Normal-Block models: distributions, criteria and explicit V(E)M solutions

Julien Chiquet, Nestor Ngalala Manguitini, Jeanne Tous

2026-09-09

This document is a reference card for every Normal-Block variant implemented in `src/` (C++/RcppArmadillo core) and validated against the R6 reference implementation. It follows the notation of Tous (2026) : $\odot$ Hadamard product, $\oslash$ term-to-term inverse, A_i the (column) transpose of the i-th row of A, and

$$A_{\text{row-sum}} = \Big(\sum_i A_{i\cdot}\Big)^\top, \qquad A_{\text{col-sum}} = \sum_j A_{\cdot j}, \qquad A_{\text{total-sum}} = \sum_{ij} A_{ij}.$$

For each variant we give: the generative model, the optimized criterion (log-likelihood or ELBO), and the explicit E/M (or VE/M) updates exactly as implemented in `src/`. All formulas below have been checked against Tous (2026), Tous and Chiquet (2026) and Nglala Manguitini et al. (2026), and validated numerically against the original reference implementation in `tests/testthat/test-cpp-normal-block.R`, `test-cpp-normal-block-mean.R` and `test-cpp-zi-normal-block.R`.

These models and their inference algorithms were developed over the PhD thesis of Jeanne Tous (Normal-Block family, §§0-9; Tous (2026), Tous and Chiquet (2026)) and the Master's research internship of Nestor Ngalala Manguitini (Mean-Block family, §§10-11; Nglala Manguitini et al. (2026)), both supervised by Julien Chiquet. Jeanne Tous also participated in the supervision of Nestor's internship.

> **Note on the use of generative AI.** The models themselves – their definition and every algebraic derivation – were worked out by the students and their advisor over these two projects, who also wrote the first working versions of the code, its object-oriented architecture, and the first simulation studies. Generative AI (Claude Sonnet and Claude Opus, Anthropic) was used afterwards to improve the documentation (including this document), produce synthesis material, and audit, test, and improve the code – in particular to port several algorithmic components to C++: the E/M and VE/M steps detailed below, their SQUAREM-accelerated variant (§18), and

the graphical lasso solver (§17), whose original R-callback implementation caused intermittent memory crashes.

Part I – The models

This part is the mathematical reference: for each variant, the generative model, the criterion being optimized, and the explicit updates.

Normal-Block models (variance clustering)

The clustering structures the *latent covariance*: two variables are in the same block when they respond identically to a shared latent factor, i.e. when they **covary** the same way (§§0-9). This is the model of Tous (2026) and Tous and Chiquet (2026), and the one the package (and this document) are named after.

0. The generative model

$$\begin{aligned} \text{Latent space:} \quad & W_i \sim \mathcal{N}(0, \Omega^{-1}), \quad W_i \in \mathbb{R}^q \\ \text{Observation space:} \quad & Y_i \mid W_i \sim \mathcal{N}(CW_i + B^\top X_i,\, D), \quad Y_i \in \mathbb{R}^p \end{aligned}$$

Y $(n \times p)$, X $(n \times d)$, W $(n \times q$, stacked row-wise). C $(p \times q)$ is either a fixed 0/1 clustering matrix (**known clusters**), or a matrix of $\text{Mult}(1, \alpha)$ membership probabilities τ (**unknown clusters**, inferred jointly with everything else). D is the residual covariance: $D = \text{diag}(d^{-1})$ (**diagonal** noise) or $D = \xi^{-1} I_p$ (**spherical** noise, i.e. d constant across variables).

1. Known clusters, diagonal noise (EM)

R6: `NormalBlockVarKnownClusters` (`noise_covariance = "diagonal"`).

C++: `norm_block_var_cov_diag_noise_known_clusters`.

E-step. Posterior $W_i \mid Y_i \sim \mathcal{N}(\mu_i, \Gamma)$, shared Γ (C is fixed, so Γ does not depend on i):

$$\Gamma = (\Omega + C^\top D^{-1} C)^{-1} \qquad [q \times q]$$

$$\mu = (Y - XB)\, D^{-1} C \Gamma \qquad [n \times q]$$

M-step. With $R_\mu = Y - XB - \mu C^\top$:

$$\hat{B} = (X^\top X)^{-1} X^\top (Y - \mu C^\top)$$

$$\hat{\Sigma} = \frac{1}{n}\mu^\top \mu + \Gamma$$

$$\hat{\Omega} = \hat{\Sigma}^{-1} \qquad \text{(or graphical lasso, §13)}$$

$$\hat{d} = \frac{(R_\mu^2)_{\text{col-sum}}}{n} + C\,\text{diag}(\Gamma)$$

Criterion (the exact marginal log-likelihood of Y, valid at *any* (B, d, Ω), not only at an M-step optimum – Y_i is marginally $\mathcal{N}(B^\top X_i, D + C\Omega^{-1}C^\top)$, and $|D + C\Omega^{-1}C^\top|/(D + C\Omega^{-1}C^\top)^{-1}$ are rewritten through $\Gamma^{-1} = \Omega + C^\top D^{-1} C$ – the same $q \times q$ matrix the E-step above inverts – via the matrix determinant lemma and the Woodbury identity, avoiding the $p \times p$ inverse entirely). With $R = Y - XB$ and μ/Γ as in the E-step (an identity valid for *any* θ, since this model's E-step is exact Gaussian conditioning, not an approximation):

$$J = -\frac{np}{2}\log(2\pi) + \frac{n}{2}\sum_j \log(d_j) + \frac{n}{2}\log|\Omega| + \frac{n}{2}\log|\Gamma| - \frac{1}{2}\Big(\underbrace{\textstyle\sum_j d_j\,(R^2)_{\text{col-sum}_j}}_{\text{weighted SSQ}} - \text{tr}(\Gamma^{-1}\mu^\top\mu)\Big)$$

> **On the relation to a simpler, M-step-only formula.** At a point where d is exactly the M-step's own argmax for *this same* μ/Γ (i.e. right after a real E-step/M-step pair, as in plain EM – but *not*, e.g., at a SQUAREM-extrapolated candidate), the bracketed quadratic term collapses to the constant np, and J reduces to $-\frac{np}{2}\log(2\pi e) + \frac{n}{2}\sum_j \log(d_j) + \frac{n}{2}\log|\Omega| + \frac{n}{2}\log|\Gamma|$. The general form above is what `objective()` now actually computes, since it remains valid off that trajectory.

2. Known clusters, spherical noise (EM)

R6: `NormalBlockVarKnownClusters` (`noise_covariance = "spherical"`).

C++: `norm_block_var_cov_spherical_noise_known_clusters`.

Identical to §1 except d is replaced by a single scalar ξ, shared across all p variables:

$$\xi^{-1} = \frac{1}{p}\sum_j \hat{d}_j = \frac{(R_\mu^2)_{\text{total-sum}}}{np} + \frac{C^\top_{\text{col-sum}}\text{diag}(\Gamma)}{p}$$

($\hat{d}$ as in §1, before collapsing to a scalar.) The criterion is the same formula as §1 with $d_j = \xi$ for all j (so $\sum_j \log(d_j) = p\log(\xi)$, and the weighted SSQ collapses to $\xi \cdot (R^2)_{\text{total-sum}}$).

3. Unknown clusters, diagonal noise (VEM)

R6: `NormalBlockVarUnknownClusters` (`noise_covariance = "diagonal"`).

C++: `norm_block_var_cov_diag_noise_unknown_clusters`.

Mean-field approximation $\mathbb{P}(W, C \mid Y) \approx \prod_i \pi_1(W_i) \prod_j \pi_2(C_j)$, with $W_i \sim \mathcal{N}(M_i, \mathrm{diag}(S))$ (S shared across rows – no zero-inflation mask to break that symmetry) and $C_j \sim \mathrm{Mult}(1, \tau_{j\cdot})$.

VE-step.

$$\tilde{\Gamma} = (\Omega + \mathrm{diag}(\tau^\top d))^{-1} \qquad [q \times q, \text{ shared}]$$

$$M = (Y - XB)\, D^{-1} \tau \tilde{\Gamma} \qquad [n \times q]$$

$$S = \mathrm{diag}(\tilde{\Gamma}) \qquad [q]$$

$$\eta = d \otimes_{\text{rows}} \left((Y - XB)^\top M\right) - \frac{1}{2}\, d\, \left(M^2_{\text{col-sum}} + nS\right)^\top + \mathbf{1}_p \log(\alpha)^\top - 1$$

$$\tau_{j\cdot} = \mathrm{softmax}(\eta_{j\cdot}) \qquad \text{(skipped if } q = 1 \text{ or fixed\_tau)}$$

