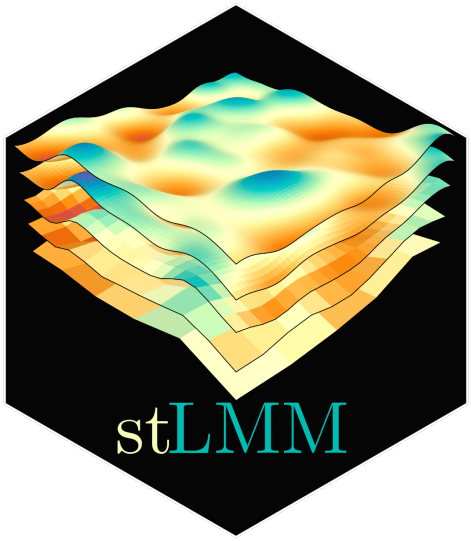
# stLMM

Modeling and Software Details

Andrew O. Finley

Updated June 16, 2026



# Contents

# 1 Motivation and overview

The `stLMM` package is in development to meet a growing interest in exploring and assessing small area estimation (SAE) models that support forest inventory for a range of space-time domains. While many R packages can fit particular SAE model classes, few single packages specialize in broad classes of unit- and area-level models tailored to data and settings encountered in inventory applications. Simplified exploration and application motivate this package's unified model specification and estimation framework. Although the specific motivation for this package was inventory applications, the methods are general and should find use in settings with similar data and inferential objectives.

The package fits linear mixed models (LMMs) with parameters estimated using Bayesian methods. To accommodate different SAE model classes, the LMMs can have fixed effects, unstructured random effects, diverse types of structured point- and area-referenced space-time random effects, and different residual variance structures. There is a particular focus on computational efficiency for space and time dependence structures appropriate for space- and time-indexed datasets, hence the `st` in the package name. The package allows fine-tuned control over random-effect specifications to explore model specification, computing, and estimation, while also providing general defaults for routine applications.

The unified design is based on a collapsed model specification, where high-dimensional structured random effects are integrated out, leaving only the low-dimensional covariance parameters that control space-time dependence to be estimated. Efficient computing using the collapsed model comes from working with structured random effects that have sparse precision matrices and estimation methods that take advantage of these sparse matrices.

Random effect terms are:

- independent and identically distributed (iid) grouped random-effect term `iid()`;
- Gaussian process (GP) term `gp()`;
- nearest-neighbor Gaussian process (NNGP) term `nngp()`;
- autoregressive AR(1) term `ar1()`;
- conditional autoregressive (CAR) term `car()`;
- directed acyclic graph autoregressive (DAGAR) term `dagar()`;
- separable CAR-time term `car_time()`;
- separable DAGAR-time term `dagar_time()`.

With the exception of `gp()`, all other process terms have sparse precision matrices.

Terms can serve as group-varying or space-time varying effects via a `:` operator. For example, varying effects on a generic covariate `x` can be written as `x:iid()`, `x:gp()`, `x:nngp()`, `x:car()`, `x:dagar()`, `x:car_time()`, or `x:dagar_time()`.

Space and space-time point-referenced data are handled using the `nngp()` and `gp()` terms, with covariance functions discoverable through `get_cor_models()`. For spatial data, these covariance functions include the Matérn and exponential. For space-time data, separable and nonseparable functions are available. The package design makes it straightforward to add new covariance models

to the registry. Any number of coordinate columns can be passed to define the Euclidean space used for constructing the covariance. For space-time covariance models, coordinate order is part of the model, with the final coordinate treated as time.

The `car()` and `car_time()` terms accommodate spatial and space-time areal data using proper CAR specifications. A spatial adjacency graph is supplied to each function. The graph can be constructed using `car_graph()`. The space-time `car_time()` term fits a separable model, where a time column defines the temporal support. The temporal component can use either an ordered-support AR(1) precision or a continuous-time exponential correlation with a temporal decay parameter. The `dagar()` term uses the same graph support as `car()`, but orients the graph according to a user-specified ordering and applies the ordered DAGAR construction of Datta et al. (2019). The `dagar_time()` term combines this ordered DAGAR spatial precision with the same temporal precision options used by `car_time()`.

The package also retains model metadata needed for post-fit recovery, fitted values, and prediction. The intended workflow is

```
fit <- stLMM(...)
rec <- recover(fit, ...)
fitted(rec)
predict(rec, newdata = ...)
```

where subsampling of posterior parameter draws can occur at the recovery and prediction steps. For Pólya-Gamma process models, `predict(fit, ...)` can also use saved in-chain process draws directly; `recover()` is then a selection and labeling step rather than a stochastic reconstruction step.

For models without structured process terms, `fitted()` and `predict()` can be called directly on the fit object. For Gaussian-likelihood models with structured process terms, `recover()` is the explicit post-fit step that samples latent process draws conditional on retained posterior parameter draws. For Pólya-Gamma process models, the sampler retains in-chain process draws by default as `save_process = list(start = 1, thin = 1)` and `recover()` selects from those saved draws. Enabling `save_process` is intentionally limited to Pólya-Gamma models with structured process terms; Gaussian-likelihood process models remain collapsed and use post-fit recovery.

## 2 Modeling framework

The package is built around a unified sparse-precision view of Bayesian Gaussian mixed models. The basic observation model is

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{\alpha} + \boldsymbol{A}\boldsymbol{w} + \boldsymbol{\varepsilon}, \tag{1}$$

where

- $\boldsymbol{X}\boldsymbol{\beta}$ are fixed effects;
- $\boldsymbol{Z}\boldsymbol{\alpha}$ are explicit iid grouped random effects;
- $\boldsymbol{w}$ is the stacked structured latent process vector;
- $\boldsymbol{A}$ maps observation rows to latent process nodes and applies any covariate scaling for space-time varying coefficient terms;

- $\boldsymbol{\varepsilon}$ is the Gaussian residual component.

The default residual model is homoskedastic

$$\boldsymbol{\varepsilon} \sim N(0, \tau^2 \boldsymbol{I}_n), \tag{2}$$

with options set via a single `resid()` term for a more general diagonal observation precision

$$\boldsymbol{W} = \mathrm{diag}(p_1, \ldots, p_n), \tag{3}$$

where $p_i = \tau^{-2}$ under the default model, $p_i = 1/\widehat{v}_i$ under fixed direct estimate residual variances, $p_i = 1/\tau_i^2$ under sampled direct-estimate residual variance models, with additional options described in the manual page.

The fixed-effect coefficient vector is indexed as

$$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top . \tag{4}$$

The explicit iid random effects have

$$\boldsymbol{\alpha} \sim N(0, \boldsymbol{D}_\alpha), \tag{5}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_q)^\top$ and $\boldsymbol{D}_\alpha$ is block diagonal by iid term. The structured process block has

$$\boldsymbol{w} \sim N(0, \boldsymbol{Q}_w^{-1}), \tag{6}$$

with $\boldsymbol{Q}_w$ assembled as a block-diagonal prior precision over process terms.

For one process term with latent support $s_1, \ldots, s_{q_k}$, the finite latent vector is

$$\boldsymbol{w}_k = (w_k(s_1), \ldots, w_k(s_{q_k}))^\top . \tag{7}$$

If there are $K$ process terms, then

$$\boldsymbol{w} = \left(\boldsymbol{w}_1^\top, \cdots, \boldsymbol{w}_K^\top\right)^\top, \qquad \boldsymbol{Q}_w = \mathrm{blockdiag}(\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K), \tag{8}$$

and the total latent dimension is

$$q_{\mathrm{lat}} = \sum_{k=1}^{K} q_k. \tag{9}$$

The main sampler is collapsed with respect to $\boldsymbol{w}$, i.e., structured latent process draws are integrated out during parameter updates. For Gaussian-likelihood models with structured process terms, process draws may be recovered after fitting by `recover()`. Pólya-Gamma process models retain the warm-running in-chain process draws used by the augmentation step, with the default retained grid `save_process = list(start = 1, thin = 1)`.

This formulation is not the fastest sampler for each individual model class, but it gives one coherent implementation for point-referenced, areal, temporal, and space-time terms.

## 3 Current interface

The user-facing functions are deliberately small:

- `stLMM()` fits the model;
- `recover()` recovers latent process draws from a fit;
- `predict()` predicts given new data;
- `log_lik()` and `waic()` compute pointwise log-likelihood matrices and WAIC on the observed rows;
- S3 methods for printing, summaries, and plots.

The `log_lik()` calculation is conditional on the latent model expression used by fitted values. Models without structured process terms can be evaluated directly from the fit object. Gaussian-likelihood process models require recovered latent process draws, while Pólya-Gamma process models can use the saved in-chain process draws when `save_process` retained them. The optional `loo` package is used only when `waic()` is called.

Backend helpers such as graph builders, correlation registries, and nearest-neighbor index constructors are internal implementation details, not part of the API.

The current `stLMM()` signature is

```
stLMM(formula, data = parent.frame(), starting = list(), tuning = NULL,
     priors = NULL, family = "gaussian", trials = NULL, size = NULL,
     n_samples, n_omp_threads = 1,
     nngp_search = c("fast", "brute"),
     verbose = TRUE, n_report = 100, warmup = TRUE,
     metropolis = list(), cholmod_control = list(), chains = 1L,
     save_process = NULL, chain_control = list(),
     describe_terms = FALSE, ...)
```

To simplify model specification, not all prior distributions need to be specified by the user. Reasonable defaults are used for some low-level quantities, such as the fixed-effect prior and default starts. Other priors, especially structured-process covariance priors, require explicit user consideration and are therefore required.

The `family` argument currently supports `"gaussian"`, `"binomial"`, and `"negative_binomial"`. For `family = "binomial"`, the response is the number of successes and `trials` supplies the number of trials. If `trials` is omitted, each row is treated as one Bernoulli trial. Binomial models use a logit link, do not use `resid()` residual variance terms, and do not have a `tau_sq` parameter. For `family = "negative_binomial"`, the response is a nonnegative integer count and `size` supplies a fixed positive negative-binomial size parameter. Negative-binomial models use a log-mean link and also omit Gaussian residual variance terms.

Standard R formula offsets are supported through `offset()`. The offset is a known additive component of the linear predictor. For count models with an exposure or expected-count scaling $E_i$, users can write `offset(log(E))`. Section 10 gives the negative-binomial model and computational rationale.

### 3.1 Control lists

`starting` and `tuning` are optional. If omitted, the sampler uses conservative defaults, which should be checked against the scale and dependence structure of the fitted model.

Priors are supplied through explicit prior constructors. Fixed effects may use `flat()` or `normal(mean, sd)`. The Gaussian likelihood defaults to `flat()` for $\boldsymbol{\beta}$, while Pólya-Gamma likelihoods default to `normal(mean = 0, sd = 10)`. The `normal()` constructor defines independent normal priors and allows scalar or coefficient-specific mean and standard-deviation vectors. The `flat()` and `normal()` constructors are accepted only for `priors$beta`.

The inverse-gamma constructor `ig(shape, scale)` uses the shape–scale parameterization. Variance parameters may use `ig()`, `log_normal()`, `gamma_dist()`, `half_normal()`, or `half_t()`, except iid grouped random-effect variances, which are currently Gibbs-updated and therefore use `ig()`. The half-normal and half-$t$ priors are specified on the standard deviation scale but are applied to the corresponding variance parameter with the appropriate density transformation. Covariance theta parameters use prior families allowed by the correlation-model registry and must have finite support so the sampler can use the bounded-logit transformation.

Parameter names are local to the model term. This follows the software API and keeps user-supplied `starting`, `tuning`, and `priors` lists readable, but it means the same symbol can have different meanings in different terms. For example, `phi` is a spatial decay parameter in an exponential GP or NNGP term, but it is an ordered-support one-step correlation in an `ar1()` term.

Process and sampled residual variance parameters are fixed via `fixed(value)` in the starting list. For example, `starting = list(nngp_1 = list(phi = fixed(6)))` fixes the `phi` parameter in the first NNGP term. Fixed parameters are skipped in the Metropolis update so they do not add computing overhead, but they still contribute to the likelihood.

### 3.2 Multiple chains and coda diagnostics

The `chains` argument runs multiple MCMC chains sequentially from the same parsed model. With `chains = 1`, `stLMM()` returns the ordinary `stLMM` object. With `chains > 1`, it returns an `stLMM_chains` object whose `chains` component is a list of ordinary `stLMM` fits. This keeps downstream behavior explicit: `recover()` returns an `stLMM_recovery_chains` object with one recovered object per chain, and `predict()` returns an `stLMM_prediction_chains` object with one prediction object per chain. Subsampling for recovery and prediction is applied within each chain.

If the user omits starting values, multi-chain fitting uses the same default centers as single-chain fitting but disperses positive variance starts on the log scale and bounded covariance starts within their prior bounds. The `chain_control` list supports `seed`, which sets the R random seed before chain starts are generated, and `dispersion`, which controls log-scale jitter for omitted positive starts.

If the user supplies a starting value, that value is treated as authoritative. Scalar starts are recycled to every chain. Vectors with length `chains` define chain-specific starts.

The package provides `as_mcmc()` as a bridge to `coda`. Single-chain objects convert to `coda::mcmc`; multi-chain objects convert to `coda::mcmc.list`. The `summary()` method for multi-chain fits reports pooled posterior summaries and coda-style chain diagnostics including point-estimate $\hat{R}$ and effective sample size where available.

