

Package ‘DynCount’

September 28, 2026

Title Bayesian Dynamic Models for Count Time Series

Version 0.2.0

Description Fits Bayesian state-space models for count time series using a latent log-rate (Poisson), latent logit (binomial) or latent additive-log-ratio (multinomial choice counts) formulation. Each latent trajectory follows a first-order random walk or a stationary AR(1) process and is sampled by Metropolis-within-Gibbs using the implied Gaussian Markov random field full conditionals. The latent increments can be Gaussian, Student-t, a finite scale mixture of normals, or follow a stochastic volatility process, and the Poisson and binomial families support zero inflation. It implements and extends the methodology of Zens and Bijak (2026) <[doi:10.1214/26-AOAS2171](https://doi.org/10.1214/26-AOAS2171)>.

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

Depends R (>= 3.5.0)

Imports generics, stats, graphics, grDevices, utils

Suggests coda, stochvol (>= 3.0.2), testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

LazyData true

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation no

Author Gregor Zens [aut, cre]

Maintainer Gregor Zens <zens@iiasa.ac.at>

Repository CRAN

Date/Publication 2026-09-28 10:10:02 UTC

Contents

dynamic_prior	2
fit_dynamic_model	5
forecast.dynamic_fit	9
med_weekly	11
plot.dynamic_fit	12
plot_fitted	13
plot_forecast	13
plot_latent	14
plot_zero_inflation	15
predict.dynamic_fit	16
simulate_dynamic_binomial	17
simulate_dynamic_multinomial	18
simulate_dynamic_poisson	19
structural_zero_prob	21
summary.dynamic_fit	22
uk_weekly	23
Index	24

dynamic_prior	<i>Specify priors for a dynamic model</i>
---------------	---

Description

Builds the prior hyperparameters used by `fit_dynamic_model()`. Called with no arguments it returns weakly informative defaults.

Usage

```
dynamic_prior(  
  var_shape = 0.01,  
  var_rate = 0.01,  
  df_min = 3,  
  df_mean_excess = 6,  
  mix_components = 2,  
  mix_concentration = 1,  
  mix_var_shape = 2.5,  
  mix_var_rate = 0.5,  
  sv_prior = NULL,  
  zi_open_a = 1,  
  zi_open_b = 1,  
  ar_rho_mean = 0,  
  ar_rho_sd = 1,  
  mu_mean = 0,  
  mu_sd = 1,  
  init_mean = 0,
```

```

    init_var = 100
)

```

Arguments

var_shape	Shape of the inverse-gamma prior on the innovation variance/scale. Default 0.01.
var_rate	Rate of the inverse-gamma prior on the innovation variance/scale. Default 0.01.
df_min	Lower bound for the Student-t degrees of freedom. Default 3.
df_mean_excess	Prior mean of $\nu - \text{df\_min}$. Default 6.
mix_components	Number of components in the scale-mixture innovation structure. Default 2.
mix_concentration	Symmetric Dirichlet concentration for the mixture weights. Default 1.
mix_var_shape, mix_var_rate	Shape and rate of the inverse-gamma prior on the relative variance of each mixture component (used only when innovations = "mixture"). Defaults 2.5 and 0.5.
sv_prior	Optional stochvol prior specification for the stochastic-volatility innovation structure. Default NULL.
zi_open_a, zi_open_b	Beta prior parameters for the gate-open probability in zero-inflated models. Default 1 and 1.
ar_rho_mean, ar_rho_sd	Mean and standard deviation of the Gaussian prior on the AR(1) coefficient ρ (used only when latent_dynamics = "ar1"). The prior is truncated to the stationary region $\rho \in (-1, 1)$. Defaults 0 and 1.
mu_mean, mu_sd	Mean and standard deviation of the Gaussian prior on the drift/intercept μ (used only when include_mu = TRUE). Defaults 0 and 1.
init_mean, init_var	Mean and variance of the Gaussian prior on the initial latent state, used under both random-walk and AR(1) dynamics. Defaults 0 and 100 (a diffuse $N(0, 100)$).

Details

The model places a GMRF latent process on the series z_t (a log-rate for the Poisson family, a logit for the binomial family, and one additive-log-ratio series per non-baseline category for the multinomial family): either a first-order random walk (latent_dynamics = "rw") or an AR(1) process (latent_dynamics = "ar1"). The increments $\varepsilon_t = z_t - \mu - \rho z_{t-1}$ (with $\rho = 1$ for the random walk and $\mu = 0$ unless a drift/intercept is included) are given one of four innovation distributions, each governed by some of the priors below.

Innovation variance (all structures). The baseline increment variance σ^2 (or, for the "t"/"mixture" structures, the overall scale) has an inverse-gamma prior

$$\sigma^2 \sim \text{InvGamma}(\text{var\_shape}, \text{var\_rate}).$$

The default `InvGamma(0.01, 0.01)` is weakly informative for typical increment standard deviations, but it is not scale-free. For very smooth or short series, with increment standard deviations of a few hundredths, the results can be sensitive to this prior. In that regime, check the sensitivity of the results to a smaller `var_rate` (which lets σ become smaller) and expect slower mixing, because a nearly constant latent path and a small σ are strongly dependent a posteriori. In the multinomial family the same prior is applied independently to every non-baseline category.

