exceed 2 (the ssm validator requires a finite observation variance); may include Inf. Defaults to <code>c(3, 4, 5, 7, 10, 15, 30, Inf)</code> .
<code>lrt_margin</code>	The parsimony margin in log-likelihood units. The Gaussian model (<code>df = Inf</code>) is preferred unless a finite <code>df</code> improves the log-likelihood by more than this much, so a finite Gaussian sample – whose profile peaks at a large but finite <code>df</code> by chance – still selects the Gaussian. Defaults to half the 95% one-degree-of-freedom chi-squared quantile (a likelihood-ratio test at the 5% level), about 1.92.

Details

The supplied `model` provides the dynamics (transition, process noise, prior) and the observation *scale* `obs_cov`; only the observation family and its degrees of freedom are varied. The log-likelihood is the Student-t predictive density accumulated by `kalman_filter()` (an approximation that treats the one-step-ahead distribution as Student-t; see `kalman_filter()`).

Value

A list with `df` (the selected degrees of freedom), `family` ("gaussian" if `df` is Inf, else "student_t"), `log_lik` (the maximised log-likelihood), `grid` (the grid evaluated) and `profile` (the log-likelihood at each grid value).

References

Roth, M., Ozkan, E. and Gustafsson, F. (2013). A Student's t filter for heavy tailed process and measurement noise. *ICASSP*, 5770–5774.

See Also

[ssm\(\)](#) for building the chosen model; [aci\(\)](#) for the causal read-out the chosen family grounds.

Other causal: [aci\(\)](#), [aci_fit\(\)](#), [causal_information_rate\(\)](#), [innov_diag\(\)](#), [innovation_diagnostics\(\)](#), [its_causal\(\)](#), [its_fit\(\)](#)

Examples

```
## A heavy-tailed series: the profile prefers a finite df over the Gaussian.
model <- ssm(
  transition = 1, observation = 1,
  state_cov = 0.04, obs_cov = 1,
  init_state = 0, init_cov = 10
)
set.seed(1)
level <- cumsum(rnorm(300, sd = 0.2))
y <- level + rt(300, df = 3)
est <- estimate_obs_df(model, y)
est$df
```

half_life

Mean-reversion half-life of a spread model

Description

Returns the time for a deviation from the long-run mean to halve in expectation, $\log 2/\kappa$, in the time units of the fit.

Usage

```
half_life(model)
```

Arguments

model A [spread_model](#).

Value

Numeric scalar.

See Also

Other classes: [spread_model\(\)](#)

Examples

```
x <- as.numeric(stats::arima.sim(list(ar = 0.8), n = 500))
half_life(ou_fit(x))
```

hmm

*A hidden Markov model specification***Description**

An S7 object specifying a discrete-state hidden Markov model with univariate Gaussian emissions, as built by the constructor of the same name. The hidden state follows a Markov chain with the given initial distribution and transition matrix; in state j the observation is Gaussian with mean `emission_mean[j]` and standard deviation `emission_sd[j]`.

Usage

```
hmm(
  init_prob = numeric(0),
  transition = numeric(0),
  emission_mean = numeric(0),
  emission_sd = numeric(0)
)
```

Arguments

<code>init_prob</code>	A length- k vector of initial state probabilities, summing to one.
<code>transition</code>	A k by k row-stochastic transition matrix.
<code>emission_mean</code>	A length- k vector of per-state emission means.
<code>emission_sd</code>	A length- k vector of per-state emission standard deviations, all strictly positive.

Value

An S7 object of class `hmm`.

See Also

[hmm_filter\(\)](#), [hmm_viterbi\(\)](#).

Other hidden-markov: [hmm_filter\(\)](#), [hmm_fit\(\)](#), [hmm_viterbi\(\)](#)

Examples

```
## A two-state mean-shift model: a low state and a high state.
model <- hmm(
  init_prob = c(0.5, 0.5),
  transition = matrix(c(0.95, 0.05, 0.05, 0.95), nrow = 2, byrow = TRUE),
  emission_mean = c(0, 3),
  emission_sd = c(1, 1)
)
model
```

hmm_filter

Hidden Markov model filtering and smoothing

Description

Runs the forward-backward recursion over an [hmm](#). The forward pass gives the filtered state posteriors $P(s_t = j \mid y_{1:t})$ and the model log-likelihood; the backward pass combines with it to give the smoothed posteriors $P(s_t = j \mid y_{1:n})$. The recursion is carried in the log domain and renormalised each step, so it is stable over long series.

Usage

```
hmm_filter(model, y)
```

Arguments

model	An hmm .
y	A numeric vector of observations.

Value

An [hmm_fit](#).

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.

See Also

[hmm_viterbi\(\)](#) for the most likely path.

Other hidden-markov: [hmm\(\)](#), [hmm_fit\(\)](#), [hmm_viterbi\(\)](#)

Examples

```
model <- hmm(
  init_prob = c(0.5, 0.5),
  transition = matrix(c(0.95, 0.05, 0.05, 0.95), nrow = 2, byrow = TRUE),
  emission_mean = c(0, 3), emission_sd = c(1, 1)
)
set.seed(1)
y <- c(rnorm(40, 0), rnorm(40, 3))
fit <- hmm_filter(model, y)
fit@log_lik
```

hmm_fit

A hidden Markov model fit

Description

An S7 object holding the output of `hmm_filter()`: the filtered and smoothed state posteriors, the smoothed most-likely state at each step, and the model log-likelihood.

Usage

```
hmm_fit(
  filtered = numeric(0),
  smoothed = numeric(0),
  state_path = integer(0),
  log_lik = numeric(0),
  model = hmm()
)
```

Arguments

<code>filtered</code>	An n by k matrix of filtered state posteriors.
<code>smoothed</code>	An n by k matrix of smoothed state posteriors.
<code>state_path</code>	An integer vector of the per-step smoothed modal state.
<code>log_lik</code>	Numeric scalar — the model log-likelihood.
<code>model</code>	The hmm that was filtered.