### 3.3 Formula model terms

The formula language distinguishes explicit iid grouped random effects from structured latent process terms:

- `iid(group)` adds an iid Gaussian random intercept over observed group levels;
- `x:iid(group)` adds an iid Gaussian random slope for numeric covariate `x`;
- `iid(group) + x:iid(group)` creates two scalar iid terms with separate variance parameters, with term objects labeled in the order they appear in the model, e.g., `iid_1` and `iid_2`;
- `nngp(...)`, `gp(...)`, and `ar1(...)` add structured latent process terms;
- `car(area, graph = g)` adds a CAR latent process over an areal graph produced by `car_graph()`; the `car_model` argument selects the proper or Leroux spatial precision;
- `car_time(area, time, graph = g, time_model = "ar1")` adds a separable CAR-by-AR1 latent process over graph areas crossed with fitted time values, again using `car_model` for the spatial CAR component;
- `x:nngp(...)`, `x:gp(...)`, and `x:ar1(...)` create covariate-scaled process terms; the same convention applies to `x:car(...)` and `x:car_time(...)`.

The iid terms use the explicit $\boldsymbol{Z\alpha}$ random-effect and do not require post-fit recovery. Structured process terms are collapsed during parameter updates. Gaussian-likelihood process fits require `recover()` before process contributions are used in fitted values or prediction, while Pólya-Gamma process fits save in-chain process draws by default and `recover()` selects from those draws when a narrower retained subset is desired. The retained Pólya-Gamma process grid can be controlled with `save_process = list(start = ..., thin = ...)` when memory is a concern.

The `resid(...)` term is not a latent random-effect term. It specifies the Gaussian residual variance model and is parsed separately from the fixed-effect, iid, and structured-process terms. Omitting `resid()` is equivalent to `resid(model = "tau_sq")`, which estimates one global residual variance. Users may also write this default explicitly when it helps make a model formula self-documenting. Other `resid()` models allow fixed row-specific variances, group-specific residual variances, and direct-estimate variance models used in Fay-Herriot-style analyses. These options are described in Section 6 and in the `?resid` manual page.

## 4 Latent supports and backend metadata

Each structured term contributes a term-specific latent support and a latent block of size $q_k$. The support may be unique point coordinates, unique AR1 time values, graph areas, or graph areas crossed with fitted time values. Different process terms do not have to share the same support or coordinates.

Repeated fitted rows can map to the same latent node. For example, observations with the same coordinates or point observations within the same group or area are allowed to map to the same latent node. Missing response values indicated with an `NA` are permitted. They do not contribute to the likelihood, but their structured effects are recovered via `recover()`; see Section 8 for details.

The fitted object retains the backend object, including the original design matrices, process terms, graphs, term descriptions, and coordinate metadata. This retained backend is essential for recovery, fitted values, prediction, and diagnostics.

# 5 Model terms

This section describes the model terms from the user-facing formula level down to their latent supports, prior precision blocks, recovery behavior, and current prediction limits.

## 5.1 Fixed effects

Fixed effects use ordinary R formula syntax and enter the model through $\boldsymbol{X\beta}$. Standard formula offsets enter as a known vector $\boldsymbol{o}$ and are not sampled. In formulas, users write `offset(expr)`; internally the fitted linear predictor is $\boldsymbol{o} + \boldsymbol{X\beta} + \boldsymbol{Z\alpha} + \boldsymbol{Aw}$. Fixed effects are sampled directly during fitting using the collapsed Gaussian update described later. Fixed effects and offsets do not require post-fit recovery.

## 5.2 iid grouped random-effect terms

`iid(group)` adds an iid Gaussian random intercept over observed group levels. `x:iid(group)` adds an iid Gaussian random slope for numeric covariate `x`. These terms use the explicit $\boldsymbol{Z\alpha}$ rather than the collapsed structured-process.

Each iid term receives its own variance parameter. For example, `iid(group) + x:iid(group)` produces two iid random-effect variance controls, typically named `iid_1` and `iid_2`. The iid random effects are sampled directly during fitting and are returned in `alpha_samples`; no `recover()` call is needed for these coefficients.

As with $\boldsymbol{\beta}$, the choice not to fold $\boldsymbol{\alpha}$ into $\boldsymbol{w}$ is deliberate. Gibbs updates for these terms are inexpensive relative to the downside of adding more nonzero elements to the collapsed precision matrix and additional parameters to the Metropolis update. It is also useful to have $\boldsymbol{\alpha}$ immediately available after fitting.

## 5.3 NNGP terms

The space-time NNGP term is specified as `nngp(coord1, coord2, ..., m = 15, ordering = "coord", cov_model = "exp")`. `ordering` options are `"coord"`, `"default"`, `"maxmin"`, `"hilbert"`, `"random"`, or a numeric permutation of the graph nodes. The term's conditional representation is held in neighbor-weight vectors $B_k$ and conditional precision factors $F_k$, updated by `update_nngp_BF()`. Here $F_{k,i}$ is the inverse conditional variance on the unit-variance correlation scale. This is a precision-factor notation; it is the reciprocal of the conditional variance often denoted $F_i$ in the NNGP literature. These quantities do not include the process variance $\sigma_k^2$. The scalar factor $\sigma_k^{-2}$ is applied during latent precision assembly so NNGP terms follow the same convention as other process terms.

Conceptually,

$$\boldsymbol{Q}_k = \sigma_k^{-2} \widetilde{\boldsymbol{Q}}_k(\theta_k), \tag{10}$$

where $\widetilde{\boldsymbol{Q}}_k(\theta_k)$ is represented compactly by NNGP neighbor weights and conditional precision factors. The NNGP contribution to $\log|\boldsymbol{Q}_w|$ is

$$\log|\boldsymbol{Q}_k| = -q_k \log(\sigma_k^2) + \sum_{i=1}^{q_k} \log(F_{k,i}), \tag{11}$$

implemented inside `logdet_Qw_blocks()`.

The latent support is the unique fitted coordinate support for that term. Repeated observations at the same coordinates share the same latent node. Recovery returns NNGP latent draws in original support order for users, while the ordered internal support is retained for prediction and fitted-value alignment. Prediction supports existing fitted nodes, independent new-node prediction, moderate-size dense joint new-node prediction using fitted-support neighbors, and scalable Vecchia-style joint prediction in which earlier prediction nodes can enter the history set for later prediction nodes.

### 5.4 Dense GP terms

Dense GP terms are implemented primarily for smaller supports and for testing. A space-time GP term is specified as `gp(coord1, coord2, ..., cov_model = "exp")`. This term uses a dense correlation matrix

$$\boldsymbol{R}_k(\theta_k) = [\rho_{\theta_k}(s_i, s_j)]_{i,j=1}^{q_k}, \tag{12}$$

then forms

$$\boldsymbol{Q}_k = \sigma_k^{-2} \boldsymbol{R}_k(\theta_k)^{-1}. \tag{13}$$

This is handled by `update_gp_Q()`, which fills the dense correlation matrix, performs a dense Cholesky factorization, computes $\log|\boldsymbol{R}_k|$, inverts the correlation matrix, and scales by $\sigma_k^{-2}$. The log-determinant contribution is

$$\log|\boldsymbol{Q}_k| = -q_k \log(\sigma_k^2) - \log|\boldsymbol{R}_k|. \tag{14}$$

The sign difference between the NNGP and dense GP log determinants reflects their internal representation: the NNGP stores a sparse scale-free precision, whereas the dense GP is specified through a correlation matrix that is inverted to obtain the prior precision.

The latent support is the unique fitted coordinate support. Dense GP recovery uses the same collapsed Gaussian recovery equations as other structured processes. Prediction can add new dense GP nodes for small supports, but dense GP terms should not be used for large latent supports.

### 5.5 Point-referenced covariance functions

The covariance functions used by `gp()` and `nngp()` are stored in a small registry. Users can inspect the registry with `get_cor_models()`. The currently implemented families are `exp`, `matern`, `sep_exp`, `multi_res_sep_exp`, and `gneiting`. The model term contributes a process variance $\sigma^2$ and the covariance function contributes theta parameters named by the registry.

For the exponential covariance,

$$C(s_i, s_j) = \sigma^2 \exp\{-\phi d_{ij}\}, \tag{15}$$

where $d_{ij}$ is Euclidean distance and $\phi > 0$ is a decay parameter. For the Matérn covariance,

$$C(s_i, s_j) = \sigma^2 \frac{(\phi d_{ij})^\nu}{2^{\nu-1}\Gamma(\nu)} K_\nu(\phi d_{ij}), \qquad d_{ij} > 0, \tag{16}$$

with $C(s_i, s_i) = \sigma^2$. Here $\phi > 0$ is a decay parameter, $\nu > 0$ is the smoothness parameter, and $K_\nu(\cdot)$ is the modified Bessel function of the second kind.

There are several separable covariance options and one nonseparable option from the Gneiting family (Gneiting, 2002). These space-time models assume the final supplied coordinate is time. Reordering coordinates changes the model, not only the neighbor search.

### 5.6 AR1 terms

AR1 terms use tridiagonal precision formulas assembled directly from $\phi$ and $\sigma_k^2$. The numerical block is assembled by `assemble_ar1_block_into_M()`, and the determinant contribution is evaluated analytically inside `logdet_Qw_blocks()`. For $q_k > 1$, the stationary ordered-support AR1 prior precision is

$$\boldsymbol{Q}_k = \frac{1}{\sigma_k^2(1-\phi^2)} \begin{bmatrix} 1 & -\phi & 0 & \cdots & 0 \\ -\phi & 1+\phi^2 & -\phi & \ddots & \vdots \\ 0 & -\phi & 1+\phi^2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\phi \\ 0 & \cdots & 0 & -\phi & 1 \end{bmatrix}. \tag{17}$$

The one-node edge case is handled as a $1 \times 1$ precision matrix,

$$\boldsymbol{Q}_k = \left[\sigma_k^{-2}\right], \tag{18}$$

independent of $\phi$. The corresponding determinant contribution is

$$\log|\boldsymbol{Q}_k| = \begin{cases} -q_k \log(\sigma_k^2) - (q_k - 1)\log(1-\phi^2), & q_k > 1, \\ -\log(\sigma_k^2), & q_k = 1. \end{cases} \tag{19}$$

In the implementation this edge case is implicit: the same determinant calculation is used and the $(q_k - 1)\log(1-\phi^2)$ term vanishes when $q_k = 1$.

The latent support is the sorted unique fitted time support. Repeated fitted rows at the same time share one latent node. Prediction follows this fitted ordered-support convention: the current implementation treats the support as an ordered AR1 chain rather than as continuous-time gaps in the numeric time values. Thus $\phi$ is a one-step correlation on the sorted support, not a per-unit-time correlation. Unequally spaced values such as years $\{2000, 2005, 2010\}$ and $\{2000, 2001, 2002\}$ have the same AR1 graph structure once sorted.

### 5.7 CAR terms

Spatial CAR terms are specified as `car(area, graph = g, car_model = "proper")` or `car(area, graph = g, car_model = "leroux")`, where $g$ is an object produced by `car_graph()`. The graph can be built from a symmetric adjacency matrix or from an `sf` polygon layer. Polygon inputs require an explicit ID column so fitted data rows can be matched to graph areas.

The default spatial CAR precision is the degree-scaled proper CAR model

$$\boldsymbol{Q}_{\text{car}} = \sigma^{-2}(\boldsymbol{D} - \rho\boldsymbol{G}), \qquad 0 < \rho < 1, \tag{20}$$

where $\boldsymbol{G}$ is the symmetric binary adjacency matrix and $\boldsymbol{D}$ is the diagonal degree matrix. The graph stored by `car_graph()` is not row-standardized. Rather, this degree-scaled precision is equivalent to a CAR conditional specification with row-standardized neighbor averaging in the conditional mean,

$$E(w_i \mid \boldsymbol{w}_{-i}) = \frac{\rho}{d_i}\sum_{j\sim i} w_j, \qquad \text{Var}(w_i \mid \boldsymbol{w}_{-i}) = \frac{\sigma^2}{d_i}, \tag{21}$$

where $d_i$ is the number of neighbors for area $i$. Thus the CAR term averages neighboring random effects in the full conditional distribution while retaining a sparse precision written in terms of the binary adjacency and degree matrix Banerjee et al. (2014).

The Leroux CAR option (Leroux et al., 1999) uses the same graph object and parameter names, but replaces the spatial precision by

$$\boldsymbol{Q}_{\text{leroux}} = \sigma^{-2}\{(1-\rho)\boldsymbol{I} + \rho(\boldsymbol{D}-\boldsymbol{G})\}, \qquad 0 < \rho < 1. \tag{22}$$

This parameterization interpolates between iid area effects as $\rho$ approaches zero and intrinsic-CAR smoothing as $\rho$ approaches one. The exact endpoint $\rho = 1$ is not used because it gives a singular intrinsic precision for connected graphs. The current software API also uses the open lower bound $\rho > 0$ for sampled CAR association parameters; an exactly independent area effect is represented by `iid()` rather than by setting CAR `rho` to zero.