Mixture component variances (`innovations = "mixture"`). The relative variances of the mixture components multiply the overall scale and are given their own `InvGamma(mix_var_shape, mix_var_rate)` prior (default `InvGamma(2.5, 0.5)`). They are kept moderately informative on purpose, as with a vague prior an empty component would be drawn from an extremely heavy-tailed distribution and the split between the overall scale and the component variances is only weakly identified.

Student-t degrees of freedom (`innovations = "t"`). The degrees of freedom are modelled as $\nu = \text{df\_min} + E$, where $E \sim \text{Exp}(\text{rate} = 1/\text{df\_mean\_excess})$. Thus $\nu \geq \text{df\_min}$ and its prior mean is $\text{df\_min} + \text{df\_mean\_excess}$. Large ν approaches the Gaussian case.

Scale mixture (`innovations = "mixture"`). The mixture uses `mix_components` variance components, each with an `InvGamma(mix_var_shape, mix_var_rate)` prior (see above), and symmetric Dirichlet weights with concentration `mix_concentration`.

Stochastic volatility (`innovations = "sv"`). The log-variance h_t of the increments follows $h_t = \mu_h + \phi(h_{t-1} - \mu_h) + \sigma_h \eta_t$, and its priors are delegated to **stochvol**. Pass a prior specification created with `stochvol::specify_priors()` via `sv_prior`, or leave it `NULL` to use the **stochvol** defaults.

Zero inflation (`zeros = "inflated"`). The probability that the observation "gate" is open (i.e. that a zero is an ordinary sampling zero rather than a structural zero) has a `Beta(zi_open_a, zi_open_b)` prior. The default `Beta(1, 1)` is uniform.

AR(1) coefficient (`latent_dynamics = "ar1"`). When the latent state follows $z_t = \mu + \rho z_{t-1} + \varepsilon_t$, the coefficient ρ is given a Gaussian prior $\rho \sim N(\text{ar\_rho\_mean}, \text{ar\_rho\_sd}^2)$, truncated to the stationary region $\rho \in (-1, 1)$. AR(1) always carries an intercept (`include_mu = TRUE`). Under `latent_dynamics = "rw"` the coefficient is fixed at $\rho = 1$ and this prior is unused.

Drift / intercept (`include_mu = TRUE`). A scalar μ in the state equation $z_t = \mu + \rho z_{t-1} + \varepsilon_t$: a *drift* under the random walk ($\rho = 1$) and an *intercept* under AR(1) (where it is always enabled). It is given a Gaussian prior $\mu \sim N(\text{mu\_mean}, \text{mu\_sd}^2)$. With `include_mu = FALSE` (random walk only), $\mu = 0$ and is not sampled.

Initial state. A proper, fixed `N(init_mean, init_var)` prior (default `N(0, 100)`) anchors the otherwise-improper GMRF on the first latent state, under both "rw" and "ar1" dynamics. With the diffuse default the initial state is effectively determined by the data.

Value

An object of class "dynamic_prior": a named list of hyperparameters.

See Also

`fit_dynamic_model()`

Examples

```
# Defaults
dynamic_prior()
```

```
# An informative variance prior and heavier-tailed t innovations
dynamic_prior(var_shape = 2.5, var_rate = 0.5, df_min = 2, df_mean_excess = 3)
```

fit_dynamic_model	<i>Fit a Bayesian dynamic count / binomial / multinomial time-series model</i>
-------------------	--

Description

Fits a GMRF state-space model in which a latent trajectory z_t evolves as either a first-order random walk (`latent_dynamics = "rw"`, the default) or a stationary AR(1) process (`latent_dynamics = "ar1"`), and the observations are linked to it through a Poisson (log link), binomial (logit link) or multinomial (additive-log-ratio link) observation model. See [DynCount-package](#) for an overview.

Usage

```
fit_dynamic_model(
  y,
  family = c("poisson", "binomial", "multinomial"),
  trials = NULL,
  innovations = c("gaussian", "t", "mixture", "sv"),
  latent_dynamics = c("rw", "ar1"),
  include_mu = FALSE,
  zeros = c("none", "inflated", "missing"),
  zero_inflation = FALSE,
  prior = dynamic_prior(),
  nsave = 4000,
  nburn = 1000,
  thin = 1,
  horizon = 0L,
  forecast_trials = NULL,
  forecast_offset = NULL,
  offset = NULL,
  baseline = "largest",
  verbose = FALSE,
  seed = NULL
)
```

Arguments

<code>y</code>	For the Poisson and binomial families a numeric vector of non-negative integer observations (counts, or numbers of successes). For the multinomial family an $n \times K$ matrix (or data frame) of non-negative integer category counts with one row per time point; its column names, if any, are used as category labels.
<code>family</code>	Observation model, "poisson" (default), "binomial" or "multinomial".