Value

An S7 object of class `hmm_fit`.

See Also

Other hidden-markov: [hmm\(\)](#), [hmm_filter\(\)](#), [hmm_viterbi\(\)](#)

hmm_viterbi

*Most likely hidden-state path (Viterbi)***Description**

Returns the single most likely sequence of hidden states given the whole observed series, by the Viterbi (1967) dynamic-programming recursion in the log domain. Unlike the per-step smoothed mode from `hmm_filter()`, the Viterbi path is jointly optimal: it is the state sequence that maximises the joint posterior, and so respects the transition structure across steps.

Usage

```
hmm_viterbi(model, y)
```

Arguments

<code>model</code>	An hmm .
<code>y</code>	A numeric vector of observations.

Value

An integer vector of length `length(y)`, the most likely state index at each step.

References

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), 260–269.

See Also

[hmm_filter\(\)](#).

Other hidden-markov: [hmm\(\)](#), [hmm_filter\(\)](#), [hmm_fit\(\)](#)

Examples

```
model <- hmm(
  init_prob = c(0.5, 0.5),
  transition = matrix(c(0.95, 0.05, 0.05, 0.95), nrow = 2, byrow = TRUE),
  emission_mean = c(0, 3), emission_sd = c(1, 1)
)
set.seed(1)
y <- c(rnorm(40, 0), rnorm(40, 3))
table(hmm_viterbi(model, y))
```

Description

An S7 object holding the model-adequacy diagnostics of `innovation_diagnostics()`: the standardised one-step-ahead innovations and the three residual tests read off them – whiteness, scale calibration and normality – together with the combined adequacy verdict.

Usage

```
innov_diag(
  standardised = numeric(),
  whiteness_p = NA_real_,
  calibration_p = NA_real_,
  normality_p = NA_real_,
  p_value = NA_real_,
  adequate = NA,
  alpha = 0.05,
  lags = NA_integer_,
  n_obs = NA_integer_,
  obs_dim = NA_integer_,
  obs_family = "gaussian",
  reason = NA_character_
)
```

Arguments

standardised	An n by d matrix of standardised innovations.
whiteness_p	Numeric scalar – the Ljung-Box whiteness p-value, Bonferroni-combined across the observation dimensions.
calibration_p	Numeric scalar – the two-sided p-value of the normalised-innovation-squared scale test.
normality_p	Numeric scalar – the Jarque-Bera normality p-value of the pooled standardised innovations.
p_value	Numeric scalar – the combined adequacy p-value, the Bonferroni minimum across the three test families.
adequate	Logical scalar – TRUE when no test rejects at alpha, FALSE when the model is rejected, NA when there is too little data to judge.
alpha	Numeric scalar – the per-battery significance level.
lags	Integer scalar – the Ljung-Box lag.
n_obs	Integer scalar – the series length.
obs_dim	Integer scalar – the observation dimension.

obs_family	Character scalar – the observation-noise family the diagnostics were taken under ("gaussian" or "student_t"), which selects the reference distribution of the scale and family-fit tests.
reason	Character scalar – a short label for the verdict ("adequate", "model_inadequate" or "insufficient_data").

Value

An S7 object of class `innov_diag`.

See Also

Other causal: `aci()`, `aci_fit()`, `causal_information_rate()`, `estimate_obs_df()`, `innovation_diagnostics()`, `its_causal()`, `its_fit()`

innovation_diagnostics

Model-adequacy diagnostics for a fitted state-space model

Description

Runs the standard innovation diagnostics of linear-Gaussian state-space model checking over a completed Kalman pass and reduces them to a single adequacy verdict. Under a correctly specified model the standardised one-step-ahead innovations are independent standard Gaussian, so three departures from that are tested: serial correlation (a Ljung-Box whiteness test on each observation dimension), a mis-scaled noise level (a two-sided test of the total normalised innovation squared against its χ^2 reference) and non-Gaussianity (a Jarque-Bera test on the pooled innovations). The model is declared inadequate when any family rejects at alpha, combined conservatively across the families by the Bonferroni rule.

Usage

```
innovation_diagnostics(object, lags = NULL, alpha = 0.05)
```

Arguments

object	A <code>kalman_fit</code> , or a <code>regime_fit</code> from <code>mixture_filter()</code> / <code>kalmix_filter()</code> , whose collapsed one-step innovations are tested the same way (the collapsed mixture predictive is treated as Gaussian, the same collapse the GPB1 recursion itself makes). Innovations at missing observations are skipped; the tests run on the complete steps.
lags	Integer scalar – the Ljung-Box lag. Defaults to <code>NULL</code> , which uses one fifth of the series length, capped between one and ten.
alpha	Numeric scalar – the per-battery significance level the adequacy verdict is taken at. Defaults to <code>0.05</code> .

Details

The diagnostics are what `aci()` uses to decide whether to trust its causal read-out: a model whose innovations reject it is not a credible data-generating mechanism, so the read-out is labelled "ungrounded" and the decision abstains. Passing the diagnostics establishes self-consistency only, which is necessary but not sufficient for a grounded causal claim (see `aci()` on declaring mechanism provenance).

Value

An `innov_diag`.

References

- Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.
- Durbin, J. and Koopman, S. J. (2012). *Time Series Analysis by State Space Methods*. 2nd ed. Oxford University Press.
- Jarque, C. M. and Bera, A. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3), 255–259.

See Also

`aci()`, which calls this to ground its causal verdict.