The restriction $\rho \in (0, 1)$ is an implementation and modeling choice that allows positive spatial association for the connected binary graphs used here. It is a deliberate narrowing of the full spectral proper-CAR parameter range, not a claim that nonpositive $\rho$ is generally improper. More generally, positive definiteness of $\boldsymbol{D} - \rho\boldsymbol{G}$ is governed by the eigenvalues of $\boldsymbol{D}^{-1/2}\boldsymbol{G}\boldsymbol{D}^{-1/2}$. If $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of this normalized adjacency matrix, then the proper range is

$$\lambda_{\min}^{-1} < \rho < \lambda_{\max}^{-1}. \tag{23}$$

For connected binary graphs, $\lambda_{\max} = 1$, so $\rho < 1$ is the upper bound; the lower bound can be negative and is ruled out by the current positive-association restriction.

`car_graph()` is conservative by default. Isolated areas and disconnected components error under `island = "error"`. For polygon layers with expected islands, `island = "nearest"` adds explicit nearest-neighbor bridge edges between disconnected components. The bridge metadata are retained in the graph object.

The graph objects have `plot()` methods intended for audit and teaching plots. For `sf`-based graphs, the plot overlays the adjacency edges on the polygon geometry and can highlight bridge edges added by `island = "nearest"`. For matrix-based graphs, the same method returns a `ggplot2` graph display using a deterministic circular layout. These plots are diagnostics of the graph construction, not part of the sampler.

CAR precision contributions are assembled sparsely into $\boldsymbol{M}$. The determinant $\log|\boldsymbol{Q}_{\text{car}}|$ is evaluated by CHOLMOD on a cached sparse CAR prior matrix. Conceptually, the proper and Leroux determinant contributions are

$$\log|\boldsymbol{Q}_{\text{car}}| = -S\log(\sigma^2) + \log|\boldsymbol{D} - \rho\boldsymbol{G}| \tag{24}$$

and

$$\log|\boldsymbol{Q}_{\text{leroux}}| = -S\log(\sigma^2) + \log|(1-\rho)\boldsymbol{I} + \rho(\boldsymbol{D}-\boldsymbol{G})|. \tag{25}$$

The sparse pattern and symbolic analysis are cached; numeric values are refreshed as $\sigma^2$ and $\rho$ change.

The latent support is the graph area support. Fitted data rows are matched to graph areas through the area variable. Recovery returns one latent value per graph area. Prediction is currently limited to areas already present in the fitted graph.

### 5.8 DAGAR terms

Spatial DAGAR terms are specified as `dagar(area, graph = g, ordering = "coord")`, where $g$ is an object produced by `car_graph()`. The graph supplies the undirected area adjacency. The `ordering` argument orients every adjacency edge from the earlier area to the later area and is therefore part of the model specification.

Let $\pi = \{\pi(1), \ldots, \pi(S)\}$ be the predetermined ordering of the graph areas, with inverse permutation $\pi^{-1}$. For area $i$, define the directed parent set

$$N_\pi(i) = \{j : i \sim j,\ \pi^{-1}(j) < \pi^{-1}(i)\}, \tag{26}$$

and let $n_\pi(i) = |N_\pi(i)|$. The directed graph $D_\pi = (V, E_\pi)$ contains the directed edges from each $j \in N_\pi(i)$ to $i$. The implementation checks that the supplied ordering is a permutation, that all directed parent indices point backward in the ordering, and hence that the constructed graph is acyclic.

Following Datta et al. (2019), the ordered DAGAR conditional model is

$$w_i \mid \boldsymbol{w}_{<i,\pi} \sim N\left(\frac{\rho}{1 + \{n_\pi(i) - 1\}\rho^2} \sum_{j \in N_\pi(i)} w_j,\ \frac{1 + \{n_\pi(i) - 1\}\rho^2}{1 - \rho^2}\right), \qquad 0 \le \rho < 1. \tag{27}$$

The second argument of the univariate normal distribution is precision. If $n_\pi(i) = 0$, then the conditional mean is zero and the conditional precision is one. This can occur for the first ordered node and can also occur for later nodes under arbitrary orderings. The term description reports the DAG parent-count summary and the number of zero-parent nodes.

The propriety range in the DAGAR construction includes $\rho = 0$, where the prior reduces to iid area effects. The current software API uses the open interval $0 < \rho < 1$ for DAGAR `rho`, matching the bounded-logit Metropolis update used for sampled spatial association parameters. Users can approximate independence with a small positive value or use an `iid()` term when an exactly independent area effect is intended. For common ordered lattice structures, Datta et al. (2019) show that $\rho$ has a direct neighbor-correlation interpretation, which is one motivation for the DAGAR construction.

Let $\boldsymbol{L}_\pi$ be the unit lower-triangular matrix whose row $i$ contains $-b_i$ for the directed parents of node $i$, with

$$b_i = \frac{\rho}{1 + \{n_\pi(i) - 1\}\rho^2}, \tag{28}$$

and let $\boldsymbol{F}_\pi$ be diagonal with entries

$$f_i = \frac{1 + \{n_\pi(i) - 1\}\rho^2}{1 - \rho^2}. \tag{29}$$

The scale-free DAGAR precision is

$$\widetilde{\boldsymbol{Q}}_{\text{dagar}}(\rho) = \boldsymbol{L}_\pi^\top \boldsymbol{F}_\pi \boldsymbol{L}_\pi. \tag{30}$$

Using the `stLMM` process-variance convention,

$$\boldsymbol{Q}_{\text{dagar}} = \sigma^{-2}\widetilde{\boldsymbol{Q}}_{\text{dagar}}(\rho), \qquad \boldsymbol{w} \sim N\{\boldsymbol{0}, \sigma^2 \widetilde{\boldsymbol{Q}}_{\text{dagar}}(\rho)^{-1}\}. \tag{31}$$

The determinant is evaluated from the conditional precisions:

$$\log |\boldsymbol{Q}_{\text{dagar}}| = -S \log(\sigma^2) + \sum_{i=1}^{S} \log f_i. \tag{32}$$

Like the NNGP term, `ordering` options are `"coord"`, `"default"`, `"maxmin"`, `"hilbert"`, `"random"`, or a numeric permutation of the graph nodes. Coordinate, max-min, and Hilbert orderings use polygon point-on-surface coordinates and therefore require an `sf`-based `car_graph()` object. For adjacency-matrix graph inputs, `"coord"` and `"default"` use the supplied row order. The max-min and Hilbert orderings are available as modeling options, but DAGAR ordering should be interpreted more cautiously than NNGP ordering: it defines the directed graph for an areal autoregression rather than a nearest-neighbor approximation to a continuous-domain Gaussian process.

The latent support is the graph area support. Internally, DAGAR precision assembly uses the ordering order, while recovered latent process samples are returned in the original graph-area order. Prediction is currently limited to areas already present in the fitted graph. The `plot()` method for internal DAGAR graph objects draws the directed parent edges implied by the selected ordering, which is useful for checking how an undirected areal graph was converted into a DAG.

### 5.9 Temporal precision used by areal time terms

The `car_time()` and `dagar_time()` terms use the same scale-free temporal precision options. For `time_model = "ar1"`, $\widetilde{\boldsymbol{Q}}_{\text{time}}$ is the ordered-support AR1 precision with parameter $\phi$. This is the same lattice AR1 convention used by `ar1()`: $\phi$ is a one-step correlation on the sorted fitted time support, not a per-unit-time correlation.

For `time_model = "exp"`, the temporal covariance is

$$C_t(t_i, t_j) = \exp\{-\lambda |t_i - t_j|\}. \tag{33}$$

For sorted one-dimensional time values, this is the covariance of a discretely observed Ornstein–Uhlenbeck process, and the inverse correlation matrix is tridiagonal even when the time gaps are unequal. If

$$r_j = \exp\{-\lambda(t_{j+1} - t_j)\}, \qquad j = 1, \ldots, T-1, \tag{34}$$

for sorted unique time values $t_1 < \cdots < t_T$, then the scale-free temporal precision $\widetilde{\boldsymbol{Q}}_{\text{time}}$ is sparse with nonzero entries only on the main diagonal and first off-diagonals. Thus exponential time correlation gives a sparse temporal precision without forming a dense space-time covariance matrix. This model has temporal decay parameter $\lambda > 0$ rather than AR1 parameter $\phi$.

### 5.10 DAGAR-time terms

Separable DAGAR-time terms are specified as `dagar_time(area, time, graph = g, ordering = "coord", time_model = "ar1")` or `dagar_time(area, time, graph = g, ordering = "coord", time_model = "exp")`. The graph support and ordering rules are the same as for `dagar()`. The fitted time support is the sorted set of unique time values in the fitting data. Internally, the latent support is ordered-area-major:

$$(\pi(1), t_1), \ldots, (\pi(1), t_T), (\pi(2), t_1), \ldots, (\pi(S), t_T). \tag{35}$$

Here $\pi$ is the DAGAR area ordering and $t_1 < \cdots < t_T$ are the fitted time values after sorting. User-facing recovered samples are returned in the original graph-area order crossed with fitted time order.

The implemented precision is

$$\boldsymbol{Q}_{\text{dagar},t} = \sigma^{-2}\{\widetilde{\boldsymbol{Q}}_{\text{dagar}}(\rho) \otimes \widetilde{\boldsymbol{Q}}_{\text{time}}\}, \tag{36}$$

where $\widetilde{\boldsymbol{Q}}_{\text{dagar}}(\rho)$ is the scale-free ordered DAGAR precision defined above and $\widetilde{\boldsymbol{Q}}_{\text{time}}$ is the scale-free temporal precision. There is one process variance $\sigma^2$ for the full area-time process.

The temporal factor uses the shared AR1 or exponential-time precision defined above. The exponential-time option supports unequal numeric time gaps.

The determinant uses the Kronecker identity:

$$\log|\boldsymbol{Q}_{\text{dagar},t}| = -ST\log(\sigma^2) + T\log|\widetilde{\boldsymbol{Q}}_{\text{dagar}}(\rho)| + S\log|\widetilde{\boldsymbol{Q}}_{\text{time}}|. \tag{37}$$

The DAGAR spatial factor and temporal factor are scale-free in this identity; the single $-ST\log(\sigma^2)$ term supplies the process-variance contribution.

Prediction for `dagar_time()` is currently limited to areas already in the fitted graph and time values already in the fitted time support. The area lookup is mapped through the fitted DAGAR ordering before indexing the internal latent vector. New graph areas and new fitted-time values error. The `plot()` method for the internal graph object displays the ordered DAGAR spatial graph used by the space-time term.

### 5.11 CAR-time terms

Separable CAR-time terms are specified as `car_time(area, time, graph = g, time_model = "ar1")` or `car_time(area, time, graph = g, time_model = "exp")`. The optional `car_model` argument selects the proper or Leroux CAR precision for the spatial factor. The fitted latent support is the Cartesian product of graph areas and sorted unique fitted time values. Internally, support is location-major:

$$(s_1, t_1), \ldots, (s_1, t_T), (s_2, t_1), \ldots, (s_S, t_T). \tag{38}$$

The current precision is

$$\boldsymbol{Q}_{\text{st}} = \sigma^{-2}(\widetilde{\boldsymbol{Q}}_{\text{space}} \otimes \widetilde{\boldsymbol{Q}}_{\text{time}}), \tag{39}$$

where $\widetilde{\boldsymbol{Q}}_{\text{space}} = \boldsymbol{D} - \rho\boldsymbol{G}$ for `car_model = "proper"` and $\widetilde{\boldsymbol{Q}}_{\text{space}} = (1-\rho)\boldsymbol{I} + \rho(\boldsymbol{D}-\boldsymbol{G})$ for `car_model = "leroux"`. The matrix $\widetilde{\boldsymbol{Q}}_{\text{time}}$ is the scale-free AR1 or exponential temporal precision for the fitted time support. There is one identifiable process variance $\sigma^2$ for the whole space-time process. Separate spatial and temporal variance parameters should not be added to this separable term because they are not identifiable.

The determinant uses the Kronecker identity:

$$\log|\boldsymbol{Q}_{\text{st}}| = -ST\log(\sigma^2) + T\log|\widetilde{\boldsymbol{Q}}_{\text{space}}| + S\log|\widetilde{\boldsymbol{Q}}_{\text{time}}|. \tag{40}$$

The spatial and temporal blocks in this determinant identity are scale-free; the single $-ST\log(\sigma^2)$ factor applies the process variance contribution to the full separable space-time precision.

The implementation assembles the sparse Kronecker precision pattern directly and reuses CHOLMOD for the collapsed $\boldsymbol{M}$ factorization.

Recovery returns one latent value per fitted area-time support point. Prediction is currently limited to existing graph areas and fitted time values. New graph areas or new time values require a separate prediction design because they change the latent support.

### 5.12 Space, time, and space-time varying terms

Space, time, and space-time varying terms are generically referred to as spatially varying coefficient (SVC) terms and are represented as covariate-scaled process terms, for example `x:nngp(...)`, `x:car(...)`, or `x:car_time(...)`. The latent process block is still a regular structured process block, but the observation-to-process map $\boldsymbol{A}$ includes the prediction or fitted-row covariate values. This means the SVC contribution to the linear predictor is $x_i w(s_i)$, not merely $w(s_i)$.

Recovery returns the underlying latent coefficient process $w(s)$. Fitted values and prediction multiply those recovered latent draws by the fitted or newdata covariate values before adding the term contribution to the linear predictor.

## 6 Residual variance models

`stLMM` specifies Gaussian residual variance models with a formula term `resid(...)`, rather than with a separate top-level argument. Only one `resid()` term is allowed in a model formula, and it is not a latent process term. It defines the diagonal observation precision used by the collapsed likelihood.