trials	For family = "binomial", the number of trials. Either a single number (recycled) or a vector the same length as y. Each y must not exceed its number of trials. For the multinomial family the row totals of y are used and this argument must be left NULL; for the Poisson family it is ignored with a warning.
innovations	Distribution of the latent increments, one of "gaussian" (default), "t", "mixture", "sv". See DynCount-package for details. "sv" requires the stochvol package.
latent_dynamics	Latent state evolution: "rw" (default; a first-order random walk, $\rho = 1$ fixed) or "ar1" (a stationary AR(1) with ρ sampled on $(-1, 1)$). "ar1" always includes an intercept (include_mu is forced to TRUE).
include_mu	Logical; include a scalar μ in the state equation $z_t = \mu + \rho z_{t-1} + \varepsilon_t$. It is a drift under latent_dynamics = "rw" ($\rho = 1$) and an intercept under "ar1". With FALSE (default) $\mu = 0$ and is not sampled; with TRUE it has the Gaussian prior in dynamic_prior() . When latent_dynamics = "ar1" this is forced to TRUE regardless of the value supplied: the intercept gives the process a non-zero stationary mean $\mu/(1 - \rho)$, without which the zero-mean stationary assumption is rarely appropriate for a log-rate/logit series.
zeros	Zero handling for the Poisson and binomial families: "none" (default), "inflated" (zero inflation with a time-constant gate-open probability; see structural_zero_prob()), or "missing" (observed zeros treated as missing data). Must be "none" for the multinomial family. See Details.
zero_inflation	A single TRUE or FALSE. TRUE is shorthand for zeros = "inflated"; combining it with a different explicit zeros is an error. Unlike the zero_inflation argument of simulate_dynamic_poisson() and simulate_dynamic_binomial() , it is not a probability.
prior	A dynamic_prior() object giving the prior hyperparameters. For the multinomial family the same prior is applied independently to every non-baseline category.
nsave	Number of posterior draws to keep. Default 4000.
nburn	Number of burn-in iterations. Default 1000.
thin	Thinning interval: one draw is kept every thin iterations, so the sampler runs nburn + nsave * thin iterations in total and retains nsave draws. Default 1.
horizon	Forecast horizon H (a non-negative integer). When $H \geq 1$, an H-step forecast is simulated after sampling and stored for retrieval with forecast.dynamic_fit() ; when $H = 0$ (default) nothing is stored, and forecasts can still be computed later with forecast(fit, horizon = H) .
forecast_trials	For the binomial and multinomial families, the number of trials (binomial) or the total count per period (multinomial) over the forecast horizon (length 1, recycled, or length horizon). If omitted it defaults to the last observed number of trials (binomial) or the last non-zero row total (multinomial). Used only when horizon ≥ 1 .
forecast_offset	Known offset over the forecast horizon. For the Poisson and binomial families a vector of length 1 (recycled) or horizon; for the multinomial family a scalar, a

	length $K - 1$ vector (one constant per non-baseline category) or an $H \times (K - 1)$ matrix on the ALR scale (columns aligned with the non-baseline categories in their original order). Defaults to $\emptyset$, with a warning if <code>offset</code> is non-zero. Used only when <code>horizon</code> ≥ 1 .
<code>offset</code>	Optional known per-observation offset on the linear-predictor scale. For the Poisson family (length 1 or n) this is a log-exposure term, so the mean is $\exp(\text{offset}_t + z_t)$; for the binomial family it shifts the logit. For the multinomial family it is a scalar, a length $K - 1$ vector (one constant per non-baseline category) or an $n \times (K - 1)$ matrix of per-category shifts on the ALR scale (columns aligned with the non-baseline categories in their original order; the baseline has no offset). Default NULL (no offset).
<code>baseline</code>	Multinomial family only: the baseline category, given as a column index or a column name of <code>y</code> , or "largest" (default) to use the category with the largest total count over the series (ties broken by column order). A category with many observations makes the ALR transform numerically well behaved. Note that the model is <i>not</i> invariant to this choice: the latent dynamics are placed on the log-ratios relative to the baseline, so if the baseline's own share moves a lot every ALR series inherits that movement. A large category with a stable share is the natural choice. For the other families a non-default value is ignored with a warning.
<code>verbose</code>	Logical; print a progress bar. Default FALSE.
<code>seed</code>	Optional random seed for reproducibility. The previous state of the global random number generator is restored after fitting.

Details

Estimation. The model is estimated by Metropolis-within-Gibbs MCMC. The latent states are updated one site at a time by adaptive random-walk Metropolis steps that use their Gaussian Markov random field (GMRF) full conditionals. States without an observation (the initial state, structural zeros, zero-total rows) are drawn exactly from their Gaussian full conditionals. The innovation parameters, the drift/intercept μ and the AR(1) coefficient ρ are updated by Gibbs steps (with a Metropolis step for the Student-t degrees of freedom).

Multinomial family. With `family = "multinomial"`, `y` is an $n \times K$ matrix of category counts (rows = time, columns = categories, $K \geq 2$); the row totals N_t are treated as known trials. One category b is the **baseline**, and the remaining $K - 1$ categories each get their own latent additive-log-ratio (ALR) series $z_{t,k} = \log(p_{t,k}/p_{t,b})$, so that

$$y_t \sim \text{Multinomial}(N_t, p_t), \quad p_{t,k} = \frac{\exp(o_{t,k} + z_{t,k})}{1 + \sum_{j \neq b} \exp(o_{t,j} + z_{t,j})}, \quad p_{t,b} = \frac{1}{1 + \sum_{j \neq b} \exp(o_{t,j} + z_{t,j})},$$

with known offsets $o_{t,k}$ (zero by default). The chosen latent dynamics, innovation structure and drift/intercept setting apply to every ALR series, but *no parameters are shared across categories*. Each series has its own innovation variance (and, where relevant, degrees of freedom, mixture components or volatility path), its own ρ and μ , and its own copy of prior. The series are coupled only through the multinomial likelihood, and each series is updated with the other categories held at their current values. A row with $N_t = 0$ carries no information about the shares and is handled like a missing observation. Zero inflation is not available for this family (zeros must be "none"). Running time grows linearly in $K - 1$.