Other causal: `aci()`, `aci_fit()`, `causal_information_rate()`, `estimate_obs_df()`, `innov_diag()`, `its_causal()`, `its_fit()`

Examples

```
## A correctly specified local level: the innovations pass the diagnostics.
model <- ssm(
  transition = 1, observation = 1,
  state_cov = 0.04, obs_cov = 1,
  init_state = 0, init_cov = 10
)
set.seed(1)
level <- cumsum(rnorm(300, sd = 0.2))
y <- level + rnorm(300)
innovation_diagnostics(kalman_filter(model, y))
```

Description

Estimates the causal effect of an intervention on a single time series by contrasting the observed post-intervention path with the counterfactual a state-space model forecasts from the pre-intervention segment. A local-level or local-linear-trend [ssm](#) is fitted to $x[1:(intervention - 1)]$, its state is projected forward across the post-intervention window, and the observed values are differenced against that forecast. The forecast covariance gives a predictive interval, so the pointwise, cumulative and average effects each carry a calibrated uncertainty band, and a genuinely null effect shows an interval that straddles zero.

Usage

```
its_causal(x, intervention, trend = FALSE, level = 0.95)
```

Arguments

<code>x</code>	A numeric vector — the full series, pre- and post-intervention.
<code>intervention</code>	Integer scalar — the index of the first post-intervention observation. The pre-period is $x[1:(intervention - 1)]$.
<code>trend</code>	Logical scalar — whether to use a local-linear-trend model (TRUE) rather than a local level (FALSE). A trend lets the counterfactual continue a pre-period slope. Defaults to FALSE.
<code>level</code>	Numeric scalar in $(0, 1)$ — the predictive-interval coverage. Defaults to 0.95.

Details

The design is the single-case (N-of-1) interrupted time series; it is causal only under the maintained assumption that, absent the intervention, the pre-period dynamics would have continued. That assumption is the user's to defend; the function quantifies the contrast it implies.

Value

An [its_fit](#).

References

Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N. and Scott, S. L. (2015). Inferring causal impact using Bayesian structural time-series models. *Annals of Applied Statistics*, 9(1), 247–274.

See Also

[ssm\(\)](#), [kalman_filter\(\)](#).

Other causal: [aci\(\)](#), [aci_fit\(\)](#), [causal_information_rate\(\)](#), [estimate_obs_df\(\)](#), [innov_diag\(\)](#), [innovation_diagnostics\(\)](#), [its_fit\(\)](#)

Examples

```
## A series with a genuine post-intervention lift of about +3.
set.seed(1)
pre <- cumsum(rnorm(60, sd = 0.2))
post <- pre[60] + cumsum(rnorm(40, sd = 0.2)) + 3
x <- c(pre, post)
fit <- its_causal(x, intervention = 61L)
fit@avg_effect
```

its_fit

A single-case interrupted-time-series causal contrast

Description

An S7 object holding the output of `its_causal()`: the observed post-intervention series, the model's counterfactual forecast and its predictive interval, the pointwise and cumulative effects, and the average post-intervention effect with an interval.

Usage

```
its_fit(
  time = numeric(0),
  observed = numeric(0),
  counterfactual = numeric(0),
  counterfactual_lower = numeric(0),
  counterfactual_upper = numeric(0),
  pointwise_effect = numeric(0),
  cumulative_effect = numeric(0),
  avg_effect = numeric(0),
  avg_effect_lower = numeric(0),
  avg_effect_upper = numeric(0),
  level = numeric(0)
)
```

Arguments

<code>time</code>	A numeric or integer vector of the post-intervention time indices.
<code>observed</code>	A numeric vector of the observed post-intervention values.
<code>counterfactual</code>	A numeric vector of the forecast counterfactual.
<code>counterfactual_lower</code> , <code>counterfactual_upper</code>	Numeric vectors — the predictive-interval bounds of the counterfactual.
<code>pointwise_effect</code>	A numeric vector — observed minus counterfactual.
<code>cumulative_effect</code>	A numeric vector — the running sum of the pointwise effect.
<code>avg_effect</code>	Numeric scalar — the average post-intervention effect.

avg_effect_lower, avg_effect_upper
 Numeric scalars — the interval on the average effect.

level
 Numeric scalar — the interval coverage used.

Value

An S7 object of class `its_fit`.

See Also

Other causal: [aci\(\)](#), [aci_fit\(\)](#), [causal_information_rate\(\)](#), [estimate_obs_df\(\)](#), [innov_diag\(\)](#), [innovation_diagnostics\(\)](#), [its_causal\(\)](#)

kalman_filter	<i>Kalman filter for a linear-Gaussian state-space model</i>
---------------	--

Description

Runs the closed-form forward recursion of Kalman (1960) over an [ssm](#). Starting from the model's prior, each step predicts the state through the transition and then updates it with the observation through the Kalman gain

$$K_t = P_{t|t-1} B_t^\top S_t^{-1}, \quad S_t = B_t P_{t|t-1} B_t^\top + R_t,$$

where S_t is the innovation covariance. The model log-likelihood is accumulated from the prediction error decomposition,

$$\log p(y_{1:n}) = -\frac{1}{2} \sum_t (d \log 2\pi + \log |S_t| + e_t^\top S_t^{-1} e_t),$$

with innovation $e_t = y_t - B_t x_{t|t-1}$.

Usage

```
kalman_filter(model, y)
```

Arguments

model An [ssm](#).

y A numeric vector (univariate) or n by d matrix (multivariate) of observations. A fully NA entry (row) marks a missing observation: the filter runs a prediction-only step there, records an NA innovation and adds nothing to the log-likelihood. Partially missing multivariate rows are refused.

Details

The recursion is exact for a linear-Gaussian model. Time-varying system matrices are honoured: a model whose transition, observation, state_cov or obs_cov was supplied as a list of n matrices uses the step-t matrix at step t.

Value

A [kalman_fit](#).

Assimilative causal inference

The filtering distribution $N(x_{t|t}, P_{t|t})$ returned here is one of the two distributions [aci\(\)](#) compares step by step: the time-resolved causal information is its relative entropy from the smoothing distribution of [rts_smoother\(\)](#) (Andreou, Chen and Bollt 2026, eq. 7). See [aci\(\)](#) and [causal_information_rate\(\)](#).

References

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45.

See Also

[rts_smoother\(\)](#) for the backward smoothing pass; [ssm\(\)](#) for the model.