The default residual model estimates one global scalar $\tau^2$, with $p_i = \tau^{-2}$ for every observed row:

$$\epsilon_i \sim N(0, \tau^2). \tag{41}$$

Omitting `resid()` is equivalent to writing either `resid()` or `resid(model = "tau_sq")`. The prior, starting value, and tuning value for this default model are supplied through the `resid` control block, for example `priors = list(resid = list(tau_sq = ig(shape, scale)))`. This remains the recommended default unless the response is a direct estimate with a meaningful design-based variance, or unless residual variation is expected to differ across known groups.

### 6.1 Fixed row-specific variances

`resid(model = "fixed", variance = vhat)` treats supplied row-specific variances as fixed:

$$\tau_i^2 = \widehat{v}_i, \qquad p_i = 1/\widehat{v}_i. \tag{42}$$

Observed response rows require finite positive $\widehat{v}_i$. Missing response rows may have missing residual variances because they do not enter the likelihood. When this option is used, $\tau^2$ is not sampled and should not be supplied in `starting`, `tuning`, or `priors`. This is the usual way to enter known sampling variances in a direct-estimate model.

### 6.2 Group-specific residual variances

`resid(model = "group", group = g)` estimates one residual variance per observed residual group:

$$\epsilon_i \mid g_i \sim N(0, \tau_{g_i}^2), \qquad p_i = 1/\tau_{g_i}^2. \tag{43}$$

The grouping variable is term-agnostic. It may be a sampling stratum, measurement protocol, forest type, ownership class, area identifier, or any other known classification where residual variation is expected to differ. It does not have to match the support of a CAR, DAGAR, iid, or other random-effect term.

Priors for this model are supplied through `priors$resid`. A single prior can be recycled across groups, e.g.,

```
priors = list(resid = list(tau_sq = half_t(df = 3, scale = 1)))
```

or a named list can provide group-specific priors. Starting and tuning values are supplied analogously through `starting$resid$tau_sq` and `tuning$resid$tau_sq`. Scalars are recycled over groups, and named vectors or lists can be used when group-specific controls are needed. Values marked with `fixed()` are held fixed and excluded from Metropolis updates.

### 6.3 Direct-estimate-informed group variances

`resid(model = "group", group = g, variance = vhat, prior = "ig")` also estimates one residual variance per group, but uses externally supplied variance information to define the prior. This is useful in small-area settings where a direct estimate is accompanied by a design-based or otherwise externally estimated sampling variance.

For `prior = "ig"`, the prior is

$$\tau_g^2 \sim \mathrm{IG}(a, b_g), \tag{44}$$

where the argument `shape` is the inverse-gamma shape $a$. The `center` argument controls the scale $b_g$: `center = "mean"` sets $b_g = (a-1)\widehat{v}_g$, while `center = "mode"` sets $b_g = (a+1)\widehat{v}_g$.

`resid(model = "group", group = g, variance = vhat, n = n_eff, prior = "shannon")` uses an effective-sample-size construction. The prior is

$$\tau_g^2 \sim \mathrm{IG}\left(n_g/2, (n_g-1)\widehat{v}_g/2\right), \tag{45}$$

where $n_g$ is the supplied group-level effective sample size. The implementation requires $\widehat{v}_g$ and $n_g$ to be constant within each residual group, and requires $n_g > 1$. This prior is not exactly centered at $\widehat{v}_g$ by either its mean or its mode. For $n_g > 2$, its mean is $(n_g-1)\widehat{v}_g/(n_g-2)$, and its mode is $(n_g-1)\widehat{v}_g/(n_g+2)$. Both approach $\widehat{v}_g$ as $n_g$ grows, so larger $n_g$ produces a more concentrated prior around the supplied direct-estimate variance. For finite $n_g > 2$, the mean is above $\widehat{v}_g$ and the mode is below $\widehat{v}_g$. The symbol $n_g$ is local to this residual-variance prior and denotes an effective sample size for group $g$; it is unrelated to the binomial trial count $n_i$ used in the binomial likelihood section.

All sampled residual variance parameters in the group models enter the adaptive Metropolis update. They are not Gibbs-updated because the structured process has been marginalized in the collapsed likelihood.

### 6.4 Scaled direct-estimate residual variances

`resid(model = "scaled", variance = vhat)` estimates a low-dimensional multiplier:

$$\tau_i^2 = \kappa\widehat{v}_i. \tag{46}$$

The parameter $\kappa$ is sampled on the log scale, so the residual variance is strictly positive as long as $\widehat{v}_i > 0$.

When an effective sample size is supplied, `resid(model = "scaled", variance = vhat, n = n_eff)` uses

$$\log(\tau_i^2) = \log(\kappa) + \omega_i \log(\widehat{v}_i) + (1 - \omega_i) \log(\tau_0^2), \qquad \omega_i = \frac{n_i}{n_i + c}, \tag{47}$$

where $c$ is the positive `shrinkage` constant. Larger $n_i$ keeps $\tau_i^2$ closer to $\widehat{v}_i$; smaller $n_i$ shrinks toward the common scale $\tau_0^2$. Both $\kappa$ and $\tau_0^2$ are sampled on the log scale. This is a log-scale, or geometric, interpolation. When many $\omega_i$ values are close to zero, $\tau_i^2$ is driven mainly by the product $\kappa\tau_0^2$, so $\kappa$ and $\tau_0^2$ can be weakly separately identified. In that setting, priors and posterior diagnostics should be read with this identifiability limitation in mind.

# 7 Collapsed matrix and basic identities

Given the stacked process map $\boldsymbol{A}$, the sampler works with the sparse latent matrix

$$\boldsymbol{M} = \boldsymbol{Q}_w + \boldsymbol{A}^\top \boldsymbol{W} \boldsymbol{A}. \tag{48}$$

In the homoskedastic Gaussian special case, $\boldsymbol{W} = \tau^{-2}\boldsymbol{I}$, so this is equivalent to $\boldsymbol{Q}_w + \tau^{-2}\boldsymbol{A}^\top\boldsymbol{A}$.

This is the central matrix of the collapsed sampler. It is built in two stages:

1. `build_M_lat_pattern_sparse()` constructs the symbolic lower-triangular sparsity pattern,
2. `assemble_M_lat_numeric()` fills the numerical entries by adding $\boldsymbol{Q}_w$ and the observation contribution $\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}$.

After assembly, the matrix is factorized by CHOLMOD. Symbolic analysis is performed once through `M_cholmod_analyze()`, and numerical factorizations are refreshed through `M_cholmod_factorize()` as covariance parameters, residual precisions, or Pólya-Gamma working precisions change. Conceptually, write the CHOLMOD factorization as

$$\boldsymbol{P}\boldsymbol{M}\boldsymbol{P}^\top = \boldsymbol{L}\boldsymbol{L}^\top, \tag{49}$$

where $\boldsymbol{P}$ is a fill-reducing permutation and $\boldsymbol{L}$ is triangular. This notation is used to express the linear algebra in the same style as sparse collapsed Gaussian-process algorithms (see, e.g., Finley et al., 2019, Section 2.1). In the implementation, $\boldsymbol{L}$ is not extracted and triangular solves are not hand-coded. CHOLMOD owns the sparse factor object, including the permutation and its internal supernodal or simplicial representation, and `stLMM` dispatches solves through `M_cholmod_solve()`.

In the algorithms below, we write this CHOLMOD-backed solve as

$$\mathrm{solve}_M(\boldsymbol{b}) = \boldsymbol{x} \qquad \text{where} \qquad \boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}. \tag{50}$$

In the implementation, `solve_M_lat()` applies $\mathrm{solve}_M(\boldsymbol{b})$ by calling `M_cholmod_solve(CHOLMOD_A, ...)` on the current CHOLMOD factor. Thus $\mathrm{solve}_M(\boldsymbol{b})$ denotes a factor-based linear solve, not multiplication by a formed $\boldsymbol{M}^{-1}$.

For residual

$$\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Z}\boldsymbol{\alpha}, \tag{51}$$

let $\boldsymbol{R} = \boldsymbol{W}^{-1}$ denote the diagonal residual covariance. The collapsed observation covariance is

$$\boldsymbol{V} = \boldsymbol{R} + \boldsymbol{A}\boldsymbol{Q}_w^{-1}\boldsymbol{A}^\top. \tag{52}$$

This covariance matrix is a mathematical object only; the code does not form $\boldsymbol{V}$, $\boldsymbol{V}^{-1}$, or $\boldsymbol{M}^{-1}$. Under the homoskedastic Gaussian residual model, $\boldsymbol{R} = \tau^2\boldsymbol{I}_n$ and $\boldsymbol{W} = \tau^{-2}\boldsymbol{I}_n$. The Woodbury identity motivates the following observed-row precision operator. For any observed-row vector $\boldsymbol{v}$, define $\mathcal{V}_{\text{op}}^{-1}(\boldsymbol{v})$ by

$$\begin{aligned} \boldsymbol{u}_1 &= \boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{v}, \\ \boldsymbol{u}_2 &= \text{solve}_M(\boldsymbol{u}_1), \\ \boldsymbol{u}_3 &= \boldsymbol{A}\boldsymbol{u}_2, \\ \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{v}) &= \boldsymbol{W}\boldsymbol{v} - \boldsymbol{W}\boldsymbol{u}_3. \end{aligned} \tag{53}$$

This is implemented by `apply_Vinv()`: `apply_A_transpose_weighted()` computes $\boldsymbol{u}_1$, `solve_M_lat()` computes $\boldsymbol{u}_2$, `apply_A()` computes $\boldsymbol{u}_3$, and the final diagonal scaling and subtraction compute $\mathcal{V}_{\text{op}}^{-1}(\boldsymbol{v})$. Algebraically, this operator is equivalent to applying $\boldsymbol{V}^{-1}$, but the computation is only a sequence of diagonal scalings, sparse map operations, and solves with the CHOLMOD factor of $\boldsymbol{M}$.

`compute_logDetV()` evaluates

$$\log|\boldsymbol{V}| = \log|\boldsymbol{R}| + \log|\boldsymbol{M}| - \log|\boldsymbol{Q}_w|, \tag{54}$$

without forming $\boldsymbol{V}$. Here $\log|\boldsymbol{R}| = -\sum_i \log p_i$ is computed from the observed-row precision vector, $\log|\boldsymbol{M}|$ is read from the CHOLMOD factor by `logDetFactor()`, and $\log|\boldsymbol{Q}_w|$ is accumulated from the term-specific precision blocks by `logdet_Qw_blocks()`. Because the process prior precision is block diagonal,

$$\boldsymbol{Q}_w = \text{blockdiag}(\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K), \qquad \log|\boldsymbol{Q}_w| = \sum_{k=1}^{K} \log|\boldsymbol{Q}_k|. \tag{55}$$

The term-specific contributions are given in Equations (11), (14), (19), (24), (25), (32), (37), and (40).

The collapsed Gaussian log likelihood is then evaluated, up to constants, as

$$\log p(\boldsymbol{y} \mid \cdot) = -\frac{1}{2}\left\{\log|\boldsymbol{V}| + \boldsymbol{r}^\top\mathcal{V}_{\text{op}}^{-1}(\boldsymbol{r})\right\}. \tag{56}$$

The quadratic contribution is computed by `quadform_Vinv()`, which calls `apply_Vinv()` to obtain $\boldsymbol{z} = \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{r})$ and then returns the dot product $\boldsymbol{r}^\top\boldsymbol{z}$. Thus the collapsed likelihood quadratic is evaluated by applying the needed operators to a vector and taking an inner product, rather than by forming a matrix product or dense inverse. `log_cov_params_target_collapsed()` combines this quadratic term, the determinant contribution, and the prior terms for the current covariance and residual-variance parameters.

## 8 Observed-row likelihood generalization

The backend allows missing response values without letting missing responses contribute likelihood information. This is needed for CAR and DAGAR models with graph nodes or area-time cells that have no observed response, and it is also useful for AR1, GP, NNGP, iid, and fitted or imputation workflows.

This section is a specialization of the collapsed identities in the previous section, not a separate likelihood. The design does not place `NA` values into precision matrices. Instead, the fitted object retains full-row and full-support objects, while the sampler uses an observed-row likelihood view. Let $\mathcal{O}$ be the set of rows with observed response and $\mathcal{M}$ be the rows with missing response. The backend stores

$$\boldsymbol{y}_{\mathcal{O}}, \quad \boldsymbol{X}_{\mathcal{O}}, \quad \boldsymbol{Z}_{\mathcal{O}}, \quad \boldsymbol{A}_{\mathcal{O}}, \tag{57}$$

for sampler updates, while also retaining $\boldsymbol{X}_{\text{full}}$, $\boldsymbol{Z}_{\text{full}}$, full process maps, and full latent supports for fitted values, recovery, prediction, and imputation.

Under the default homoskedastic Gaussian observation model, the collapsed matrix is

$$\boldsymbol{M} = \boldsymbol{Q}_w + \tau^{-2}\boldsymbol{A}_{\mathcal{O}}^{\top}\boldsymbol{A}_{\mathcal{O}}, \tag{58}$$

and the determinant identity uses the number of observed responses:

$$\log|\boldsymbol{V}_{\mathcal{O}}| = n_{\mathcal{O}}\log(\tau^2) + \log|\boldsymbol{M}| - \log|\boldsymbol{Q}_w|. \tag{59}$$

Rows in $\mathcal{M}$ add no likelihood contribution. A latent process node with no observed rows remains in $\boldsymbol{Q}_w$ and can still be estimated through the prior and neighboring latent nodes.