Zero handling. Under `zeros = "inflated"` a latent gate decides, for each observed zero, whether it is structural (gate closed) or an ordinary sampling zero produced by the Poisson/binomial process (gate open); the gate-open probability is a single parameter that is constant over time. See `structural_zero_prob()`. Under `zeros = "missing"` the observed zeros are treated as missing values.

Forecasting. Forecasts are obtained by forward simulation from the posterior draws, so they can be computed after fitting for any horizon with `forecast.dynamic_fit()`. Setting `horizon = H` additionally simulates an H-step forecast right after sampling and stores it in the fit (under the same seed).

Reproducibility. With a seed, the sampler and the fit-time forecast run with that seed, and the previous state of the global random number generator is restored afterwards.

Value

An object of class "dynamic_fit": a list with three elements.

`draws` Posterior draws only. For the Poisson and binomial families a list of matrices/vectors with one row (or element) per stored draw:

`z` Latent states aligned with the observations (`draws x n`).

`z0` The initial latent state, one period before the first observation, which carries the `init_mean` / `init_var` prior.

`sig2` Variance of the increment leading into each z_t (`draws x n`); the first column is the increment from `z0`.

`fitted`, `yrep` **Unconditional** fitted means and posterior predictive replicates; under `zeros = "inflated"` they include the zero-inflation gate.

`fitted_open`, `yrep_open` Their **conditional-on-gate-open** counterparts: the latent-implied mean, and a replicate drawn straight from the observation model.

`gate`, `pi_open` Zero-inflation gate indicators and gate-open probability; NULL unless `zeros = "inflated"`.

`rho`, `mu` AR(1) coefficient (1 under the random walk) and drift/intercept (0 unless `include_mu = TRUE`).

`innov_var` A representative innovation variance whose definition depends on innovations: the estimated constant increment variance σ^2 for "gaussian"; the marginal increment variance for "t" ($\sigma^2 \nu / (\nu - 2)$) and "mixture" ($\sigma^2 \sum_h \eta_h \sigma_h^2$); and the average of the per-increment variances $\exp(h_t)$ over the in-sample transitions for "sv". For "t", if a draw has $\nu \leq 2$ (possible only when `df_min` <= 2), the undefined variance factor is replaced by the convention 3.

`scale` The overall innovation scale σ^2 (NULL for "sv").

`nu` Student-t degrees of freedom (NULL unless `innovations = "t"`).

`mix_weight`, `mix_var` Mixture weights and component variances ($\sigma^2 \sigma_h^2$), `draws x mix_components` (NULL unless `innovations = "mixture"`). The components are exchangeable and not identified individually. Within each draw they are stored in order of increasing variance, which is a labelling convention rather than an identification, so per-component summaries are meaningful only if the components are clearly separated. Label-invariant quantities, such as `innov_var` and the forecasts, are unaffected.

`sv_mu`, `sv_phi`, `sv_sigma` Level, persistence and volatility of the AR(1) log-variance process (NULL unless `innovations = "sv"`).

forecast_z, forecast_y Latent and response forecasts when horizon ≥ 1 (NULL otherwise); forecast_y is unconditional, i.e. includes the gate.

Use yrep, not yrep_open, for posterior predictive checks; without zero inflation the conditional and unconditional pairs are identical. See [predict.dynamic_fit\(\)](#).

For the multinomial family the same components carry an extra trailing *category* dimension, named by the category labels: latent quantities (z, sig2, forecast_z, mix_weight, mix_var) are arrays with one slice per non-baseline category, and z0, rho, mu, innov_var, scale, nu and the sv_* parameters are draws $\times$ (K - 1) matrices; response-scale quantities (fitted - the expected counts $N_{tp_{t,k}}$, yrep, forecast_y, and the additional fitted_prob / forecast_prob holding the category shares) are draws $\times$ time $\times$ K arrays including the baseline, in the original column order. fitted_open, yrep_open, gate and pi_open are NULL.

data The observed inputs: y, trials (the row totals for the multinomial family), the series length n, and the resolved per-observation offset. For the multinomial family also K, the categories (column labels), and the baseline index and baseline_name.

spec The model and MCMC specification: family, innovations, latent_dynamics, include_mu, zeros, prior, nsave, nburn, thin, horizon, and the forecast-period inputs forecast_offset / forecast_trials used for the stored forecast (NULL when not applicable).

See Also

[dynamic_prior\(\)](#), [forecast.dynamic_fit\(\)](#), [predict.dynamic_fit\(\)](#), [plot_fitted\(\)](#), [structural_zero_prob\(\)](#), [simulate_dynamic_multinomial\(\)](#)

Examples

```
sim <- simulate_dynamic_poisson(n = 60, sigma = 0.2, log_rate0 = 2, seed = 1)
fit <- fit_dynamic_model(sim$y, family = "poisson", nsave = 300, nburn = 200,
                        seed = 1)

summary(fit)
forecast(fit, horizon = 5)

# multinomial choice counts: three categories, baseline chosen automatically
simm <- simulate_dynamic_multinomial(n = 40, sigma = 0.15, trials = 100,
                                    alr0 = c(-1, -0.5), seed = 2)
fitm <- fit_dynamic_model(simm$y, family = "multinomial",
                        nsave = 200, nburn = 100, seed = 2)

fitm
summary(fitm)$params
```

Description

Returns posterior predictive forecasts for the H periods after the end of the series. The forecasts are obtained by forward simulation from the stored posterior draws, so any horizon can be requested after fitting. If the model was fitted with $\text{horizon} \geq 1$, the forecast draws stored in the fit are returned unless horizon , forecast_offset or forecast_trials is supplied.