Other state-space: [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

Examples

```
## Recover a smooth signal from a noisy local-level series.
model <- ssm(
  transition = 1, observation = 1,
  state_cov = 0.01, obs_cov = 1,
  init_state = 0, init_cov = 10
)
set.seed(1)
level <- cumsum(rnorm(100, sd = 0.1))
y <- level + rnorm(100)
fit <- kalman_filter(model, y)
fit@log_lik
```

kalman_fit

A Kalman filter pass

Description

An S7 object holding the output of a forward Kalman pass over an [ssm](#), as returned by [kalman_filter\(\)](#). The filtered quantities condition on observations up to and including each step; the predicted quantities condition on observations strictly before it. The innovations and their covariances are retained because the log-likelihood and the interrupted-time-series contrast are read off them.

Usage

```

kalman_fit(
  filtered_mean = numeric(0),
  filtered_cov = list(),
  predicted_mean = numeric(0),
  predicted_cov = list(),
  innovation = numeric(0),
  innovation_cov = list(),
  log_lik = numeric(0),
  model = ssm()
)

```

Arguments

filtered_mean An n by m matrix of filtered state means, row t being $E[x_t | y_{1:t}]$.

filtered_cov A list of n filtered state covariances.

predicted_mean An n by m matrix of one-step-ahead predicted state means, row t being $E[x_t | y_{1:t-1}]$.

predicted_cov A list of n one-step-ahead predicted state covariances.

innovation An n by d matrix of innovations (prediction errors).

innovation_cov A list of n innovation covariances.

log_lik Numeric scalar — the model log-likelihood from the prediction error decomposition.

model The [ssm](#) that was filtered.

Value

An S7 object of class `kalman_fit`.

See Also

Other state-space: [kalman_filter\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

kalmix_filter

Filter a regime-switching spread state (mixture Kalman)

Description

Filters the latent mean-reverting state of a spread through a finite Gaussian-mixture Kalman recursion, the regime-switching generalisation of the scalar filter of Elliott, van der Hoek and Malcolm (2005). The fitted [spread_model](#) is expanded into `n_regimes` linear-Gaussian regimes that share the spread's mean-reversion dynamics but differ in observation noise, from calm to turbulent; the observed spread is filtered through [mixture_filter\(\)](#), and the returned [regime_fit](#) carries both the filtered spread state and the posterior probability of each volatility regime at every step.

Usage

```
kalmix_filter(model, y, n_regimes = 2L, vol_multipliers = NULL)
```

Arguments

model	A spread_model giving the mean-reversion dynamics of the spread, as returned by ou_fit() .
y	A numeric vector — the observed spread series to filter.
n_regimes	Integer scalar — the number of volatility regimes to track. Defaults to 2.
vol_multipliers	A numeric vector of length n_regimes giving each regime's observation-noise multiple of the fitted spread variance, or NULL for a geometric spread from calm to turbulent. Defaults to NULL.

Details

The recursion runs on the package's native engine and needs no sibling package.

The regime models operate on deviations from the fitted long-run mean: the discrete-time Ornstein-Uhlenbeck dynamics are the plain autoregressive contraction of $x_t - \theta$, so the observed spread is de-measured by `model@theta` before filtering and the returned state estimates are reported back on the observed scale. A spread whose long-run mean is far from zero is therefore tracked around θ , not pulled toward zero.

Value

A [regime_fit](#) over the spread's volatility regimes. When `n_regimes = 1` the regime machinery is skipped and the [kalman_fit](#) of a plain Kalman pass is returned instead.

References

Elliott, R. J., van der Hoek, J. and Malcolm, W. P. (2005). Pairs trading. *Quantitative Finance*, 5(3), 271–276.

See Also

[mixture_filter\(\)](#) for the general engine; [ou_fit\(\)](#) for the model.

Examples

```
x <- as.numeric(stats::arima.sim(list(ar = 0.85), n = 400))
model <- ou_fit(x)
fit <- kalmix_filter(model, x)
tail(round(fit@regime_prob, 2))
```

mixture_filter

*Regime-switching Gaussian-mixture filter***Description**

Filters a series generated by a finite set of linear-Gaussian regimes with a Markov chain switching between them. Each regime is supplied as its own [ssm](#); the regimes must share a common state and observation dimension. At each step the filter advances every regime by one Kalman predict-and-update from the shared mixed prior, weights the regimes by their one-step evidence and the regime-transition prior, and collapses the per-regime posteriors back to a single Gaussian (a Gaussian pseudo-Bayesian order-one recursion). The returned regime probabilities are the filtered posterior over the active regime.

Usage

```
mixture_filter(models, y, transition = NULL, init_prob = NULL)
```

Arguments

models	A list of two or more ssm objects, one per regime, sharing a common state and observation dimension.
y	A numeric vector or n by d matrix of observations. A fully NA entry (row) is a missing observation: every regime runs a prediction-only step, the regime weights carry the chain prior forward unchanged and the step adds nothing to the log-likelihood. Partially missing multivariate rows are refused.
transition	A k by k regime-transition probability matrix (rows sum to one), or NULL for a default that stays in the current regime with high probability. Defaults to NULL.
init_prob	A length-k vector of initial regime probabilities, or NULL for a uniform prior. Defaults to NULL.

Value

A [regime_fit](#).

References

Bar-Shalom, Y., Li, X. R. and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. Wiley. (GPB and IMM filters.)

See Also

[kalman_filter\(\)](#) for the single-regime base case; [kalmix_filter\(\)](#) for the pairs-trading wrapper.