M-step. With $R = Y - XB$ (computed at the *start* of the VEM iteration, refreshed once B is updated below – see Part II) and $A = R^2 - 2R \odot (M\tau^\top) + (M^2 + S)\tau^\top$:

$$\hat{B} = (X^\top X)^{-1} X^\top (Y - M\tau^\top)$$

$$\hat{d} = \mathrm{mean}_{\text{rows}}(A)$$

$$\hat{\Sigma}_q = \frac{1}{n} M^\top M + \mathrm{diag}(S)$$

$$\hat{\Omega} = \hat{\Sigma}_q^{-1} \qquad \text{(or graphical lasso, §13)}$$

$$\hat{\alpha} = \mathrm{mean}_{\text{cols}}(\tau)$$

Criterion (ELBO):

$$J = -\frac{np}{2}\log(2\pi e) + \frac{n}{2}\sum_j \log(d_j) + \frac{n}{2}\log|\Omega| + \frac{n}{2}\sum_k \log(S_k) + \sum_j \tau_{j\cdot}\log(\alpha) - \sum_{jk}\tau_{jk}\log(\tau_{jk})$$

4. Unknown clusters, spherical noise (VEM)

R6: `NormalBlockVarUnknownClusters` (`noise_covariance = "spherical"`).

C++: `norm_block_var_cov_spherical_noise_unknown_clusters`.

Identical to §3 with $d_j = \xi$ for all j:

$$\xi^{-1} = \frac{1}{p}\sum_j \hat{d}_j = \frac{A_{\text{total-sum}}}{np}$$

5. Zero-inflation extension

R6: `ZINormalBlockVarKnownClusters`, `ZINormalBlockVarUnknownClusters`. C++: `ZINormalBlockData`, `ZINormalBlockVarKnownClusters` / `ZINormalBlockVarUnknownClusters`.

$$\begin{aligned}\text{Excess-of-zero layer:}\quad & Z_i \sim \bigotimes_j \text{Bernoulli}(\kappa_j)\\ \text{Observation space:}\quad & Y_i \mid Z_i, W_i = Z_i \odot \delta_0 + (1 - Z_i) \odot (CW_i + \mathcal{N}(B^\top X_i, D))\end{aligned}$$

We write $\mathbf{1}_Y =$ `zeros_bar` ($n \times p$, 1 where $Y_{ij} \neq 0$), $\mathbf{0}_Y = 1 - \mathbf{1}_Y$, $nY = (\mathbf{1}_Y)_{\text{col-sum}}$ (per-variable count of non-zero observations), $npY = (\mathbf{1}_Y)_{\text{total-sum}}$.

κ is estimated once, upfront, outside the (V)EM loop: a per-variable logistic regression of $\mathbf{0}_Y[\cdot, j]$ on the zero-inflation design matrix X_0 (defaulting to an intercept-only column when X_0 is not supplied, in which case κ_j reduces exactly to the empirical zero-proportion of variable j, as in `nb_block.pdf` §3.C). κ is never updated during the (V)EM recursion; only its fixed contribution to the log-likelihood,

$$\texttt{zi\_cond\_mean} = \sum(\mathbf{0}_Y \odot \log(\kappa)) + \sum(\mathbf{1}_Y \odot \log(1 - \kappa)),$$

is carried through the C++ core (`ZINormalBlockData::zi_cond_mean`).

Throughout, $W = \mathbf{1}_Y \odot D^{-1}$ (broadcast d over rows, masked by $\mathbf{1}_Y$) is the "masked weight" matrix used in every closed-form solve below (`dm1_mat` / `DM1` in the code).

6. ZI, known clusters, diagonal noise (EM)

C++: `zi_norm_block_var_cov_diag_noise_known_clusters`.

Because the zero-inflation mask varies row by row, the posterior covariance of $W_i \mid Y_i$ is **row-dependent**: Γ is a family of n distinct $q \times q$ matrices (`arma::cube`), not a single shared matrix as in §1.

E-step. With $R = Y - XB$ (current B):

$$\forall i: \quad \Gamma^{(i)} = \left(\Omega + \operatorname{diag}((WC)_{i\cdot})\right)^{-1} \qquad [q \times q]$$

$$\forall i: \quad \mu_i = \left((R \odot W)C\right)_{i\cdot} \Gamma^{(i)} \qquad [q]$$

M-step. B has no closed form *as a single linear system* (the mask makes each column's weights different), but the per-column subproblem stays exactly quadratic, hence solvable column-by-column (`nb_optim::solve_wls`, Part II):

$$\hat{B} = \arg\min_B \sum \left(W \odot (Y - XB - \mu C^\top)^2\right) \qquad \text{[solved one column at a time]}$$