Usage

```
## S3 method for class 'dynamic_fit'
forecast(
  object,
  horizon = NULL,
  forecast_offset = NULL,
  forecast_trials = NULL,
  probs = c(0.025, 0.5, 0.975),
  seed = NULL,
  ...
)
```

Arguments

<code>object</code>	A "dynamic_fit" object.
<code>horizon</code>	Forecast horizon H (a positive integer). If <code>NULL</code> (default), the forecast stored in the fit is returned; the fit must then have been created with $\text{horizon} \geq 1$.
<code>forecast_offset</code>	Known offset over the forecast horizon, in the same format as in fit_dynamic_model() . Defaults to 0; a warning is issued if the model was fitted with a non-zero offset but no <code>forecast_offset</code> is supplied.
<code>forecast_trials</code>	Binomial and multinomial families: the number of trials (binomial) or the total count per period (multinomial) over the forecast horizon, of length 1 (recycled) or horizon . Defaults to the last observed number of trials (binomial) or the last non-zero row total (multinomial).
<code>probs</code>	Quantile probabilities for the summary. Default <code>c(0.025, 0.5, 0.975)</code> .
<code>seed</code>	Optional random seed for the forward simulation. The previous state of the global random number generator is restored afterwards.
<code>...</code>	Unused.

Details

For every stored posterior draw, the latent path is propagated from the last in-sample state z_n with $z_{n+h} = \mu + \rho z_{n+h-1} + \varepsilon_{n+h}$, drawing the increments from the fitted innovation structure (Gaussian with the drawn variance, Student-t with the drawn scale and degrees of freedom, the drawn scale mixture, or the stochastic-volatility process continued from its last in-sample log-variance). A response is then drawn from the observation model at each simulated state. The intervals therefore reflect parameter, state and innovation uncertainty.

Zero inflation. For fits with `zeros = "inflated"` the response forecast is **unconditional**: each draw includes the zero-inflation gate, so structural zeros are reproduced and the draws are directly comparable to future observations (like the in-sample `yrep`, unlike `yrep_open`). The latent forecast summary in `$latent` is gate-free by construction.

Multinomial family. Response forecasts are multinomial draws with the totals given by `forecast_trials`. The summaries are in long format with a category column. `summary` and `final` cover all K categories on the count scale, `prob` gives the forecast category shares, and `latent` the $K - 1$ ALR series. `draws` is a `draws × H × K` array and `final_draws` a `draws × K` matrix.

Value

An object of class "dynamic_forecast": a list with `summary` (one row per horizon $1, \dots, H$, on the response scale), `latent` (summary of the forecast latent path), `draws` (the predictive draws, `draws × H`), `latent_draws` (the latent forecast draws), `final` (the single-row summary of the final H -step-ahead forecast), `final_draws` (the predictive draws at horizon H), the forecast horizon, and the `forecast_offset / forecast_trials` that were used. For the multinomial family additionally `prob` (share summary) and `prob_draws` (`draws × H × K` share draws); see Details for the layout.

See Also

`fit_dynamic_model()`, `plot_forecast()`

Examples

```
sim <- simulate_dynamic_poisson(60, 0.2, 2, seed = 1)
fit <- fit_dynamic_model(sim$y, nsave = 300, nburn = 200, seed = 1)
fc <- forecast(fit, horizon = 8)
fc$summary
fc$final
```

med_weekly

Weekly Mediterranean crossings (Mediterranean example)

Description

A longer weekly count series of irregular sea crossings on the Mediterranean route, covering the ISO weeks 2015-W40 to 2025-W11 (494 weeks). The counts are larger in magnitude with fewer zeros than [uk_weekly](#). The series is used in the irregular-migration application of Zens and Bijak (2026).

Usage

med_weekly

Format

A data frame with 494 rows and 3 variables:

week ISO week label, e.g. "2015-W40".

count Non-negative integer count of weekly crossings.

date The Monday of the ISO week, as a Date.

Source

Weekly aggregates of detected irregular Mediterranean crossings, compiled from operational/agency records as described in Zens and Bijak (2026), [doi:10.1214/26AOAS2171](#).

References

Zens, G. and Bijak, J. (2026). Dynamic Count Models with Flexible Innovation Processes for Irregular Maritime Migration. *The Annals of Applied Statistics*, 20(2), 1671–1690. [doi:10.1214/26AOAS2171](#).

Examples

```
summary(med_weekly$count)
plot(med_weekly$date, med_weekly$count, type = "h", xlab = "week", ylab = "crossings")
```

plot.dynamic_fit	<i>Plot method for fitted dynamic models</i>
------------------	--

Description

A convenience wrapper that dispatches to the dedicated plotting functions.