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

Examples

```
## Two volatility regimes for a local-level series.
calm <- ssm(
  transition = 1, observation = 1, state_cov = 0.01, obs_cov = 0.25,
  init_state = 0, init_cov = 1
)
wild <- ssm(
  transition = 1, observation = 1, state_cov = 0.01, obs_cov = 4,
  init_state = 0, init_cov = 1
)
set.seed(1)
y <- c(rnorm(60, sd = 0.5), rnorm(60, sd = 2))
fit <- mixture_filter(list(calm, wild), y)
tail(round(fit@regime_prob, 2))
```

mixture_smoother	<i>Regime-switching (Kim) smoother</i>
------------------	--

Description

Smooths a regime-switching series: runs the Gaussian pseudo-Bayesian (GPB1) forward pass of [mixture_filter\(\)](#) and the backward recursion of Kim (1994), returning the regime probabilities and collapsed state moments conditioned on the whole series. Smoothed regime probabilities use the future as well as the past, so a regime that only becomes apparent in hindsight is attributed to the steps where it held.

Usage

```
mixture_smoother(models, y, transition = NULL, init_prob = NULL)
```

Arguments

models	A list of two or more ssm objects, one per regime, sharing a common state and observation dimension.
y	A numeric vector or n by d matrix of observations (fully NA rows allowed).
transition	A k by k regime-transition probability matrix, or NULL for the sticky default of mixture_filter() . Defaults to NULL.
init_prob	A length-k vector of initial regime probabilities, or NULL for uniform. Defaults to NULL.

Details

Each backward step collapses a Gaussian mixture to one Gaussian per regime under the smoothed pair weights, mirroring the forward collapse, so the smoother is the standard approximation for Markov-switching state-space models rather than the (exponentially costly) exact mixture smoother. Missing observations are handled by the forward pass as prediction-only steps and flow through the backward pass unchanged.

Value

A [regime_smooth](#).

References

Kim, C.-J. (1994). Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60(1–2), 1–22.

See Also

[mixture_filter\(\)](#) for the forward pass; [rts_smoother\(\)](#) for the single-regime smoother.

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

Examples

```
calm <- ssm(
  transition = 1, observation = 1, state_cov = 0.01, obs_cov = 0.25,
  init_state = 0, init_cov = 1
)
wild <- ssm(
  transition = 1, observation = 1, state_cov = 0.01, obs_cov = 4,
  init_state = 0, init_cov = 1
)
set.seed(1)
y <- c(rnorm(60, sd = 0.5), rnorm(60, sd = 2))
sm <- mixture_smoother(list(calm, wild), y)
tail(round(sm@smoothed_prob, 2))
```

ou_fit

Fit Ornstein-Uhlenbeck mean reversion to a spread

Description

Estimates the mean-reverting (Ornstein-Uhlenbeck) dynamics of a spread by fitting the exact discrete-time autoregression

$$X_t = a + b X_{t-1} + \varepsilon_t,$$

then mapping the autoregressive coefficients back to the continuous-time parameters: $\kappa = -\log(b)/\Delta t$, $\theta = a/(1 - b)$, and $\sigma^2 = 2\kappa \text{Var}(\varepsilon)/(1 - b^2)$.

Usage

```
ou_fit(x, dt = 1)
```

Arguments

x Numeric vector — the spread series (for example from [spread_series\(\)](#)).

dt Numeric scalar — the sampling interval, in the time units the reported kappa should use. Defaults to 1.

Details

A spread that is not mean-reverting has an autoregressive coefficient outside $(0, 1)$; the function refuses such a series rather than report a meaningless negative half-life.

Value

A [spread_model](#).

See Also

Other spread: [spread_series\(\)](#)

Examples

```
x <- as.numeric(stats::arima.sim(list(ar = 0.85), n = 800))
ou_fit(x)
```

regime_fit

A regime-switching mixture filter pass

Description

An S7 object holding the output of [mixture_filter\(\)](#). The regime probabilities give the filtered posterior over which regime is active at each step; the collapsed state mean and covariance are the mixture-marginalised state estimate.

Usage

```
regime_fit(
  regime_prob = numeric(0),
  state_mean = numeric(0),
  state_cov = list(),
  innovation = numeric(0),
  innovation_cov = list(),
  innovation_pit = numeric(0),
  log_lik = numeric(0),
  n_regimes = integer(0)
)
```

Arguments

regime_prob	An n by k matrix of filtered regime probabilities, row t summing to one over the k regimes.
state_mean	An n by m matrix of collapsed (mixture-marginalised) state means.
state_cov	A list of n collapsed state covariances.

innovation	An n by d matrix of collapsed one-step innovations: the observation minus the mixture predictive mean under the chain-prior regime weights (NA at a missing observation).
innovation_cov	A list of n collapsed one-step predictive covariances matching innovation.
innovation_pit	For a univariate series, the length- n probability-integral transform of each observation under its one-step mixture predictive (NA at a missing observation; empty for $d > 1$). This is the exact reference for the adequacy diagnostics, where the collapsed Gaussian would wrongly fail a well-specified mixture.
log_lik	Numeric scalar — the mixture model log-likelihood.
n_regimes	Integer scalar — the number of regimes k .

Value

An S7 object of class `regime_fit`.

See Also

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

regime_smooth	<i>A regime-switching smoother pass</i>
---------------	---

Description

An S7 object holding the output of [mixture_smoother\(\)](#): the smoothed regime probabilities and the collapsed smoothed state moments, each conditioning on the whole observed series.

Usage

```
regime_smooth(
  smoothed_prob = numeric(0),
  smoothed_mean = numeric(0),
  smoothed_cov = list(),
  log_lik = numeric(0),
  n_regimes = integer(0)
)
```

Arguments

smoothed_prob	An n by k matrix of smoothed regime probabilities, row t summing to one.
smoothed_mean	An n by m matrix of collapsed smoothed state means.
smoothed_cov	A list of n collapsed smoothed state covariances.
log_lik	Numeric scalar — the mixture model log-likelihood, carried through from the forward pass.
n_regimes	Integer scalar — the number of regimes k .

Value

An S7 object of class `regime_smooth`.

See Also

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

<code>rts_fit</code>	<i>A Rauch-Tung-Striebel smoother pass</i>
----------------------	--

Description

An S7 object holding the output of a backward RTS smoothing pass, as returned by [rts_smoother\(\)](#). Each smoothed quantity conditions on the whole observed series, $E[x_t \mid y_{1:n}]$ and its covariance.