With $R_\mu = R - \mu C^\top$ (R refreshed with the new $\hat{B}$ first, see Part II) and $\left(C\operatorname{diag}(\Gamma)\right)_{i} := C\,\operatorname{diag}(\Gamma^{(i)})$ (expansion of each row's posterior variance to the variable level via the cluster structure):

$$A = R_\mu^2 + \left(C\operatorname{diag}(\Gamma)\right)_{i\cdot} \qquad [n \times p]$$

$$\hat{d} = nY \oslash (\mathbf{1}_Y \odot A)_{\text{col-sum}}$$

$$\hat{\Sigma} = \frac{1}{n}\left(\mu^\top \mu + \sum_i \Gamma^{(i)}\right)$$

$$\hat{\Omega} = \hat{\Sigma}^{-1} \qquad \text{(or graphical lasso, §13)}$$

Criterion (mirrors §1's general form: one marginal per row, restricted to that row's own observed/non-zero-inflated columns, via the same Woodbury trick applied to each row's own $\Gamma^{(i),-1}$):

$$J = -\frac{npY}{2}\log(2\pi) + \frac{1}{2}\sum_j nY_j \log(d_j) + \frac{n}{2}\log|\Omega| + \frac{1}{2}\sum_i \log|\Gamma^{(i)}|$$
$$- \frac{1}{2}\Big(\underbrace{\sum(W \odot R^2)}_{\text{weighted SSQ}} - \sum_i \mu_i^\top \Gamma^{(i),-1}\mu_i \Big) + \texttt{zi\_cond\_mean}.$$

valid at any (B, d, Ω), for the same reason as §1. At an M-step-consistent point the bracketed term collapses to npY and this reduces to

$$J = -\frac{npY}{2}\log(2\pi e) + \frac{1}{2}\sum_j nY_j \log(d_j) + \frac{n}{2}\log|\Omega| + \frac{1}{2}\sum_i \log|\Gamma^{(i)}| + \texttt{zi\_cond\_mean}.$$

7. ZI, known clusters, spherical noise (EM)

C++: `zi_norm_block_var_cov_spherical_noise_known_clusters`. Identical to §6 with:

$$\xi^{-1} = \frac{npY}{(\mathbf{1}_Y \odot A)_{\text{total-sum}}}$$

8. ZI, unknown clusters, diagonal noise (VEM)

C++: `zi_norm_block_var_cov_diag_noise_unknown_clusters`.

Same mean-field approximation as §3, but the zero-inflation mask makes the variational variance *row-dependent*: S is now an $n \times q$ matrix, not a length-q vector.

VE-step. With $R = Y - XB$:

$$M = \arg\max_M \; -\frac{1}{2}\Big(\sum \left(W \odot (M^2\tau^\top - 2R \odot (M\tau^\top))\right) + \sum \left((M\Omega) \odot M\right)\Big)$$

$$S = \left(W\tau + \mathbf{1}_n \mathrm{diag}(\Omega)^\top\right)^{\oslash} \qquad [n \times q]$$

$$\eta = -\frac{1}{2}\Big(W^\top(M^2 + S) - 2(W \odot R)^\top M\Big) + \mathbf{1}_p \log(\alpha)^\top - 1$$

$$\tau_{j\cdot} = \mathrm{softmax}(\eta_{j\cdot}) \qquad \text{(skipped if } q = 1 \text{ or fixed\_tau)}$$

M's objective is exactly quadratic and *row-separable* (mirrors $\Gamma^{(i)}/\mu_i$ in §6): each row solves an independent $q \times q$ ridge system (`nb_optim::solve_M_ridge`, Part II):

$$\forall i: \quad M_i = \left(\Omega + \operatorname{diag}((WC)_{i\cdot})\right)^{-1}\left((W \odot R)C\right)_{i\cdot}$$

M-step. With $A = R^2 - 2R \odot (M\tau^\top) + (M^2 + S)\tau^\top$ and R refreshed with the new $\hat{B}$ before estimating d (Part II):

$$\hat{B} = \arg\min_B \sum \left(W \odot (Y - XB - M\tau^\top)^2\right) \qquad \text{[solved one column at a time]}$$

$$\hat{d} = nY \oslash (\mathbf{1}_Y \odot A)_{\text{col-sum}}$$

$$\hat{\alpha} = \text{mean}_{\text{cols}}(\tau)$$

$$\hat{\Sigma}_q = \frac{1}{n} M^\top M + \operatorname{diag}(\text{mean}_{\text{cols}}(S))$$

$$\hat{\Omega} = \hat{\Sigma}_q^{-1} \qquad \text{(or graphical lasso, §13)}$$

Criterion:

$$\begin{aligned} J = &-\frac{npY}{2}\log(2\pi e) + \frac{1}{2}\sum_j nY_j \log(d_j) + \frac{n}{2}\log|\Omega| + \frac{1}{2}\sum_{ik}\log(S_{ik}) \\ &+ \sum_j \tau_{j\cdot}\log(\alpha) - \sum_{jk}\tau_{jk}\log(\tau_{jk}) + \texttt{zi\_cond\_mean} \end{aligned}$$

($\sum_{ik}\log(S_{ik})$ now ranges over all $n \times q$ entries, since S is row-dependent.)