Usage

```
## S3 method for class 'dynamic_fit'
plot(x, which = c("fitted", "latent", "forecast", "zeros"), ...)
```

Arguments

x	A "dynamic_fit" object.
which	One of "fitted" (default), "latent", "forecast", "zeros".
...	Passed to the underlying plotting function (e.g. category for multinomial fits, or horizon for which = "forecast").

Value

Invisibly, the result of the underlying plotting function.

plot_fitted	<i>Plot observed versus fitted values</i>
-------------	---

Description

Observed series with the posterior median fitted mean and a credible band on the response scale. For the multinomial family, one panel per category showing the observed and expected counts $N_t p_{t,k}$.

Usage

```
plot_fitted(object, level = 0.95, category = NULL, ...)
```

Arguments

object	A "dynamic_fit" object.
level	Credible level. Default 0.95.
category	Multinomial family only: which categories to draw (names or indices among the non-baseline categories for <code>plot_latent()</code> ; among all K categories for <code>plot_fitted()</code> and <code>plot_forecast()</code>). NULL (default) draws one panel per category.
...	Passed to <code>graphics::plot()</code> ; may override the default main, xlab, ylab, ylim and other plot arguments.

Value

Invisibly, the fitted summary.

Examples

```
sim <- simulate_dynamic_poisson(60, 0.2, 2, seed = 1)
fit <- fit_dynamic_model(sim$y, nsave = 300, nburn = 200, seed = 1)
plot_fitted(fit)
```

plot_forecast	<i>Plot observed history, fitted values and forecast with uncertainty</i>
---------------	---

Description

Shows the observed series together with the in-sample fitted mean (median and credible band) and the forecast (median and credible band) over the forecast horizon. The forecast is the one stored in the fit, or a new one computed with `forecast.dynamic_fit()` when horizon (or another forecast input) is supplied. For the multinomial family, one panel per category.

Usage

```
plot_forecast(
  object,
  level = 0.95,
  category = NULL,
  horizon = NULL,
  forecast_offset = NULL,
  forecast_trials = NULL,
  seed = NULL,
  ...
)
```

Arguments

object	A "dynamic_fit" object.
level	Credible level. Default 0.95.
category	Multinomial family only: which categories to draw (names or indices among the non-baseline categories for <code>plot_latent()</code> ; among all K categories for <code>plot_fitted()</code> and <code>plot_forecast()</code>). NULL (default) draws one panel per category.
horizon, forecast_offset, forecast_trials, seed	Passed to <code>forecast.dynamic_fit()</code> . With the defaults, the forecast stored in the fit is plotted.
...	Passed to <code>graphics::plot()</code> ; may override the default main, xlab, ylab, ylim and other plot arguments.

Value

Invisibly, the "dynamic_forecast" object.

Examples

```
sim <- simulate_dynamic_poisson(60, 0.2, 2, seed = 1)
fit <- fit_dynamic_model(sim$y, nsave = 300, nburn = 200, seed = 1)
plot_forecast(fit, horizon = 12)
```

plot_latent	<i>Plot the fitted latent trajectory</i>
-------------	--

Description

Shows the posterior median of the latent state (log-rate for the Poisson family, logit for the binomial family, one additive-log-ratio series per non-baseline category for the multinomial family) with a credible band.

Usage

```
plot_latent(object, level = 0.95, category = NULL, ...)
```

predict.dynamic_fit *In-sample fitted values and posterior predictive replicates*

Description

In-sample fitted values and posterior predictive replicates

Usage

```
## S3 method for class 'dynamic_fit'
predict(
  object,
  type = c("mean", "response", "prob"),
  conditional = FALSE,
  probs = c(0.025, 0.5, 0.975),
  ...
)
```

Arguments

object	A "dynamic_fit" object.
type	"mean" (default) returns the posterior of the mean of y ; "response" returns posterior predictive replicates of y . For the multinomial family "prob" returns the posterior of the category shares $p_{t,k}$ (the mean is then the expected count $N_t p_{t,k}$).
conditional	Logical. Leave FALSE (the default) for anything compared against observed data – posterior predictive checks, calibration – and set TRUE only to inspect the latent intensity process. With FALSE, means and replicates come from the full zero-inflated model (the gate is included, so structural zeros are reproduced); with TRUE, they are conditional on the gate being open, i.e. drawn straight from the Poisson/binomial observation model, and show systematically too few zeros under zero inflation. Without zero inflation the two versions are identical, and the argument is ignored for the multinomial family (which has no gate). See DynCount-package for the gate notation.
probs	Quantile probabilities for the summary. Default <code>c(0.025, 0.5, 0.975)</code> .
...	Unused.

Value

A list with summary (a data frame, one row per observation) and draws (the underlying draws matrix, draws x time). For the multinomial family summary is in long format with one row per observation and category (columns time, category, observed, ...) and draws is a draws x time x K array.

Examples

```
sim <- simulate_dynamic_poisson(40, 0.2, 2, seed = 1)
fit <- fit_dynamic_model(sim$y, nsave = 200, nburn = 100, seed = 1)
head(predict(fit)$summary)           # posterior of the mean of y
head(predict(fit, type = "response")$summary) # posterior predictive
```

```
simulate_dynamic_binomial
```

Simulate a binomial dynamic series

Description

Generates a latent logit process (random walk or AR(1)) and binomial counts, optionally with structural (zero-inflation) zeros.