Usage

```
rts_fit(
  smoothed_mean = numeric(0),
  smoothed_cov = list(),
  log_lik = numeric(0),
  model = ssm()
)
```

Arguments

<code>smoothed_mean</code>	An n by m matrix of smoothed state means.
<code>smoothed_cov</code>	A list of n smoothed state covariances.
<code>log_lik</code>	Numeric scalar — the model log-likelihood (carried through from the filter pass for convenience).
<code>model</code>	The ssm that was smoothed.

Value

An S7 object of class `rts_fit`.

See Also

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

rts_smoother	<i>Rauch-Tung-Striebel smoother for a linear-Gaussian state-space model</i>
--------------	---

Description

Runs the backward recursion of Rauch, Tung and Striebel (1965) over a completed Kalman pass to obtain the state estimates conditioned on the whole series, $E[x_t \mid y_{1:n}]$. From the last filtered state backwards, each step blends the filtered estimate with the smoothed estimate of the next state through the smoother gain

$$C_t = P_{t|t} A_{t+1}^\top P_{t+1|t}^{-1},$$

giving $x_{t|n} = x_{t|t} + C_t(x_{t+1|n} - x_{t+1|t})$ and the matching covariance update. The result is exact for a linear-Gaussian model.

Usage

```
rts_smoother(object, y = NULL)
```

Arguments

object	A kalman_fit from kalman_filter() , or an ssm .
y	A numeric vector or matrix of observations. Required when object is an ssm ; ignored when object is already a kalman_fit .

Details

Either a fitted [kalman_fit](#) or an [ssm](#) plus observations may be supplied; in the latter case the filter pass is run first.

Value

An [rts_fit](#).

Assimilative causal inference

This smoother supplies the second of the two Gaussian distributions [aci\(\)](#) compares: the time-resolved causal information is the relative entropy of the smoothing distribution $N(x_{t|n}, P_{t|n})$ from the filtering distribution (Andreou, Chen and Boltt 2026, eq. 7). See [aci\(\)](#).

References

Rauch, H. E., Tung, F. and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8), 1445–1450.

See Also

[kalman_filter\(\)](#).

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

Examples

```
model <- ssm(
  transition = 1, observation = 1,
  state_cov = 0.01, obs_cov = 1,
  init_state = 0, init_cov = 10
)
set.seed(1)
y <- cumsum(rnorm(100, sd = 0.1)) + rnorm(100)
smoothed <- rts_smoother(model, y)
head(smoothed@smoothed_mean)
```

spread_model

A fitted mean-reverting spread model

Description

An S7 object holding the Ornstein-Uhlenbeck (OU) parameters of a cointegration spread, as returned by [ou_fit\(\)](#). The continuous-time model is

$$dX_t = \kappa(\theta - X_t) dt + \sigma dW_t,$$

so kappa is the speed of mean reversion, theta the long-run mean, and sigma the instantaneous volatility. The mean-reversion half-life is $\log 2 / \kappa$.

Usage

```
spread_model(
  kappa = numeric(0),
  theta = numeric(0),
  sigma = numeric(0),
  dt = 1,
  n_obs = NA_integer_,
  hedge_ratio = NA_real_,
  call = NULL
)
```

Arguments

kappa	Numeric scalar — the mean-reversion speed (per unit time). Strictly positive for a mean-reverting spread.
theta	Numeric scalar — the long-run mean the spread reverts to.

sigma	Numeric scalar — the instantaneous volatility. Strictly positive.
dt	Numeric scalar — the sampling interval the model was fitted at, in the same time units as kappa.
n_obs	Integer scalar — the number of spread observations used.
hedge_ratio	Numeric scalar — the cointegration hedge ratio the spread was built with, or NA_real_ if the spread was supplied directly.
call	The matched call to <code>ou_fit()</code> .

Value

An S7 object of class `spread_model`.

See Also

Other classes: `half_life()`

Examples

```
x <- as.numeric(stats::arima.sim(list(ar = 0.8), n = 500))
ou_fit(x)
```

spread_series	<i>Build a cointegration spread from two price series</i>
---------------	---

Description

Forms the spread $a - \alpha - \beta * b$ whose stationarity underlies a pairs trade, where β is the ordinary-least-squares hedge ratio of a on b and α the intercept. The hedge ratio and intercept are attached to the result as attributes.

Usage

```
spread_series(a, b, intercept = TRUE)
```

Arguments

a	Numeric vector — the first asset's price (or log-price) series.
b	Numeric vector — the second asset's series, the same length as a.
intercept	Logical scalar — whether to include an intercept in the hedge regression. Defaults to TRUE.

Value

A numeric vector of the same length as a , the spread, carrying `hedge_ratio` and `intercept` attributes.

See Also

Other spread: [ou_fit\(\)](#)

Examples

```
set.seed(1)
b <- cumsum(stats::rnorm(300))
a <- 2 * b + stats::rnorm(300)
s <- spread_series(a, b)
attr(s, "hedge_ratio")
```

ssm

A linear-Gaussian state-space model

Description

An S7 object specifying a discrete-time linear-Gaussian state-space model, built by the constructor of the same name. The latent state of dimension `m` evolves through a transition matrix, an optional intercept (control input) and Gaussian process noise, and is observed through an observation matrix, an optional observation intercept and Gaussian observation noise:

$$x_t = A_t x_{t-1} + c_t + w_t, \quad w_t \sim N(0, Q_t),$$

$$y_t = B_t x_t + d_t + v_t, \quad v_t \sim N(0, R_t).$$

The intercepts default to zero, recovering the intercept-free form; a non-zero c_t is the standard way to give a stationary state a non-zero long-run mean (for an AR(1) state with coefficient ϕ and mean θ , set $c = \theta(1 - \phi)$).