9. ZI, unknown clusters, spherical noise (VEM)

C++: `zi_norm_block_var_cov_spherical_noise_unknown_clusters`. Identical to §8 with:

$$\xi^{-1} = \frac{npY}{(\mathbf{1}_Y \odot A)_{\text{total-sum}}}$$

Mean-Block models (response clustering)

The clustering structures the *mean*: two variables are in the same block when they respond identically to the covariates, i.e. when they share a *regression profile* (§§10-11). See Nglala Manguitini et al. (2026) for the model's original derivation.

10. The generative model

R6: `NormalBlockMeanKnownClusters`, `NormalBlockMeanUnknownClusters` (`normal_block(..., model = "mean")`).

C++: `norm_block_mean_known_clusters`, `norm_block_mean_unknown_clusters`.

$$Y_i = CB^\top X_i + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \Sigma), \quad Y_i, \varepsilon_i \in \mathbb{R}^p$$

C ($p \times q$) is the same clustering matrix as in §0 – fixed 0/1, or a variational τ – but it now multiplies the mean rather than a latent vector. B is $d \times q$, *one column per cluster* (against $d \times p$, one per variable, in §§1-9): variables in the same block are constrained to respond *identically* to the covariates. In matrix form $\mathbb{E}[Y] = XBC^\top$. Σ is the $p \times p$ residual covariance, $\Omega = \Sigma^{-1}$, with three possible shapes (§10.3).

The two families are complementary, not variants of one another:

	variance-block (§§1-9)	mean-block (§§10-11)
what the clustering constrains	the latent covariance	the mean response to X
B	$d \times p$, unconstrained	$d \times q$, shared within a block
covariance	D diagonal/spherical, plus $C\Omega^{-1}C^\top$	Σ, unconstrained up to its shape
latent variable	$W_i \in \mathbb{R}^q$	none
free parameters	$pd + q + \binom{q}{2} + \lvert D\rvert$	$qd + \lvert\Sigma\rvert$

Because there is no latent variable, the known-clusters case is not an EM at all: it is block coordinate ascent on the exact log-likelihood, alternating B and Σ (the C++ `E_step()` is a deliberate no-op there). Only the unknown-clusters case needs a variational treatment, and only for C.

10.1 Known clusters

M-step. Maximizing $-\frac{1}{2}\mathrm{tr}((Y - XBC^\top)\Omega(Y - XBC^\top)^\top)$ in B gives the normal equations $X^\top Y\,\Omega C = X^\top X\,B\,C^\top\Omega C$, hence

$$\hat{B} = (X^\top X)^{-1}X^\top Y\,\Omega\,C\,(C^\top\Omega C)^{-1} \qquad [d \times q]$$

$$R = Y - XBC^\top, \qquad \hat{\Sigma} = \frac{1}{n}R^\top R \qquad \text{(shape-restricted, §10.3)}$$

Criterion (exact log-likelihood, valid at any (B, Ω)):

$$J = -\frac{np}{2}\log(2\pi) + \frac{n}{2}\log|\Omega| - \frac{1}{2}\mathrm{tr}(R\,\Omega\,R^\top)$$

10.2 Unknown clusters (VEM)

Mean-field $\mathbb{P}(C \mid Y) \approx \prod_j \pi(C_j)$ with $C_j \sim \mathrm{Mult}(1, \tau_{j\cdot})$ and prior α. Write $w = \mathrm{diag}(\Omega)$ and $M = B^\top X^\top XB$ $(q \times q)$.

The only quantity the mean-field introduces is $\mathbb{E}_q[C^\top\Omega C]$, which is *not* $\tau^\top\Omega\tau$: the diagonal blocks $\mathbb{E}[C_jC_j^\top] = \mathrm{Diag}(\tau_{j\cdot})$ differ from $\tau_{j\cdot}\tau_{j\cdot}^\top$. Hence

$$\Psi = \mathbb{E}_q[C^\top\Omega C] = \tau^\top\Omega\tau + \underbrace{\mathrm{Diag}(\tau^\top w) - \tau^\top\mathrm{Diag}(w)\,\tau}_{\Phi} \qquad [q \times q]$$

Φ is the variance part, and is what survives in the ELBO's trace term below.