Usage

```
simulate_dynamic_binomial(
  n,
  sigma,
  trials,
  logit0 = 0,
  zero_inflation = 0,
  rho = 1,
  mu = 0,
  offset = 0,
  seed = NULL
)
```

Arguments

n	Number of observations.
sigma	Standard deviation of the latent increments (Gaussian).
trials	Number of trials: a single number (recycled) or a length-n vector.
logit0	Initial logit z_1 . Default 0.
zero_inflation	Structural-zero probability. With probability zero_inflation an observation is forced to a structural zero (gate closed) regardless of the binomial draw. Default 0 (no inflation).
rho	AR(1) coefficient of the latent process $z_t = \mu + \rho z_{t-1} + \varepsilon_t$. Default 1 (a random walk).
mu	Drift (random walk) / intercept (AR(1)) of the latent process. Default 0.
offset	Known offset on the logit scale (length 1 or n); the success probability is $\text{logit}^{-1}(\text{offset}_t + z_t)$. Default 0.
seed	Optional random seed. The previous state of the global random number generator is restored afterwards.

Value

A list with components `y` (successes), `trials`, `logit` (latent path z_t), `prob` (`plogis(offset + logit)`), `offset`, and `structural` (logical, TRUE for structural zeros).

Examples

```
sim <- simulate_dynamic_binomial(n = 50, sigma = 0.15, trials = 40, seed = 1)
head(sim$y)
# with structural zeros:
zi <- simulate_dynamic_binomial(50, 0.15, trials = 40, zero_inflation = 0.2, seed = 1)
mean(zi$structural)
```

simulate_dynamic_multinomial

Simulate a multinomial dynamic series

Description

Generates $K - 1$ independent latent additive-log-ratio (ALR) processes (random walk or AR(1)), one per non-baseline category, and multinomial choice counts with known totals. The ALR series $z_{t,k} = \log(p_{t,k}/p_{t,b})$ share no parameters: `sigma`, `rho` and `mu` may each be a single value (recycled) or a vector of length $K - 1$ giving one value per non-baseline category.

Usage

```
simulate_dynamic_multinomial(
  n,
  sigma,
  trials,
  alr0 = c(0, 0),
  baseline = length(alr0) + 1L,
  rho = 1,
  mu = 0,
  offset = 0,
  categories = NULL,
  seed = NULL
)
```

Arguments

<code>n</code>	Number of observations.
<code>sigma</code>	Standard deviation(s) of the latent increments (Gaussian); length 1 or $K - 1$.
<code>trials</code>	Total count per period: a single number (recycled) or a length- <code>n</code> vector.
<code>alr0</code>	Numeric vector of length $K - 1$: the initial ALR values $z_{1,k}$ of the non-baseline categories, in column order. Its length determines the number of categories K . Default <code>c(0, 0)</code> (three equally likely categories).

baseline	The baseline category: a column index in $1 \dots K$ of the returned count matrix, or one of the categories. Default K (last column). Pass the returned baseline to <code>fit_dynamic_model()</code> to fit the model on the same ALR scale as the simulation.
rho	AR(1) coefficient(s) of the latent processes $z_{t,k} = \mu_k + \rho_k z_{t-1,k} + \varepsilon_{t,k}$; length 1 or $K - 1$. Default 1 (random walks).
mu	Drift (random walk) / intercept (AR(1)) of the latent processes; length 1 or $K - 1$. Default 0.
offset	Known offset on the ALR scale: a scalar, a length $K - 1$ vector (one constant per non-baseline category) or an $n \times (K - 1)$ matrix (columns aligned with the non-baseline categories). Default 0.
categories	Optional character vector of length K with the category labels (column names of the returned counts). Default "cat1", ..., "catK".
seed	Optional random seed. The previous state of the global random number generator is restored afterwards.

Value

A list with components `y` (an $n \times K$ matrix of counts with the baseline in column `baseline`), `trials`, `alr` (the $n \times (K - 1)$ latent ALR paths $z_{t,k}$), `prob` (the $n \times K$ matrix of category probabilities), `offset`, `baseline` (the column index) and `categories`.

Examples

```
sim <- simulate_dynamic_multinomial(n = 50, sigma = 0.15, trials = 200,
                                   alr0 = c(-0.5, 0.5), seed = 1)
head(sim$y)
colSums(sim$y)
# fit on the simulated ALR scale by passing the simulated baseline
fit <- fit_dynamic_model(sim$y, family = "multinomial", baseline = sim$baseline,
                        nsave = 200, nburn = 100, seed = 1)
summary(fit)$params
```

simulate_dynamic_poisson

Simulate a Poisson dynamic series

Description

Generates a latent log-rate process (random walk or AR(1)) and Poisson counts, optionally with zero inflation.

Usage

```
simulate_dynamic_poisson(
  n,
  sigma,
  log_rate0 = 1,
  zero_inflation = 0,
  rho = 1,
  mu = 0,
  offset = 0,
  seed = NULL
)
```

Arguments

<code>n</code>	Number of observations.
<code>sigma</code>	Standard deviation of the latent increments (Gaussian).
<code>log_rate0</code>	Initial log-rate z_1 . Default 1.
<code>zero_inflation</code>	Probability that the gate is closed (i.e. the probability of a structural zero) at each time point. 0 (default) gives an ordinary Poisson series.
<code>rho</code>	AR(1) coefficient of the latent process $z_t = \mu + \rho z_{t-1} + \varepsilon_t$. Default 1 (a random walk).
<code>mu</code>	Drift (random walk) / intercept (AR(1)) of the latent process. Default 0.
<code>offset</code>	Known log-exposure offset (length 1 or n); the Poisson mean is $\exp(\text{offset}_t + z_t)$. Default 0.
<code>seed</code>	Optional random seed. The previous state of the global random number generator is restored afterwards.