Usage

```
ssm(
  transition,
  observation,
  state_cov,
  obs_cov,
  init_state,
  init_cov,
  state_intercept = 0,
  obs_intercept = 0,
  obs_family = c("gaussian", "student_t"),
  obs_df = Inf
)
```

Arguments

transition	The state transition matrix A (m by m), or a list of such matrices for a time-varying transition.
observation	The observation matrix B (d by m , where d is the observation dimension), or a list of such matrices.
state_cov	The process-noise covariance Q (m by m , symmetric positive semi-definite), or a list of such matrices.
obs_cov	The observation-noise covariance R (d by d , symmetric positive definite), or a list of such matrices.
init_state	The prior mean of the initial state x_0 , a numeric vector of length m .
init_cov	The prior covariance of the initial state, an m by m symmetric positive semi-definite matrix.
state_intercept	The state intercept c (a scalar or length- m vector, or a list of such for a time-varying control input). Defaults to $\emptyset$.
obs_intercept	The observation intercept d (a scalar or length- d vector, or a list of such). Defaults to $\emptyset$.
obs_family	The observation-noise family, "gaussian" (the default) or "student_t". A Student-t family gives a heavy-tailed observation model whose filter is robust to outliers; <code>obs_cov</code> is then the t <i>scale</i> matrix R rather than a covariance. See kalman_filter() for the recursion.
obs_df	The Student-t degrees of freedom ν (a finite scalar greater than two) when <code>obs_family = "student_t"</code> ; ignored, and held at Inf , for the Gaussian family. estimate_obs_df() selects it by profile likelihood. As $\nu \rightarrow \infty$ the Student-t model collapses to the Gaussian one.

Details

Each of `transition`, `observation`, `state_cov` and `obs_cov` is accepted as a single matrix (static, reused at every step) or as a list of matrices, one per time step (time-varying). Scalars and length- m vectors are promoted to matrices, so a univariate local-level model can be written with bare numbers. Internally every system matrix is stored as a list of one or more matrices; the filtering and smoothing recursions index it per step.

The dimensions must be mutually consistent: with state dimension m and observation dimension d , the transition matrices are m by m , the observation matrices d by m , the process covariances m by m , and the observation covariances d by d . The initial covariance is m by m .

Value

An S7 object of class `ssm`.

References

Durbin, J. and Koopman, S. J. (2012). *Time Series Analysis by State Space Methods*, 2nd ed. Oxford University Press.

See Also

[kalman_filter\(\)](#), [rts_smoother\(\)](#), [mixture_filter\(\)](#).

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm_fit\(\)](#), [ssm_mle\(\)](#)

Examples

```
## A local-level (random-walk-plus-noise) model written with bare numbers.
```

```
model <- ssm(
  transition = 1,
  observation = 1,
  state_cov = 0.1,
  obs_cov = 1,
  init_state = 0,
  init_cov = 10
)
model
```

```
## A constant-velocity tracking model with a vector state.
```

```
velocity <- ssm(
  transition = matrix(c(1, 0, 1, 1), nrow = 2),
  observation = matrix(c(1, 0), nrow = 1),
  state_cov = diag(c(0.01, 0.01)),
  obs_cov = 1,
  init_state = c(0, 0),
  init_cov = diag(c(10, 10))
)
velocity
```

ssm_fit

Fit ssm noise parameters by maximum likelihood

Description

Estimates the process covariance, the observation covariance and (optionally) the transition matrix of an [ssm](#) by maximising the prediction-error log-likelihood of [kalman_filter\(\)](#) with `stats::optim()`.

The supplied model is the template: its non-estimated components, its prior and its observation family are kept as given, and its current values seed the first optimisation start.

Usage

```
ssm_fit(
  model,
  y,
  estimate = c("state_cov", "obs_cov"),
  transition = FALSE,
  n_start = 3L,
  control = list()
)
```

Arguments

model	An ssm — the template supplying structure, prior, family and starting values. Estimated components must be time-invariant.
y	A numeric vector or n by d matrix of observations (NA rows allowed).
estimate	A character vector naming the components to estimate, from "state_cov" and "obs_cov". Defaults to both.
transition	Logical scalar — whether to estimate the transition matrix as well (unconstrained; a fit whose spectral radius reaches one is reported with a warning). Defaults to FALSE.
n_start	Integer scalar — the number of optimisation starts. Defaults to 3.
control	A list passed to <code>stats::optim()</code> 's control argument.

Details

Covariances are optimised through their log-Cholesky factors (the logarithm of the factor's diagonal, the off-diagonal entries free), so the search is unconstrained and every iterate is a valid covariance. The fit is repeated from `n_start` starts (the template plus a deterministic ladder of log-scale shifts, so no seed is involved) and the best optimum is kept, guarding against local optima at a modest cost. Standard errors are read from the numerical Hessian at the optimum and mapped to the covariance entries by a numerical delta method; they are asymptotic and can be optimistic on short series, where the likelihood separates the two noise sources weakly (a wide interval there is a finding about the data, not a defect of the fit).

Missing observations (fully NA rows) are handled by the filter as prediction-only steps, so a gappy series can be fitted directly. Under a Student-t observation family the same machinery maximises the robust filter's likelihood at the template's fixed degrees of freedom; combine with `estimate_obs_df()` to choose the degrees of freedom.

Value

An [ssm_mle](#).

References

- Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press. (Prediction-error decomposition maximum likelihood.)
- Durbin, J. and Koopman, S. J. (2012). *Time Series Analysis by State Space Methods*. 2nd ed. Oxford University Press.

See Also

[kalman_filter\(\)](#) for the likelihood; [estimate_obs_df\(\)](#) for the Student-t degrees of freedom; [ssm\(\)](#) for the template.

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_mle\(\)](#)

Examples

```
## Recover the noise split of a local-level series. One start keeps the
## example fast; the default multi-start is the safer everyday setting.
set.seed(1)
level <- cumsum(rnorm(150, sd = 0.5))
y <- level + rnorm(150)
template <- ssm(
  transition = 1, observation = 1, state_cov = 1, obs_cov = 1,
  init_state = y[1], init_cov = 10
)
fit <- ssm_fit(template, y, n_start = 1L)
fit@estimates$state_cov
fit@estimates$obs_cov
```

ssm_mle

A maximum-likelihood ssm fit

Description

An S7 object holding the output of `ssm_fit()`: the fitted `ssm` (ready for every verb that accepts one), the maximised log-likelihood, the estimates and standard errors of the estimated entries, and the optimiser diagnostics.