M-step.

$$\hat{B} = (X^\top X)^{-1}X^\top Y\,\Omega\,\tau\,\Psi^{-1}$$

$$\bar{R} = Y - XB\tau^\top, \qquad \Lambda = \mathrm{Diag}(\lambda), \quad \lambda_j = \frac{\tau_{j\cdot}^\top\mathrm{diag}(M) - \tau_{j\cdot}^\top M\tau_{j\cdot}}{n}$$

$$\hat{\Sigma} = \frac{1}{n}\bar{R}^\top\bar{R} + \Lambda \qquad \text{(shape-restricted, §10.3)}, \qquad \hat{\alpha} = \frac{1}{p}\tau_{\text{col-sum}}$$

Λ is the variational correction: $\mathbb{E}_q\|Y_i - CB^\top X_i\|^2$ exceeds the plug-in residual $\bar{R}$ by exactly the within-row variance of the cluster assignment.

VE-step. Each row of τ has a closed-form softmax; with $G = Y^\top XB - \tau M$,

$$\eta_{jk} = \log \alpha_k + (\Omega_{j\cdot} G)_k + w_j \, (\tau_{j\cdot}^\top M)_k - \tfrac{1}{2} w_j M_{kk}, \qquad \tau_{j\cdot} = \mathrm{softmax}(\eta_{j\cdot})$$

The rows are swept sequentially (Gauss-Seidel), G being refreshed after each: see Part II, which explains why updating them simultaneously is not merely slower but can cycle.

ELBO.

$$J = -\frac{np}{2} \log(2\pi) + \frac{n}{2} \log |\Omega| + \sum_{jk} \tau_{jk} \log \alpha_k - \sum_{jk} \tau_{jk} \log \tau_{jk} - \frac{1}{2} \Big(\mathrm{tr}(\bar{R}\, \Omega\, \bar{R}^\top) + \mathrm{tr}(\Phi M) \Big)$$

10.3 The shape of Σ

Σ is $p \times p$ here, not $q \times q$, so its shape is a modelling decision rather than a detail. Writing $S = \frac{1}{n} \bar{R}^\top \bar{R} + \Lambda$ (or $\frac{1}{n} R^\top R$ for known clusters):

`noise_covariance`	$\hat{\Omega}$	free parameters
`"full"`	S^{-1}, or the graphical lasso of §13	$p + \binom{p}{2}$
`"diagonal"` (default)	$\mathrm{Diag}(1 \oslash \mathrm{diag}(S))$	p
`"spherical"`	$\mathrm{Diag}(1/\overline{\mathrm{diag}(S)})$	1

`"diagonal"` is the default, for a statistical reason rather than a computational one: a full Σ spends $p(p+1)/2$ parameters that drown the mean structure BIC/ICL are being asked to weigh, and it selects q markedly worse as p approaches n – 10/12 correct selections against 6/12 at $n/p = 1.3$ in a simulation study, *even when the data were generated with a full* Σ (`inst/mean_block_analyses/covariance_shape_selection_study.R`). Clustering quality at fixed q is the same either way; what a full Σ costs is the choice of q. It wins back at a comfortable n/p, where the covariance is well estimated.

Two consequences of the restricted shapes: they never invert anything, so unlike `"full"` they carry no $n > p$ requirement; and with Ω diagonal, Ψ collapses to $\mathrm{Diag}(\tau^\top w)$ and the τ-dependence in η above cancels, making the VE-step a genuine one-shot softmax. Asking for `sparsity > 0` implies `"full"`: there are no off-diagonal terms for the graphical lasso to penalize otherwise.

11. Zero-inflated mean-block

R6: `ZINormalBlockMeanKnownClusters`, `ZINormalBlockMeanUnknownClusters`. C++: `zi_norm_block_mean_known_clusters`, `zi_norm_block_mean_unknown_clusters`.

Same excess-of-zero layer as §5, same κ estimated once upfront, same fixed `zi_cond_mean` contribution. Restricted to `"diagonal"` and `"spherical"` Σ, and for a structural reason: a full Σ ties the variables together within each row, and the mask leaves a *different* set of them observed in every row, so each row would need its own $p_i \times p_i$ submatrix inverse – a missing-data EM rather than a reweighting.

With Σ diagonal the Gaussian part factorizes over (i, j) and the mask is simply a 0/1 weight. Writing w for the precision vector, $Z = XB$ $(n \times q)$, T for the membership matrix (C if known, τ if not), and $\mathbf{1}_Y$ as in §5, everything the recursion needs is carried by two $p \times q$ statistics:

$$A_{jk} = \sum_i (\mathbf{1}_Y)_{ij} Z_{ik}^2, \qquad P_{jk} = \sum_i (\mathbf{1}_Y)_{ij} Y_{ij} Z_{ik}, \qquad \mathrm{ssq}_j = \sum_i (\mathbf{1}_Y)_{ij} Y_{ij}^2$$

$$s_j \;=\; \mathbb{E}_T\Big[\sum_i (\mathbf{1}_Y)_{ij}\,(Y_{ij} - Z_{i,c_j})^2\Big] \;=\; \mathrm{ssq}_j - 2\sum_k T_{jk} P_{jk} + \sum_k T_{jk} A_{jk}$$

No $p \times p$ matrix is ever formed, and no separate Λ is needed: the variational variance is already inside the T-weighted A term.