This layer rejects missing fixed-effect covariates, iid grouping IDs, iid slope covariates, process coordinates, and SVC covariates. Only the response is allowed to be missing. The current implementation is organized around a diagonal observed-row precision representation:

$$\boldsymbol{W}_{\mathcal{O}} = \operatorname{diag}(p_i), \qquad \boldsymbol{R}_{\mathcal{O}} = \boldsymbol{W}_{\mathcal{O}}^{-1}, \qquad \boldsymbol{M} = \boldsymbol{Q}_w + \boldsymbol{A}_{\mathcal{O}}^{\top}\boldsymbol{W}_{\mathcal{O}}\boldsymbol{A}_{\mathcal{O}}, \tag{60}$$

with the same factor-based operators defined in the previous section. For any observed-row vector $\boldsymbol{v}_{\mathcal{O}}$,

$$\begin{aligned} \boldsymbol{u}_1 &= \boldsymbol{A}_{\mathcal{O}}^{\top}\boldsymbol{W}_{\mathcal{O}}\boldsymbol{v}_{\mathcal{O}}, \\ \boldsymbol{u}_2 &= \operatorname{solve}_M(\boldsymbol{u}_1), \\ \boldsymbol{u}_3 &= \boldsymbol{A}_{\mathcal{O}}\boldsymbol{u}_2, \\ \mathcal{V}_{\mathcal{O},\text{op}}^{-1}(\boldsymbol{v}_{\mathcal{O}}) &= \boldsymbol{W}_{\mathcal{O}}\boldsymbol{v}_{\mathcal{O}} - \boldsymbol{W}_{\mathcal{O}}\boldsymbol{u}_3, \\ \log|\boldsymbol{V}_{\mathcal{O}}| &= \log|\boldsymbol{R}_{\mathcal{O}}| + \log|\boldsymbol{M}| - \log|\boldsymbol{Q}_w|. \end{aligned} \tag{61}$$

The matrix $\boldsymbol{V}_{\mathcal{O}}$ is not formed; only the action of the observed-row precision operator and its determinant identity are used. This diagonal precision layer supports fixed direct-estimate variances, sampled group-specific residual variances, and low-dimensional scaled direct-estimate variance models. For the homoskedastic Gaussian case, $p_i = \tau^{-2}$ and $\log|\boldsymbol{R}_{\mathcal{O}}| = n_{\mathcal{O}}\log(\tau^2)$.

# 9 Binomial observation model

For `family = "binomial"`, the observed response $y_i$ is the number of successes out of $n_i$ trials. The linear predictor is

$$\eta_i = o_i + \boldsymbol{x}_i^{\top}\boldsymbol{\beta} + \boldsymbol{u}_i^{\top}\boldsymbol{\alpha} + \boldsymbol{a}_i^{\top}\boldsymbol{w}, \tag{62}$$

and

$$y_i \mid \eta_i \sim \text{Binomial}\left(n_i, \frac{\exp(\eta_i)}{1+\exp(\eta_i)}\right). \tag{63}$$

The vector $\boldsymbol{u}_i$ denotes the row of the grouped-random-effect design matrix $\boldsymbol{Z}$; the symbol $\boldsymbol{z}$ is reserved for standard normal random vectors in sampling formulas.

The sampler uses the Pólya-Gamma identity. With

$$\kappa_i = y_i - n_i/2, \tag{64}$$

the augmented likelihood is conditionally Gaussian given

$$\omega_i \mid \eta_i, n_i \sim \mathrm{PG}(n_i, \eta_i). \tag{65}$$

The symbol $\omega_i$ in this section is local to Pólya-Gamma augmentation and is unrelated to the deterministic shrinkage weight $\omega_i = n_i/(n_i + c)$ used in the scaled residual variance model. Equivalently, the working response

$$z_i^* = \kappa_i/\omega_i \tag{66}$$

has conditional mean $\eta_i$ and variance $\omega_i^{-1}$. The implementation keeps the Gaussian working model on the sampled part of the predictor by using $z_i^* - o_i$ as the working response and adding $o_i$ only when refreshing Pólya-Gamma variables, fitted values, and predictions. Thus binomial updates reuse the Gaussian collapsed machinery by setting the observed-row precision to

$$p_i = \omega_i. \tag{67}$$

The Pólya-Gamma variables are refreshed at the start of each MCMC iteration under the binomial family. Because the structured process is collapsed during parameter updates, the binomial sampler keeps a warm-running internal latent process realization for the full linear predictor used in the Pólya-Gamma tilt. After each retained collapsed update, a fresh conditional process draw is generated with the current CHOLMOD factor and carried forward to the next Pólya-Gamma refresh. For structured process models, these in-chain process draws are retained by default as `save_process = list(start = 1, thin = 1)` and form the process sample used by fitted values, prediction, and `recover()`. The retained grid can be thinned at fit time with `save_process = list(start = ..., thin = ...)`. Pólya-Gamma random variates are generated through the hybrid sampler exposed by `BayesLogit`, which dispatches among its available algorithms as a function of the binomial trial count.

`resid()` terms are not used for binomial models. There is no Gaussian residual $\tau^2$ in this likelihood, and posterior predictive response samples are generated from the binomial distribution using the posterior linear-predictor draws and the relevant trial counts.

One practical consideration for binomial logit models is imbalance. When fitted probabilities are close to zero or one for many observations, or when binomial counts are concentrated near zero or near the trial count, standard data-augmentation samplers for logistic models can mix slowly. This issue is not unique to `stLMM` and is discussed for Pólya-Gamma and related data-augmentation samplers in Johndrow et al. (2019) and Zens et al. (2024). Zens et al. (2024) propose "ultimate" Pólya-Gamma samplers that combine latent-utility representations with marginal data augmentation using working location and scale parameters.

## 10 Negative-binomial observation model

The fixed-size negative-binomial family is intended for count responses while preserving the current collapsed Gaussian computational structure. It is not a direct implementation of the classical Poisson log-link model. An exact Poisson log-link likelihood does not have the same Pólya-Gamma quadratic augmentation used by the binomial logit likelihood. A negative-binomial model with a

logit probability parameterization does have this form (Polson et al., 2013; Windle et al., 2013; Zhou et al., 2012).

The current implementation uses a fixed size parameter $r > 0$. With mean $\mu_i$,

$$y_i \mid \mu_i, r \sim \text{NegBin}(r, \mu_i), \qquad \text{E}(y_i) = \mu_i, \qquad \text{Var}(y_i) = \mu_i + \mu_i^2/r. \tag{68}$$

The user-facing linear predictor is on the log-mean scale:

$$\eta_i = \log(\mu_i) = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \boldsymbol{u}_i^\top \boldsymbol{\alpha} + \boldsymbol{a}_i^\top \boldsymbol{w} + o_i, \tag{69}$$

where $o_i$ is an optional offset. For count models with known exposure or expected-count scaling $E_i$, setting $o_i = \log(E_i)$ gives $\mu_i = E_i \exp(\eta_i - o_i)$, so $\exp(\eta_i - o_i)$ is the modeled rate or ratio per unit of $E_i$. The Pólya-Gamma representation uses the logit of the negative-binomial probability parameter,

$$\psi_i = \eta_i - \log(r). \tag{70}$$

Under this parameterization, $\mu_i = r \exp(\psi_i)$ and the likelihood kernel is

$$p(y_i \mid \psi_i, r) \propto \frac{\exp(y_i \psi_i)}{\{1 + \exp(\psi_i)\}^{y_i + r}}. \tag{71}$$

This is the same Pólya-Gamma form as the binomial likelihood, with

$$b_i = y_i + r, \qquad \kappa_i = \frac{y_i - r}{2}, \qquad \omega_i \mid \psi_i, r \sim \text{PG}(y_i + r, \psi_i). \tag{72}$$

Conditional on $\omega_i$, the working response for the log-mean predictor is

$$z_i^* = \kappa_i / \omega_i + \log(r), \tag{73}$$

with conditional mean $\eta_i$ and variance $\omega_i^{-1}$. As in the binomial case, the implementation uses $z_i^* - o_i$ as the Gaussian working response for the sampled part of the predictor and uses the full $\eta_i$ only for the Pólya-Gamma tilt, fitted values, and predictions. Thus the negative-binomial update can reuse the same Gaussian collapsed machinery as the binomial update by setting

$$p_i = \omega_i. \tag{74}$$

The only differences from the binomial working model are the Pólya-Gamma shape $y_i + r$, the value of $\kappa_i$, and the $\log(r)$ offset in $z_i^*$.

This $\log(r)$ adjustment is the main implementation trap when adapting the binomial Pólya-Gamma update: draw the Pólya-Gamma variate using the tilt $\psi_i = \eta_i - \log(r)$, not $\eta_i$, and then add $\log(r)$ back in the working response $z_i^*$. When a formula offset is present, subtract $o_i$ from $z_i^*$ before passing the working response to the collapsed Gaussian machinery.

The initial fixed-$r$ design is deliberate. In spatial and space-time count models, a free negative-binomial dispersion parameter can compete with structured effects and unstructured grouped random effects: extra-Poisson variation can be explained by smaller $r$, by larger process variance, by iid random-effect variance, or by some combination of these. Fixing $r$ avoids adding another weakly identified variance-like component while the count likelihood, fitted values, prediction, and recovery behavior are validated. A large fixed $r$ also provides a Poisson-like model because $\mu_i^2/r$ becomes small and the negative-binomial distribution approaches the Poisson distribution as $r \to \infty$.

There is a computational tradeoff. The Pólya-Gamma draw has shape $y_i + r$. Windle et al. discuss that Pólya-Gamma methods for negative-binomial count models work especially well for small or moderate counts, while large count sizes can make augmentation slower (Windle et al., 2013). Windle, Polson, and Scott also study sampling methods for $\mathrm{PG}(b, z)$ when $b$ is not one, which is the relevant case here (Windle et al., 2014). Consequently, choosing an extremely large $r$ only to mimic a Poisson likelihood can be computationally unattractive. In practice, fixed-$r$ sensitivity checks should use values large enough that $\mu_i^2/r$ is small on the relevant mean scale, but not so large that Pólya-Gamma refreshes dominate runtime without changing inference.

Future versions could allow $r$ to be estimated as a likelihood parameter, not as a covariance or residual variance parameter. One possible route is the Chinese-restaurant-table augmentation used in negative-binomial process and mixed negative-binomial regression work (Zhou and Carin, 2015; Zhou et al., 2012). With a gamma prior $r \sim \mathrm{Gamma}(a_0, b_0)$, using the shape–rate convention, the usual logit-probability parameterization introduces

$$L_i \mid y_i, r \sim \mathrm{CRT}(y_i, r) \tag{75}$$

and gives the Gamma full conditional

$$r \mid \cdot \sim \mathrm{Gamma}\left(a_0 + \sum_i L_i,\ b_0 + \sum_i \log\{1 + \exp(\psi_i)\}\right), \tag{76}$$

equivalently with rate $b_0 - \sum_i \log(1 - p_i)$. This is a Gibbs update rather than a Metropolis update under that parameterization. That extension should have its own likelihood-control grammar, for example a `starting$likelihood` and `priors$likelihood` block, rather than placing $r$ inside process covariance controls. It would also require careful documentation of parameterization: in the log-mean form above, changing $r$ changes $\psi_i = \eta_i - \log(r)$, so the full conditional structure differs from a model that puts the linear predictor directly on $\psi_i = \mathrm{logit}(p_i)$. Sampling $r$ should therefore be treated as a later overdispersed negative-binomial mode, not as the default spatial count model.

# 11 Gaussian Gibbs updates

## 11.1 Update of $\boldsymbol{\beta}$

Conditional on the current $\boldsymbol{\alpha}$, define the partial residual

$$\boldsymbol{y}_\beta = \boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\alpha}. \tag{77}$$

The implementation first applies the collapsed observed-precision operator to the fixed-effect design columns and to this partial residual:

$$\boldsymbol{U}_X = \mathcal{V}_{\mathrm{op}}^{-1}(\boldsymbol{X}), \qquad \boldsymbol{u}_y = \mathcal{V}_{\mathrm{op}}^{-1}(\boldsymbol{y}_\beta). \tag{78}$$

Here $\boldsymbol{U}_X$ is computed by `apply_Vinv_multiple()` and $\boldsymbol{u}_y$ by `apply_Vinv()`; both reuse the current CHOLMOD factor of $\boldsymbol{M}$. The flat-prior fixed-effect precision and right-hand side are then assembled by dense BLAS products:

$$\boldsymbol{Q}_\beta = \boldsymbol{X}^\top \boldsymbol{U}_X, \qquad \boldsymbol{b}_\beta = \boldsymbol{X}^\top \boldsymbol{u}_y. \tag{79}$$

These are the usual fixed-effect full-conditional quantities, but they are computed from operator applications rather than a formed observed covariance or precision matrix. The final draw is

produced by `draw_gaussian_from_precision()`, which factorizes the small dense precision $\boldsymbol{Q}_\beta$, solves for the mean, and adds triangular-solve Gaussian noise. With an independent normal prior

$$\boldsymbol{\beta} \sim N(\boldsymbol{\mu}_\beta, \boldsymbol{D}_\beta), \tag{80}$$

where $\boldsymbol{D}_\beta$ is diagonal, the update is

$$\begin{aligned} \boldsymbol{Q}_\beta &= \boldsymbol{X}^\top \boldsymbol{U}_X + \boldsymbol{D}_\beta^{-1}, \\ \boldsymbol{b}_\beta &= \boldsymbol{X}^\top \boldsymbol{u}_y + \boldsymbol{D}_\beta^{-1} \boldsymbol{\mu}_\beta. \end{aligned} \tag{81}$$

The Gaussian likelihood defaults to the flat-prior form. Pólya-Gamma likelihoods default to a proper independent normal prior with mean zero and standard deviation 10, unless the user supplies `priors$beta`.