Usage

```
ssm_mle(
  model = ssm(),
  log_lik = numeric(0),
  estimates = list(),
  std_errors = list(),
  convergence = integer(0),
  n_start = integer(0),
  start_log_lik = numeric(0)
)
```

Arguments

<code>model</code>	The fitted <code>ssm</code> .
<code>log_lik</code>	Numeric scalar — the maximised prediction-error log-likelihood.
<code>estimates</code>	A named list of the estimated matrices (<code>state_cov</code> , <code>obs_cov</code> and, when requested, <code>transition</code>).
<code>std_errors</code>	A named list matching estimates, each entry a matrix of delta-method standard errors for the corresponding estimates.
<code>convergence</code>	Integer scalar — the <code>stats::optim()</code> convergence code of the winning start (0 means converged).
<code>n_start</code>	Integer scalar — the number of optimisation starts run.
<code>start_log_lik</code>	A numeric vector with the best log-likelihood reached from each start.

Value

An S7 object of class `ssm_mle`.

See Also

Other state-space: [kalman_filter\(\)](#), [kalman_fit\(\)](#), [mixture_filter\(\)](#), [mixture_smoother\(\)](#), [regime_fit\(\)](#), [regime_smooth\(\)](#), [rts_fit\(\)](#), [rts_smoother\(\)](#), [ssm\(\)](#), [ssm_fit\(\)](#)

Index

- * **causal**
 - aci, 3
 - aci_fit, 6
 - causal_information_rate, 7
 - estimate_obs_df, 13
 - innov_diag, 19
 - innovation_diagnostics, 20
 - its_causal, 21
 - its_fit, 23
 - * **change-point**
 - changepoint_fit, 10
 - detect_changepoint, 11
 - * **classes**
 - half_life, 14
 - spread_model, 35
 - * **filter**
 - kalmix_filter, 26
 - * **hidden-markov**
 - hmm, 15
 - hmm_filter, 16
 - hmm_fit, 17
 - hmm_viterbi, 18
 - * **spread**
 - ou_fit, 30
 - spread_series, 36
 - * **state-space**
 - kalman_filter, 24
 - kalman_fit, 25
 - mixture_filter, 28
 - mixture_smoother, 29
 - regime_fit, 31
 - regime_smooth, 32
 - rts_fit, 33
 - rts_smoother, 34
 - ssm, 37
 - ssm_fit, 39
 - ssm_mle, 41
- aci, 3
aci(), 6, 7, 10, 14, 20–22, 24, 25, 34
- aci_fit, 4, 5, 6
aci_fit(), 5, 10, 14, 20–22, 24
- causal_information_rate, 7
causal_information_rate(), 4, 5, 7, 14, 20–22, 24, 25
- changepoint_fit, 10, 12
changepoint_fit(), 12
- detect_changepoint, 11
detect_changepoint(), 10, 11
- estimate_obs_df, 13
estimate_obs_df(), 5, 7, 10, 20–22, 24, 38, 40
- half_life, 14
half_life(), 36
- hmm, 15, 16–18
hmm(), 16–18
hmm_filter, 16
hmm_filter(), 15, 17, 18
hmm_fit, 16, 17
hmm_fit(), 15, 16, 18
hmm_viterbi, 18
hmm_viterbi(), 15–17
- innov_diag, 7, 19, 21
innov_diag(), 5, 7, 10, 14, 21, 22, 24
- innovation_diagnostics, 20
innovation_diagnostics(), 4, 5, 7, 10, 14, 19, 20, 22, 24
- its_causal, 21
its_causal(), 5, 7, 10, 14, 20, 21, 23, 24
- its_fit, 22, 23
its_fit(), 5, 7, 10, 14, 20–22
- kalman_filter, 24
kalman_filter(), 3, 5, 13, 22, 25, 26, 28, 30, 32–35, 38–40, 42
- kalman_fit, 6, 20, 25, 25, 27, 34

`kalman_fit()`, 25, 28, 30, 32, 33, 35, 39, 40, 42
`kalmix_filter`, 26
`kalmix_filter()`, 20, 28

`mixture_filter`, 28
`mixture_filter()`, 4, 9, 20, 25–27, 29–33, 35, 39, 40, 42
`mixture_smoother`, 29
`mixture_smoother()`, 9, 25, 26, 28, 32, 33, 35, 39, 40, 42

`ou_fit`, 30
`ou_fit()`, 27, 35–37

`regime_fit`, 6, 20, 26–28, 31
`regime_fit()`, 25, 26, 28, 30, 33, 35, 39, 40, 42
`regime_smooth`, 6, 30, 32
`regime_smooth()`, 25, 26, 28, 30, 32, 33, 35, 39, 40, 42
`rts_fit`, 6, 33, 34
`rts_fit()`, 25, 26, 28, 30, 32, 33, 35, 39, 40, 42
`rts_smoother`, 34
`rts_smoother()`, 3, 5, 25, 26, 28, 30, 32, 33, 39, 40, 42

`spread_model`, 14, 26, 27, 31, 35
`spread_model()`, 14
`spread_series`, 36
`spread_series()`, 30, 31
`ssm`, 3, 4, 8, 9, 13, 22, 24–26, 28, 29, 33, 34, 37, 39–41
`ssm()`, 14, 22, 25, 26, 28, 30, 32, 33, 35, 40, 42
`ssm_fit`, 39
`ssm_fit()`, 25, 26, 28, 30, 32, 33, 35, 39, 41, 42
`ssm_mle`, 40, 41
`ssm_mle()`, 25, 26, 28, 30, 32, 33, 35, 39, 40
`stats::optim()`, 39–41