M-step. The B-update loses the single closed form of §10.1. The weight of observation i for cluster k, $a_{ik} = \sum_j (\mathbf{1}_Y)_{ij} w_j T_{jk}$, now depends on i, so the one $(X^\top X)^{-1}$ solve splits into q weighted least squares in dimension d:

$$\hat{\beta}_k = \left(X^\top \mathrm{Diag}(a_{\cdot k}) X\right)^{-1} X^\top b_{\cdot k}, \qquad b_{ik} = \sum_j (\mathbf{1}_Y)_{ij}\, w_j\, Y_{ij}\, T_{jk}$$

$$\hat{w}_j = \frac{nY_j}{s_j} \quad \text{(diagonal)}, \qquad \hat{w} = \frac{npY}{\sum_j s_j} \quad \text{(spherical)}, \qquad \hat{\alpha} = \frac{1}{p}\tau_{\text{col-sum}}$$

VE-step (unknown clusters). The ELBO is linear in each row of τ up to the entropy, and the rows are independent, so the softmax is the exact maximizer in one shot: no sweep, unlike §10.2, whose full Σ couples the rows:

$$\tau_{jk} \propto \alpha_k \exp\Big(w_j\big(P_{jk} - \tfrac{1}{2}A_{jk}\big)\Big)$$

Criterion.

$$J = -\frac{npY}{2}\log(2\pi) + \frac{1}{2}\sum_j nY_j \log(w_j) - \frac{1}{2}\sum_j w_j s_j + \texttt{zi\_cond\_mean}$$

plus $\sum_{jk} \tau_{jk} \log \alpha_k - \sum_{jk} \tau_{jk} \log \tau_{jk}$ for unknown clusters.

As a check on the mask handling: on data with no zeros at all, $\kappa = 0$, `zi_cond_mean` $= 0$, and these reduce exactly to §10 – verified numerically to 10^{-5} in `tests/testthat/test-ZINormalBlockMean*.R`.

Combining both families

The two remaining topics apply across both families rather than to one: chaining them (§12), and the sparsity penalty (§13) shared by every variant that carries a full precision matrix to regularize (§§1-9's cluster-level Ω, §10's variable-level one under `noise_covariance = "full"`).

12. Sequential mean-then-variance fits

`normal_block_sequential()` fits the two families one after the other: a mean-block model on Y, then a variance-block model on its residuals with an intercept-only design (the covariate effects having been removed by the first stage). This is a heuristic two-stage estimator, not a joint model.

It is worth naming because it answers a question the two families raise together. On a positive control carrying two genuinely distinct structures it recovers both exactly (ARI = 1 on each). On `brca_rppa` the two partitions are unrelated (ARI 0.018), so a joint model forcing them equal would be badly misspecified – but the covariance clustering is largely unchanged by removing the mean structure first (ARI 0.712 against fitting it directly), which suggests the two structures are *separable* rather than competing. A genuinely joint mean+covariance model, with free or linked partitions, remains open.

13. Sparse precision matrix (graphical lasso)

In every variant above, whenever `sparsity` > 0, $\hat{\Omega}$ is no longer the exact inverse of $\hat{\Sigma}$ (or $\hat{\Sigma}_q$): it is estimated by the graphical lasso (`src/graphical_lasso.h`, reached through `nb_omega::estimate()` in `src/omega_estimation.h`) applied to $\hat{\Sigma}$ with penalty $\lambda \odot W_\lambda$ (W_λ a $q \times q$ matrix of per-pair weights, `sparsity_weights`, diagonal excluded from the penalty, and λ the scalar `sparsity`).

For the **known-clusters criteria (§1, §2, §6, §7)**, this needs no special case at all: the general formula given there is $\log p(Y;\theta)$ for *any* symmetric positive-definite Ω, sparse or not – it never assumes $\hat{\Omega}$ is $\hat{\Sigma}$'s exact inverse, so no correction term is added when $\lambda > 0$.