This update is implemented in `update_beta_draw_collapsed()`. The expensive step is applying $\mathcal{V}_{\text{op}}^{-1}$ to the columns of $\boldsymbol{X}$ and to $\boldsymbol{y}_\beta$, which reuses the CHOLMOD factorization of $\boldsymbol{M}$.

### 11.2 Update of $\boldsymbol{\alpha}$

Conditional on the current $\boldsymbol{\beta}$, define

$$\boldsymbol{y}_\alpha = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}. \tag{82}$$

The grouped random-effect update applies the same operator to columns of the sparse grouped-design matrix and to the partial residual. Operationally, `form_ZtVinvZ_and_rhs()` scatters one grouped-design column to dense observed-row workspace, applies `apply_Vinv()`, and uses sparse grouped-design dot products to fill

$$\boldsymbol{Q}_{\alpha,0} = \boldsymbol{Z}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{Z}), \qquad \boldsymbol{b}_\alpha = \boldsymbol{Z}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{y}_\alpha). \tag{83}$$

The prior precision for the explicit iid random effects is then added to the diagonal:

$$\boldsymbol{Q}_\alpha = \boldsymbol{Q}_{\alpha,0} + \boldsymbol{D}_\alpha^{-1}. \tag{84}$$

These are the usual grouped-random-effect full-conditional quantities, but they are computed from operator applications rather than a formed observed covariance or precision matrix. The final draw uses `draw_gaussian_from_precision()` with the small dense precision $\boldsymbol{Q}_\alpha$ and right-hand side $\boldsymbol{b}_\alpha$. This step is implemented in `update_alpha_draw_collapsed()`.

### 11.3 Update of explicit random-effect variances

For iid random-effect block $b$, $\sigma_{\alpha,b}^2$ is the variance used in the corresponding diagonal block of $\boldsymbol{D}_\alpha$. The inverse-gamma prior parameters $a_b$ and $b_b$ are supplied through `priors`. Given the current random effects in block $b$, the block variance update is Gibbs with an inverse-gamma full conditional. For block size $m_b$ and

$$s_b = \boldsymbol{\alpha}_b^\top \boldsymbol{\alpha}_b, \tag{85}$$

the update is

$$\begin{aligned} a_b^* &= a_b + \frac{m_b}{2}, \\ b_b^* &= b_b + \frac{s_b}{2}, \\ \sigma_{\alpha,b}^2 \mid \cdot &\sim \text{IG}(a_b^*, b_b^*). \end{aligned} \tag{86}$$

This is implemented by `update_sigmaSqRE_gibbs()`.

## 12 Adaptive covariance Metropolis blocks

The current sampler updates covariance and sampled residual variance parameters with adaptive random-walk Metropolis steps on transformed scales. The adaptive Metropolis strategy draws on unpublished adaptive Metropolis notes by Benjamin Shaby and on the broader adaptive MCMC literature (Andrieu and Thoms, 2008; Roberts and Rosenthal, 2009, 2007).

We use $\boldsymbol{\theta}_k$ for the covariance-function parameters of structured process term $k$, for example a range, decay, correlation, smoothness, or nonseparable mixing parameter. The broader Metropolis-updated parameter collection is denoted $\boldsymbol{\psi}$ and can include scalar residual variance parameters, process variances, and $\boldsymbol{\theta}_k$ values. The default is still one joint block over all entries of $\boldsymbol{\psi}$, but the `metropolis` control list can choose a different blocking rule without changing the model being fit.

On the model scale, the parameter collection can include:

- $\tau^2$ under the default Gaussian residual model,
- sampled residual variances under direct-estimate residual models,
- each process variance $\sigma_k^2$,
- each covariance-function parameter $\theta_{k,j}$.

The transformed adaptive Metropolis vector is

$$\boldsymbol{\eta} = h(\boldsymbol{\psi}), \tag{87}$$

where $h(\cdot)$ applies log transformations to positive variance parameters and bounded-logit transformations to bounded covariance-function parameters. Thus the transformed entries can include:

- $\log(\tau^2)$,
- each sampled residual variance on the log scale,
- each $\log(\sigma_k^2)$,
- each bounded-logit transformed process parameter

$$\eta_{k,j} = \log\left(\frac{\theta_{k,j} - a_{k,j}}{b_{k,j} - \theta_{k,j}}\right). \tag{88}$$

Parameters in the starting list specified via `fixed()` are excluded from the Metropolis updates. If all covariance and sampled residual variance tuning entries are zero, the Metropolis step is skipped.

The implemented blocking choices are:

- `"joint"`: all covariance and sampled residual variance parameters in one block;
- `"by_term"`: residual variance parameters in one block and each process term's variance plus covariance parameters in a separate block;
- `"residual_process"`: residual variance parameters in one block and all process covariance parameters in another;

- `"variance_theta"`: variance parameters in one block and range/correlation/covariance-shape parameters in another;
- `"process_variance"`: residual variance, process variance, and covariance-function theta parameters in separate blocks;
- `"scalar"`: one parameter per block, using scalar random-walk adaptation.

The default `metropolis = list(blocking = "joint")` preserves the original behavior. A single string such as `metropolis = "by_term"` is also accepted. Retained-sampling adaptation is controlled by `metropolis$target_accept` and `metropolis$batch_length`. The default target acceptance is 0.234 for block updates and 0.44 for scalar updates; the default retained-adaptation batch length is 25.

Before retained sampling, the default behavior is to run a discarded covariance warmup phase controlled by `warmup`. The default `warmup = TRUE` runs a short proposal-scale calibration for Metropolis blocks; `warmup = FALSE` disables it. Warmup iterations are not included in the returned posterior sample matrices, so `n_samples` always denotes post-warmup retained samples.

The warmup phase uses the same Gibbs and covariance Metropolis update as the main sampler, but stores no posterior samples. It adjusts proposal scales from batch acceptance rates and stops when the blocks fall inside the target interval after any required minimum batches, or when the maximum number of warmup batches is reached.

The retained update is implemented through `update_cov_params_mh_collapsed()`, which loops over proposal blocks. For a multivariate block $b$, proposals have covariance $\lambda_b \boldsymbol{\Sigma}_b$, where $\boldsymbol{\Sigma}_b$ is the adaptive empirical covariance shape and $\lambda_b$ is a variance-like adaptive multiplier. For scalar blocks, the proposal variance is $\delta_b^2 \Sigma_b$. Here $\Sigma_b > 0$ is the scalar proposal shape, including any scale folded in from the warmup phase, and $\delta_b$ is a one-dimensional random-walk scale. Multivariate blocks adapt an empirical covariance shape and a multiplier on that covariance, while scalar blocks adapt a direct step size. Proposals are evaluated through a scratch covariance state and the live covariance state is changed only on acceptance.

For a current residual $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Z}\boldsymbol{\alpha}$, the collapsed target contribution for covariance and sampled residual variance parameters is

$$\ell(\boldsymbol{\psi}) = -\frac{1}{2}\left\{\log|\boldsymbol{V}(\boldsymbol{\psi})| + \boldsymbol{r}^\top \mathcal{V}_{\mathrm{op},\boldsymbol{\psi}}^{-1}(\boldsymbol{r})\right\} + \log p(\boldsymbol{\psi}) + \log|J_h(\boldsymbol{\psi})|, \tag{89}$$

where the Jacobian term is included for transformed Metropolis parameters. The first two terms are the collapsed likelihood in Equation (56), evaluated at the proposed covariance and residual-precision values. Operationally, each proposal assembles a candidate $\boldsymbol{M}$, refreshes its CHOLMOD factor, computes $\log|\boldsymbol{V}|$ with Equation (54), computes the quadratic form by applying $\mathcal{V}_{\mathrm{op},\boldsymbol{\psi}}^{-1}$ to $\boldsymbol{r}$, and then accepts or rejects the proposal. No dense $\boldsymbol{V}$, $\boldsymbol{V}^{-1}$, $\boldsymbol{M}^{-1}$, or $\boldsymbol{Q}_w^{-1}$ is formed.

The fitted object reports:

- `covariance_acceptance`: aggregate acceptance across covariance and sampled residual variance proposals,
- `adaptive_metropolis`: transformed parameter labels, current transformed values, initial covariance, final proposal covariance, batch acceptance history, proposal scale history, global acceptance, and a nested `warmup` list,

- `adaptive_metropolis$blocks`: one entry per retained Metropolis block, including block labels, block dimension, acceptance, proposal covariance or scalar scale history, and block-specific warmup diagnostics.

The nested `adaptive_metropolis$warmup` list records:

- whether warmup was enabled,
- warmup batch length, maximum batches, target interval, and stop reason,
- number of attempted and accepted warmup proposals,
- warmup batch acceptance history and proposal-scale history,
- starting and ending transformed covariance parameters,
- starting and ending proposal covariance matrices.

This metadata is intentionally separate from posterior samples. It makes automatic tuning auditable without confusing warmup draws with posterior draws. Diagnostics that intentionally set tuning values to zero may show flat covariance chains or no covariance acceptance; that is expected because those parameters are fixed and the corresponding blocks are omitted.

Blocking changes the proposal partition, not the posterior distribution. It can improve a difficult fit when one large joint proposal is poorly scaled, but it can also be slower because every block proposal repeats the collapsed likelihood calculation and sparse factorization work. The scalar option is mainly a diagnostic and robustness option; it is not expected to be the fastest choice for expensive process models. There is no Gibbs update for sampled residual variances in the collapsed likelihood; they are Metropolis parameters because the structured process has been marginalized.

## 13 Collapsed sampler algorithm

The main routine is `stLMM_collapsed_sampler()`. After initialization, the sampler builds the symbolic sparsity pattern of $\boldsymbol{M}$, performs CHOLMOD symbolic analysis, assembles the initial numeric values, and factorizes $\boldsymbol{M}$. It then runs the discarded covariance warmup phase when enabled and when there is at least one covariance or sampled residual variance parameter. Ignoring storage and reporting, each retained post-warmup iteration follows Algorithm 1.

**Algorithm 1: one retained collapsed-sampler iteration.**

1. If the likelihood is Pólya-Gamma, refresh the augmentation variables, working response, observed-row precision $\boldsymbol{W}$, collapsed matrix $\boldsymbol{M}$, and CHOLMOD factor using `update_pg_working_model()`.
2. Sample $\boldsymbol{\beta}$ from its Gaussian full conditional using $\boldsymbol{X}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{X})$ and $\boldsymbol{X}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\alpha})$.
3. Sample $\boldsymbol{\alpha}$ from its Gaussian full conditional using $\boldsymbol{Z}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{Z})$ and $\boldsymbol{Z}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$.
4. Sample explicit iid random-effect variances with `update_sigmaSqRE_gibbs()`.
5. Form the running residual $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Z}\boldsymbol{\alpha}$ for the covariance-parameter update.
6. For each adaptive Metropolis block, propose covariance or sampled residual variance parameters, assemble and factor the candidate $\boldsymbol{M}$, evaluate Equation (89), and accept or reject the proposal.
7. If the likelihood is Pólya-Gamma and structured process terms are present, refresh $\boldsymbol{r}$ and draw the current latent process from its conditional distribution. This draw is always made to seed the next Pólya-Gamma refresh; it is stored only on the `save_process` grid.

Step 1 is skipped for Gaussian likelihoods. Under Pólya-Gamma likelihoods, currently `family = "binomial"` and `family = "negative_binomial"` with fixed size, the iteration begins by refreshing the Pólya-Gamma variables using the current warm-running latent process draw in the full linear predictor. After the augmentation is refreshed, the sampler uses the same collapsed coefficient and covariance update machinery with $\boldsymbol{W} = \text{diag}(\omega_i)$. At the end of the iteration, for structured process models, the sampler draws a fresh conditional $\boldsymbol{w}$ using the current factorized $\boldsymbol{M}$ and carries it forward to the next Pólya-Gamma refresh. These internal process draws are retained by default for Pólya-Gamma process models, avoiding a separate post-fit Pólya-Gamma/process reconstruction chain.

## 14 Latent process recovery and fitted values

For Gaussian-likelihood models with structured process terms, recovery is a standalone post-fit step using `recover()`. For each selected posterior draw, recovery samples

$$\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{W}, \{\sigma_k^2\}, \boldsymbol{\theta} \tag{90}$$

using the stored backend, current posterior parameter draw, sparse $\boldsymbol{M}$ assembly, and CHOLMOD factorization. The recovered conditional precision is

$$\boldsymbol{M}_{\text{rec}} = \boldsymbol{Q}_w + \boldsymbol{A}^\top \boldsymbol{W} \boldsymbol{A}. \tag{91}$$

This is the same sparse matrix $\boldsymbol{M}$ defined in the collapsed sampler, refactored with the covariance and residual-precision values for the selected posterior draw; recovery does not introduce a separate computational matrix.