Value

A list with components `y` (observed counts), `log_rate` (the latent log-rate path z_t), `rate` (the mean $\exp(\text{offset} + \text{log\_rate})$), `offset`, and `structural` (logical, TRUE where a structural zero was forced).

Examples

```
sim <- simulate_dynamic_poisson(n = 50, sigma = 0.2, log_rate0 = 2,
                               zero_inflation = 0.2, seed = 1)
table(sim$y == 0, sim$structural)
```

summary.dynamic_fit	<i>Summarise a fitted dynamic model</i>
---------------------	---

Description

Summarise a fitted dynamic model

Usage

```
## S3 method for class 'dynamic_fit'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	A "dynamic_fit" object.
probs	Quantile probabilities. Default c(0.025, 0.5, 0.975).
...	Unused.

Value

An object of class "summary.dynamic_fit" containing posterior summaries of the global parameters (params, one row per parameter) and of the fitted values (fitted). The rows of params and the draws they summarise are

innov_sd sqrt(draws\$innov_var): the estimated increment SD for "gaussian" innovations, the marginal SD for "t" and "mixture", and the root of the series-average variance for "sv".

t_df draws\$n, the Student-t degrees of freedom.

sv_mu, sv_phi, sv_sigma draws\$sv_mu, draws\$sv_phi and draws\$sv_sigma, the parameters of the AR(1) log-variance process.

ar1_rho draws\$rho, the AR(1) coefficient.

drift_mu / intercept_mu draws\$mu, the random-walk drift or the AR(1) intercept.

gate_open_prob draws\$pi_open, the probability that the zero-inflation gate is open (one minus the structural-zero probability).

Rows are included only where relevant. For "mixture" innovations only innov_sd is reported. The mixture components are exchangeable and not identified individually, so per-component summaries are not given. The draws of the weights and component variances are in draws\$mix_weight and draws\$mix_var. For the multinomial family every latent parameter is reported once per non-baseline category, with the category label in square brackets (e.g. innov_sd[B]), and the fitted summary is a long data frame with a category column covering all K categories.

Examples

```
sim <- simulate_dynamic_poisson(50, 0.2, 2, seed = 1)
fit <- fit_dynamic_model(sim$y, nsave = 200, nburn = 100, seed = 1)
summary(fit)
```

uk_weekly

*Weekly English Channel crossings (UK example)***Description**

A weekly count series of irregular small-boat crossings of the English Channel towards the United Kingdom, covering the ISO weeks 2018-W01 to 2025-W11 (376 weeks). The series has frequent zeros in its early weeks (43% of the first 130 weeks), which makes it a useful example for zero inflation with the Poisson model. The series is used in the irregular-migration application of Zens and Bijak (2026).

Usage

uk_weekly

Format

A data frame with 376 rows and 3 variables:

week ISO week label, e.g. "2018-W01".

count Non-negative integer count of weekly crossings.

date The Monday of the ISO week, as a Date.

Source

Weekly aggregates of detected irregular English Channel crossings, compiled from operational/agency records as described in Zens and Bijak (2026), [doi:10.1214/26AOAS2171](https://doi.org/10.1214/26AOAS2171).

References

Zens, G. and Bijak, J. (2026). Dynamic Count Models with Flexible Innovation Processes for Irregular Maritime Migration. *The Annals of Applied Statistics*, 20(2), 1671–1690. [doi:10.1214/26AOAS2171](https://doi.org/10.1214/26AOAS2171).

Examples

```
plot(uk_weekly$date, uk_weekly$count, type = "h", xlab = "week", ylab = "crossings")
mean(uk_weekly$count == 0)
```

Index

* datasets

med_weekly, 11

uk_weekly, 23

dynamic_prior, 2

dynamic_prior(), 6, 9

DynCount-package, 5, 6, 16

fit_dynamic_model, 5

fit_dynamic_model(), 2, 4, 10, 11, 19

forecast.dynamic_fit, 9

forecast.dynamic_fit(), 6, 8, 9, 13, 14

graphics::barplot(), 15

graphics::plot(), 13–15

med_weekly, 11

plot.dynamic_fit, 12

plot_fitted, 13

plot_fitted(), 9, 13–15

plot_forecast, 13

plot_forecast(), 11, 13–15

plot_latent, 14

plot_latent(), 13–15

plot_zero_inflation, 15

predict.dynamic_fit, 16

predict.dynamic_fit(), 9

simulate_dynamic_binomial, 17

simulate_dynamic_binomial(), 6

simulate_dynamic_multinomial, 18

simulate_dynamic_multinomial(), 9

simulate_dynamic_poisson, 19

simulate_dynamic_poisson(), 6

stochvol::specify_priors(), 4

structural_zero_prob, 21

structural_zero_prob(), 6, 8, 9, 15

summary.dynamic_fit, 22

uk_weekly, 11, 23