Under the observed-row likelihood layer, recovery samples latent process draws on each term's full latent support. Only observed rows contribute to the recovery conditional:

$$\text{Prec}(\boldsymbol{w} \mid \cdot) = \boldsymbol{Q}_w + \boldsymbol{A}_{\mathcal{O}}^\top \boldsymbol{W}_{\mathcal{O}} \boldsymbol{A}_{\mathcal{O}}, \tag{92}$$

and the conditional mean uses observed-row residual information only:

$$\boldsymbol{b}_w = \boldsymbol{A}_{\mathcal{O}}^{\top}\boldsymbol{W}_{\mathcal{O}}\left(\boldsymbol{y}_{\mathcal{O}} - \boldsymbol{X}_{\mathcal{O}}\boldsymbol{\beta} - \boldsymbol{Z}_{\mathcal{O}}\boldsymbol{\alpha}\right). \tag{93}$$

The conditional mean is obtained by solving with the recovered conditional precision; $\boldsymbol{Q}_w^{-1}$ is never formed explicitly. Rows with missing responses do not enter $\boldsymbol{b}_w$, but their mapped latent nodes remain in $\boldsymbol{w}$. Thus missing-response AR1, GP, NNGP, CAR, DAGAR, CAR-time, and DAGAR-time nodes are recovered at `recover()` time whenever they are part of the fitted term support. Their posterior information comes from the prior precision and any observed neighboring or repeated nodes, not from an imputed response value.

For Pólya-Gamma process models, `stLMM()` saves the in-chain process draws by default as `save_process = list(start = 1, thin = 1)`. In this case `recover()` is a selection and labeling step: it subsets the saved process draws according to `sub_sample$start` and `sub_sample$thin`, preserves the corresponding `recover_iter` indices, and returns the usual `stLMM_recovery` object. The main posterior parameter matrices are not physically thinned when process draws are saved; downstream methods align parameter rows to `recover_iter`. If a Pólya-Gamma process model was fit with process saving disabled, post-fit recovery is not available and the model should be refit with `save_process = TRUE` or an explicit `save_process = list(start = ..., thin = ...)`. Enabling `save_process` for Gaussian-likelihood process models is disallowed, because drawing and storing $\boldsymbol{w}$ in the main Gaussian sampler would undo the collapsed-sampler design.

Stochastic recovery uses the CHOLMOD factorization of the recovered conditional precision:

$$\boldsymbol{w} = \boldsymbol{P}^{\top}\boldsymbol{L}^{-\top}\boldsymbol{z} + \mathrm{E}(\boldsymbol{w} \mid \cdot), \qquad \boldsymbol{z} \sim N(0, \boldsymbol{I}). \tag{94}$$

Here $\boldsymbol{P}$ is the fill-reducing permutation and $\boldsymbol{L}$ is the triangular factor. This draws from $N\{\mathrm{E}(\boldsymbol{w} \mid \cdot), \boldsymbol{M}_{\mathrm{rec}}^{-1}\}$ without forming $\boldsymbol{M}_{\mathrm{rec}}^{-1}$.

**Algorithm 2: latent process recovery for one posterior draw.**

1. Set the posterior draw's covariance and residual-precision parameters in the backend state and assemble $\boldsymbol{M}_{\mathrm{rec}}$ in Equation (91).
2. Factor $\boldsymbol{M}_{\mathrm{rec}}$ with CHOLMOD, conceptually $\boldsymbol{P}\boldsymbol{M}_{\mathrm{rec}}\boldsymbol{P}^{\top} = \boldsymbol{L}\boldsymbol{L}^{\top}$.
3. Form $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Z}\boldsymbol{\alpha}$ and $\boldsymbol{b}_w = \boldsymbol{A}^{\top}\boldsymbol{W}\boldsymbol{r}$, using observed rows only.
4. Compute the conditional mean $\bar{\boldsymbol{w}}$ by solving $\boldsymbol{M}_{\mathrm{rec}}\bar{\boldsymbol{w}} = \boldsymbol{b}_w$ with `M_cholmod_solve(CHOLMOD_A, ...)`.
5. Draw $\boldsymbol{z} \sim N(0, \boldsymbol{I})$ and compute $\boldsymbol{\varepsilon} = \boldsymbol{P}^{\top}\boldsymbol{L}^{-\top}\boldsymbol{z}$ with `M_cholmod_solve(CHOLMOD_Lt, ...)` followed by `M_cholmod_solve(CHOLMOD_Pt, ...)`.
6. Return $\boldsymbol{w} = \bar{\boldsymbol{w}} + \boldsymbol{\varepsilon}$.

These steps are implemented in `recover_w_draw_collapsed()`. For Pólya-Gamma process fits with saved in-chain process draws, `recover()` subsets and labels the saved draws instead of re-running Algorithm 2.

Recovered process samples are returned by term. For NNGP terms, `w_samples` is in original support order for users, while `w_samples_ordered` keeps the internal graph order needed by fitted values and prediction. `recover_iter` records the original MCMC iteration indices, and prediction/fitted methods use it to align recovered latent draws with parameter samples.

`fitted.stLMM()` supports fit objects without process terms directly. For models with process terms, fitted values require a recovered object and include fixed effects, explicit grouped random effects, and recovered structured process contributions.

With missing responses, fitted values are full-row quantities. Observed rows report fitted values for observed data, while missing-response rows report posterior mean quantities using the retained full-row design and the recovered full-support latent process draws.

## 15 Prediction design and implementation scope

Prediction is a post-fit layer. It reuses the fitted backend, posterior samples, and recovered latent process draws rather than adding prediction to the MCMC sampler.

The current prediction API is:

```
predict.stLMM(object, newdata = NULL, y_samples = FALSE,
              sub_sample = list(start = 1L, thin = 1L),
              scale = c("response", "link"), ...)

predict.stLMM_recovery(object, newdata = NULL, y_samples = FALSE,
                       joint = FALSE,
                       joint_method = c("full", "vecchia"),
                       pred_m = NULL, pred_ordering = "maxmin",
                       st_scale = 1, return_w_samples = TRUE,
                       sub_sample = list(start = 1L, thin = 1L),
                       scale = c("response", "link"), ...)
```

The currently implemented behavior is:

- fit objects can be predicted directly only when the model has no structured process terms;
- process models require saved or recovered latent process samples;
- `sub_sample` selects posterior draw indices for `predict.stLMM()` and recovered draw indices through `recover_iter` for `predict.stLMM_recovery()`;
- prediction returns posterior sample matrices, not summaries;
- `mu_samples` is always returned;
- `y_samples` is returned only when `y_samples = TRUE`;
- `w_samples` is returned only by recovered-process prediction when `return_w_samples = TRUE`.

`joint`, `joint_method`, `pred_m`, `pred_ordering`, and `st_scale` are arguments to `predict.stLMM_recovery()` only, because new process-node prediction requires saved or recovered latent process draws. Formula offsets are evaluated in `newdata` when present and are included in both link-scale and response-scale predictions. For binomial fits, `scale = "response"` returns probabilities for fitted means, while `scale = "link"` returns the linear predictor. Posterior predictive `y_samples` are binomial draws using fitted or prediction trial counts. For the fixed-size negative-binomial family, `scale =`

`"response"` returns $\exp(\eta_i)$ and posterior predictive `y_samples` are negative-binomial draws using the fixed fitted size.

For process terms, prediction rows are shared but process supports are term-specific. The prediction backend maps each prediction row separately for each fitted process term, preserving that term's graph, coordinate order, support, ordering, and SVC covariate scale. This is essential for models such as

```
y ~ x +
    nngp(lon, lat, time, cov_model = "sep_exp") +
    x:nngp(lon, lat, time, cov_model = "sep_exp") +
    ar1(day)
```

where the prediction rows are common but the latent supports and row-to-node maps are term-specific.

Implemented process prediction includes:

- fitted-data prediction with `newdata = NULL`;
- `newdata` whose grouped random-effect levels and process coordinates already exist in the fitted support;
- new AR1 nodes for `joint = FALSE`;
- new dense GP nodes for `joint = FALSE` and exact dense GP joint prediction for `joint = TRUE`;
- new NNGP nodes for `joint = FALSE`, moderate-size `joint = TRUE, joint_method = "full"`, and scalable `joint = TRUE, joint_method = "vecchia"`;
- existing-area CAR, DAGAR, CAR-time, and DAGAR-time prediction;
- repeated prediction rows sharing a new latent node reuse the same simulated latent draw within each posterior sample;
- SVC prediction contributions are multiplied by the prediction-row covariate value;
- NNGP `joint = FALSE` new-node simulation is implemented in compiled code.

## 15.1 Prediction algorithms

Prediction is assembled draw-by-draw on the linear predictor scale. For posterior draw $b$ and prediction row $i$,

$$\mu_i^{(b)} = o_i + \boldsymbol{x}_i^\top \boldsymbol{\beta}^{(b)} + \boldsymbol{u}_i^\top \boldsymbol{\alpha}^{(b)} + \sum_k a_{ik} w_k^{(b)}, \tag{95}$$

where the final sum is over fitted process terms. For ordinary process terms, $a_{ik}$ is one for the mapped latent node and zero otherwise. For an SVC term, $a_{ik}$ includes the prediction-row covariate value, so the contribution is $x_i^{\text{svc}} w_k^{(b)}$. Here $\boldsymbol{u}_i$ is the prediction-row random-effect design vector, i.e., the corresponding row of $\boldsymbol{Z}_0$. For Gaussian fits, if `y_samples = TRUE`, posterior predictive responses are generated as

$$y_i^{(b)} \sim N(\mu_i^{(b)}, \tau_i^{2(b)}). \tag{96}$$

Under the default residual model, $\tau_i^{2(b)} = \tau^{2(b)}$ for all rows. For fixed, group-specific, or scaled residual variance models, the prediction code uses the fitted residual model to obtain the row-specific predictive standard deviation whenever posterior predictive response samples are requested.

For binomial fits, $\mu_i^{(b)}$ denotes the linear predictor when `scale = "link"`. On the response scale,

$$\pi_i^{(b)} = \frac{\exp(\mu_i^{(b)})}{1 + \exp(\mu_i^{(b)})}. \tag{97}$$

If `y_samples = TRUE`, posterior predictive responses are generated as

$$y_i^{(b)} \sim \text{Binomial}(n_i, \pi_i^{(b)}), \tag{98}$$

where $n_i$ is the fitted or prediction trial count. With `newdata`, a column named `trials` is used when present; otherwise one trial per row is assumed.

For the fixed-size negative-binomial family, $\mu_i^{(b)}$ denotes the linear predictor when `scale = "link"`. On the response scale,

$$\lambda_i^{(b)} = \exp(\mu_i^{(b)}). \tag{99}$$

If `y_samples = TRUE`, posterior predictive responses are generated as

$$y_i^{(b)} \sim \text{NegBin}(r, \lambda_i^{(b)}), \tag{100}$$

where $r$ is the fixed fitted size parameter and the second argument is the mean.

For models without structured process terms, `predict.stLMM()` uses the posterior samples stored in the fit object. It rebuilds $\boldsymbol{X}_0$ and $\boldsymbol{Z}_0$ for `newdata` and returns

$$\boldsymbol{\mu}_0^{(b)} = \boldsymbol{o}_0 + \boldsymbol{X}_0\boldsymbol{\beta}^{(b)} + \boldsymbol{Z}_0\boldsymbol{\alpha}^{(b)}. \tag{101}$$

For `newdata = NULL`, the retained full-row design is used, including rows whose response was missing during fitting.

For process models, `predict.stLMM_recovery()` first maps `sub_sample` through `recover_iter`; this keeps $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$, $\tau^2$, covariance parameters, and recovered $\boldsymbol{w}$ aligned to the same original MCMC iterations. Existing process nodes are direct lookups from recovered `w_samples`. Repeated prediction rows that map to the same process node share the same latent draw within a posterior sample.

**AR1 new-node prediction.** For a new AR1 support value under `joint = FALSE`, the prediction code uses the fitted ordered-support AR1 convention. A new value is inserted into the sorted unique support only for identifying its nearest fitted neighbors. The conditional distribution is derived from the AR1 covariance implied by that ordered support. If the new value has one fitted neighbor, the conditional mean is proportional to that neighbor's recovered draw. If it lies between two fitted neighbors, the conditional mean uses both adjacent fitted neighbors and the conditional variance is the corresponding scalar Schur complement. This is adjacency in ordered support, not a numeric time-gap power rule. Consequently, for unevenly spaced time values, $\phi$ should be read as a correlation per adjacent support position rather than as a correlation per unit of elapsed time.

**Dense GP new-node prediction.** For a dense GP term, let $\boldsymbol{C}_{oo}^{(b)}$ be the fitted-support covariance for draw $b$. For a unique new node $s_0$, let $\boldsymbol{c}_{0o}^{(b)}$ be its covariance vector with the fitted support and let $c_{00}^{(b)}$ be its marginal variance. Under `joint = FALSE`, each unique new node is simulated from

$$w(s_0)^{(b)} \mid \boldsymbol{w}_o^{(b)} \sim N\left\{\boldsymbol{c}_{0o}^{(b)}(\boldsymbol{C}_{oo}^{(b)})^{-1}\boldsymbol{w}_o^{(b)}, c_{00}^{(b)} - \boldsymbol{c}_{0o}^{(b)}(\boldsymbol{C}_{oo}^{(b)})^{-1}\boldsymbol{c}_{o0}^{(b)}\right\}. \tag{102}$$

Repeated rows sharing the same new coordinates reuse the simulated node draw. Under `joint = TRUE`, let $\boldsymbol{C}_{no}^{(b)}$ be the covariance between all unique new nodes and the fitted support, and let $\boldsymbol{C}_{nn}^{(b)}$ be the covariance among the unique new nodes. The unique new-node vector is drawn exactly from

$$\boldsymbol{w}_n^{(b)} \mid \boldsymbol{w}_o^{(b)} \sim N\left[\boldsymbol{C}_{no}^{(b)}\{\boldsymbol{C}_{oo}^{(b)}\}^{-1}\boldsymbol{w}_o^{(b)}, \boldsymbol{C}_{nn}^{(b)} - \boldsymbol{C}_{no}^{(b)}\{\boldsymbol{C}_{oo}^{(b)}\}^{-1}\boldsymbol{C}_{on}^{(b)}\right]. \tag{103}$$

This is exact conditional GP prediction but dense in the number of unique new nodes, so it is intended for small prediction sets.

**NNGP `joint = FALSE` new-node prediction.** For an NNGP term, each unique new coordinate $s_0$ chooses up to $m$ nearest neighbors from the fitted support only. For space-time NNGP terms, `st_scale` is applied only while ranking neighbors by distance: the final coordinate is multiplied by `st_scale`, while the spatial coordinates remain on their original scale. This is not automatic standardization of space and time. If spatial coordinates are kilometers and time is years, `st_scale = 50` means one year contributes like about 50 kilometers in the neighbor-search distance. All covariance calculations use the original coordinates after the neighbor set is selected. Let $N(s_0)$ be the selected fitted-neighbor set. For posterior draw $b$,

$$\boldsymbol{B}_0^{(b)} = \boldsymbol{c}_{0N}^{(b)}\{\boldsymbol{C}_{NN}^{(b)}\}^{-1}, \qquad V_0^{(b)} = C_{00}^{(b)} - \boldsymbol{B}_0^{(b)}\boldsymbol{c}_{N0}^{(b)}. \tag{104}$$

Then

$$w(s_0)^{(b)} \mid \boldsymbol{w}_N^{(b)} \sim N\{\boldsymbol{B}_0^{(b)}\boldsymbol{w}_N^{(b)}, V_0^{(b)}\}. \tag{105}$$

The current implementation builds neighbor indices once and uses a compiled kernel for the repeated draw-wise conditional calculations.

**NNGP `joint = TRUE, joint_method = "full"` new-node prediction.** The `joint_method = "full"` NNGP uses the same fitted-support neighbor sets and the same conditional means as `joint = FALSE`, but simulates the unique new nodes jointly. For unique new nodes $s_i$ and $s_j$, define the NNGP residual

$$e_i = w(s_i) - \boldsymbol{B}_i\boldsymbol{w}_{N_i}. \tag{106}$$

The residual covariance used for joint simulation is

$$\mathrm{Cov}(e_i, e_j) = C_{ij} - \boldsymbol{B}_i\boldsymbol{C}_{N_i j} - \boldsymbol{C}_{iN_j}\boldsymbol{B}_j^\top + \boldsymbol{B}_i\boldsymbol{C}_{N_iN_j}\boldsymbol{B}_j^\top. \tag{107}$$

For each posterior draw, the implementation builds this dense residual covariance over the unique new prediction nodes, adds the conditional mean vector $\boldsymbol{B}_i\boldsymbol{w}_{N_i}^{(b)}$, and draws the joint residual with a Cholesky factorization. This changes cross-location predictive dependence while preserving the same marginal conditional means as `joint = FALSE`. The fitted-support parent sets and conditional means remain the NNGP approximation, while the joint new-node residual covariance uses full GP covariance evaluations among the prediction nodes and their fitted parents. Because this residual covariance is dense, this is a moderate-size prediction feature.

**NNGP `joint = TRUE, joint_method = "vecchia"` new-node prediction.** The `joint_method = "vecchia"` is the scalable joint prediction option. Unique new prediction nodes are ordered by `pred_ordering`. For each ordered prediction node, neighbors are selected from the combined history made of all fitted support nodes plus earlier prediction nodes in that ordering. The optional `pred_m` can differ from the fitted model's `m`; if `pred_m = NULL`, the fitted `m` is used. The draw is then generated sequentially, so earlier simulated prediction nodes can enter the conditional mean of later prediction nodes. This is still an approximation and is ordering dependent, but it avoids the dense residual covariance required by `joint_method = "full"`.

**CAR, DAGAR, CAR-time, and DAGAR-time prediction.** CAR prediction maps prediction rows to areas already present in the fitted CAR graph and looks up the recovered latent draw for that area. DAGAR prediction does the same lookup after mapping the area through the fitted DAGAR ordering. CAR-time prediction maps rows to existing fitted area-time nodes in location-major support. DAGAR-time prediction maps rows to existing fitted area-time nodes in ordered-area-major support. New CAR or DAGAR areas, new CAR-time or DAGAR-time areas, and new CAR-time or DAGAR-time values currently error. This keeps areal prediction faithful to the graph and time support used for fitting.

With the observed-row likelihood layer, `newdata = NULL` uses the retained full-row design, not only the observed-row likelihood view. For rows whose response was missing during fitting, prediction uses the same retained fixed-effect, iid, and process maps plus recovered full-support latent draws. In that sense, recovered AR1, GP, NNGP, CAR, DAGAR, CAR-time, and DAGAR-time effects for missing-response support rows become the process contribution used by fitted values and prediction on those rows. Missing responses should be generated only as posterior predictive or imputation quantities; they should not feed back into sampler updates.

CAR and DAGAR prediction currently require areas that exist in the fitted graph. CAR-time and DAGAR-time prediction require both fitted graph areas and fitted time values. New areas and new areal time values error until a model-based policy for extending areal supports is designed.

For space-time NNGP prediction, `st_scale` affects nearest-neighbor ranking only. It multiplies the final coordinate before neighbor search; the spatial coordinates are not rescaled, and the covariance calculation still uses the original coordinates. Users should choose `st_scale` to balance spatial and temporal neighbor extents in the units used by their data.

For large NNGP prediction sets, `joint = FALSE` is the simplest and fastest option when cross-prediction dependence is not needed, while `joint_method = "vecchia"` is the scalable option when it is.

Large prediction remains a simple call over user-supplied `newdata` and selected posterior draws. Users control prediction size through `newdata` and `sub_sample`.

# 16 Implementation function map

Table 1 summarizes the principal computational operators in the current collapsed sampler. It is intended as an implementation reference for the equations and algorithms above.

Table 1: Main computational functions in the current collapsed sampler.

| Function | Role |
|---|---|
| `build_M_lat_pattern_sparse()` | Build symbolic lower-triangular sparsity pattern of $\boldsymbol{M}$. |
| `assemble_M_lat_numeric()` | Fill numerical entries of $\boldsymbol{M} = \boldsymbol{Q}_w + \boldsymbol{A}^\top \boldsymbol{W} \boldsymbol{A}$. |
| `assemble_nngp_block_into_M()` | Add one NNGP prior precision block into $\boldsymbol{M}$. |
| `assemble_gp_block_into_M()` | Add one dense GP prior precision block into $\boldsymbol{M}$. |
| `assemble_ar1_block_into_M()` | Add one AR1 prior precision block into $\boldsymbol{M}$. |
| `assemble_car_block_into_M()` | Add one sparse spatial CAR prior precision block into $\boldsymbol{M}$. |
| `assemble_dagar_block_into_M()` | Add one sparse ordered DAGAR prior precision block into $\boldsymbol{M}$. |
| `assemble_car_time_block_into_M()` | Add one sparse separable CAR-time prior precision block into $\boldsymbol{M}$. |
| `assemble_dagar_time_block_into_M()` | Add one sparse separable DAGAR-time prior precision block into $\boldsymbol{M}$. |
| `refresh_observation_precision()` | Refresh the observed-row precision vector from scalar, fixed, group-specific, scaled residual variance, or Pólya-Gamma working precision values. |
| `add_observation_precision_AtA_into_M()` | Add the $\boldsymbol{A}^\top \boldsymbol{W} \boldsymbol{A}$ data contribution. |
| `solve_M_lat()` | Apply $\mathrm{solve}_M(\cdot)$ to one right-hand side using the current CHOLMOD factor of $\boldsymbol{M}$. |
| `solve_M_lat_multiple()` | Apply $\mathrm{solve}_M(\cdot)$ to multiple right-hand sides using the current CHOLMOD factor of $\boldsymbol{M}$. |
| `apply_A_transpose()` | Compute $\boldsymbol{A}^\top \boldsymbol{v}$ for a single observation-space vector. |
| `apply_A_transpose_weighted()` | Compute $\boldsymbol{A}^\top \boldsymbol{W} \boldsymbol{v}$ for a single observation-space vector. |
| `apply_A()` | Compute $\boldsymbol{A}\boldsymbol{v}$ for a single latent vector. |
| `apply_A_transpose_multiple()` | Apply $\boldsymbol{A}^\top$ to multiple columns. |
| `apply_A_transpose_weighted_multiple()` | Apply $\boldsymbol{A}^\top \boldsymbol{W}$ to multiple columns. |
| `apply_A_multiple()` | Apply $\boldsymbol{A}$ to multiple columns. |
| `apply_Vinv()` | Apply $\mathcal{V}_{\mathrm{op}}^{-1}$ to a single vector. |
| `apply_Vinv_multiple()` | Apply $\mathcal{V}_{\mathrm{op}}^{-1}$ to multiple columns. |
| `form_XtVinvX_and_rhs()` | Form $\boldsymbol{X}^\top \mathcal{V}_{\mathrm{op}}^{-1}(\boldsymbol{X})$ and $\boldsymbol{X}^\top \mathcal{V}_{\mathrm{op}}^{-1}(\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\alpha})$ from operator applications and dense coefficient-space products. |
| `form_ZtVinvZ_and_rhs()` | Form $\boldsymbol{Z}^\top \mathcal{V}_{\mathrm{op}}^{-1}(\boldsymbol{Z})$ and $\boldsymbol{Z}^\top \mathcal{V}_{\mathrm{op}}^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$ from operator applications and sparse grouped-design dot products. |
| `compute_logDetV()` | Evaluate $\log \lvert \boldsymbol{V} \rvert$ using Equation (54). |

| Function | Role |
|---|---|
| `logdet_Qw_blocks()` | Evaluate prior precision log determinants for NNGP, GP, AR1, CAR, DAGAR, CAR-time, and DAGAR-time blocks. |
| `quadform_Vinv()` | Evaluate $\boldsymbol{r}^\top \mathcal{V}_{\text{op}}^{-1}(\boldsymbol{r})$ as an operator application followed by a dot product. |
| `recover_w_draw_collapsed()` | Draw $\boldsymbol{w}$ from the conditional latent-process distribution using Algorithm 2. |

## References


Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2014). *Hierarchical Modeling and Analysis for Spatial Data*. CRC Press, Boca Raton, FL, second edition.

Datta, A., Banerjee, S., Hodges, J. S., and Gao, L. (2019). Spatial disease mapping using directed acyclic graph auto-regressive (DAGAR) models. *Bayesian Analysis*, 14(4):1221–1244.

Finley, A. O., Datta, A., Cook, B. D., Morton, D. C., Andersen, H.-E., and Banerjee, S. (2019). Efficient algorithms for bayesian nearest neighbor gaussian processes. *Journal of Computational and Graphical Statistics*, 28(2):401–414.

Gneiting, T. (2002). Nonseparable, stationary covariance functions for space-time data. *Journal of the American Statistical Association*, 97(458):590–600.

Johndrow, J. E., Smith, A., Pillai, N., and Dunson, D. B. (2019). MCMC for imbalanced categorical data. *Journal of the American Statistical Association*, 114(527):1394–1403.

Leroux, B. G., Lei, X., and Breslow, N. (1999). Estimation of disease rates in small areas: A new mixed model for spatial dependence. In Halloran, M. E. and Berry, D., editors, *Statistical Models in Epidemiology, the Environment, and Clinical Trials*, pages 135–178. Springer, New York.

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using Pólya-gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349.

Roberts, G. O. and Rosenthal, J. S. (2007). Coupling and ergodicity of adaptive markov chain monte carlo algorithms. *Journal of Applied Probability*, 44(2):458–475.

Roberts, G. O. and Rosenthal, J. S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367.

Windle, J., Carvalho, C. M., Scott, J. G., and Sun, L. (2013). Efficient data augmentation in dynamic models for binary and count data. *arXiv preprint arXiv:1308.0774*.

Windle, J., Polson, N. G., and Scott, J. G. (2014). Sampling Pólya-gamma random variates: Alternate and approximate techniques. *arXiv preprint arXiv:1405.0506*.

Zens, G., Frühwirth-Schnatter, S., and Wagner, H. (2024). Ultimate Pólya gamma samplers–efficient MCMC for possibly imbalanced binary and categorical data. *Journal of the American Statistical Association*, 119(548):2548–2559.

Zhou, M. and Carin, L. (2015). Negative binomial process count and mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):307–320.

Zhou, M., Li, L., Dunson, D., and Carin, L. (2012). Lognormal and gamma mixed negative binomial regression. *Proceedings of the 29th International Conference on Machine Learning*.

# Index