For the **unknown-clusters criteria (§3, §4, §8, §9)**, which still use the profiled (M-step-only) ELBO form, $\hat{\Omega}$ no longer satisfies $\operatorname{tr}(\hat{\Omega}\hat{\Sigma}) = q$ there (the identity that makes the corresponding term collapse to a constant when the inversion is exact), so those four criteria gain an explicit correction term when $\lambda > 0$:

$$J_{\text{struct}} = J + \frac{nq}{2} - \frac{n}{2}\operatorname{tr}(\hat{\Omega}\hat{\Sigma}) - \lambda \sum \left|W_\lambda \odot \hat{\Omega}\right|$$

where $\hat{\Sigma}$ is the corresponding M-step covariance estimate of each variant. This is implemented identically (modulo the variant-specific $\hat{\Sigma}$) in `NormalBlockVarUnknownClusters` and `ZINormalBlockVarUnknownClusters`'s `objective()`.

The **mean-block unknown-clusters criterion (§10.2)** needs no such correction, for a different reason than the known-clusters case above: it is not a profiled shortcut to begin with. Its ELBO (the formula in §10.2) is written out in full from (B, Ω) – via $\log|\Omega|$ and the trace terms $\operatorname{tr}(\bar{R}\,\Omega\,\bar{R}^\top) + \operatorname{tr}(\Phi M)$ – and never assumes $\Omega = \hat{\Sigma}^{-1}$ at any point, sparse or not; the penalty term is simply subtracted from it directly (`NormalBlockMeanUnknownClusters::objective()`, `src/normal_block_mean_unknown_clusters.h`).

Part II – Implementation and numerical notes

Nothing in this part changes any model of Part I. It records how those formulas are actually evaluated, which shortcuts are exact and which are approximations, and the numerical traps found along the way.

14. Correctness invariants of the recursion

- **The residual** $R = Y - XB$ is always refreshed after B is updated. Every d/`dm1` update above (§§1-4, 6-9) uses the residual computed with the *current* B (i.e. B is updated first within the M-step, then R is recomputed before estimating d). Earlier revisions of the R6 reference implementation (`NormalBlockVarUnknownClusters`, `ZINormalBlockVarKnownClusters`, `ZINormalBlockVarUnknownClusters`) recomputed R only in the known-clusters non-ZI model (§§1, 2) and reused a one-iteration-stale R in the other three; this has been fixed on both the R and C++ sides for consistency (validated to machine/optimizer precision against R6 in `tests/testthat/`).

15. Algebraic shortcuts and caching

- **No iterative optimizer is used anywhere.** Originally, the reference document Tous and Chiquet (2026) and Tous (2026) presents the ZI B- and M-updates as requiring gradient ascent (the R6 reference used to call `nloptr`'s L-BFGS for both, within the (V)EM recursion). In fact, the zero-inflation mask only reweights residuals (it introduces no nonlinearity) so both subproblems stay exactly quadratic: B's normal equations decouple into one weighted least-squares solve per column (`nb_optim::solve_wls`), and M's decouple into one $q \times q$ ridge solve per row (`nb_optim::solve_M_ridge`, identical in structure to $\Gamma^{(i)}/\mu_i$ in the ZI known-clusters E-step). Both are implemented in `src/zi_closed_form_solvers.h`. The same observation applies to the R-side heuristic initialization of B (`zi_weighted_fit()` in `R/utils.R`), which solves the same per-column weighted least squares problem directly instead of calling `nloptr`. This removes any dependency on `nloptr`/NLopt entirely, and is substantially faster.
- **The zero-inflated per-row systems are solved as the symmetric positive definite systems they are.** The mask makes each row's posterior distinct, so `solve_M_ridge()` solves one $q \times q$ system per observation and `solve_wls()` one $d \times d$ system per variable; the ZI known-clusters E-step inverts one $\Gamma^{(i)}$ per row. That much is intrinsic. Paying a fresh allocation and a *general* linear solver for each of them is not: Ω is positive definite and the diagonal added to it is non-negative, so every one of these is SPD. Reusing a buffer and asking Armadillo for the SPD path (`solve_opts::likely_sympd`, with a guarded fallback that has never fired in practice) is 1.7-2x on a whole zero-inflated fit at $q = 4..25$ – at these sizes the cost is the per-call overhead, not the $O(nq^3)$ arithmetic.
- **The mean-block VE-step sweeps** τ's rows sequentially, and must. Each row's softmax (§10.2) maximizes the ELBO *exactly* in that row, its quadratic terms cancelling; a sweep visiting the rows one at a time, refreshing G after each, is therefore a coordinate ascent and cannot decrease the ELBO. Updating every row simultaneously from the same G is not: it was observed to settle into a period-2 cycle, alternating between two assignments forever. With Ω diagonal the τ-dependence in η cancels and one sweep suffices – which is why the zero-inflated variant (§11), diagonal by construction, gets a genuine one-shot softmax.

16. Numerical traps

- **The per-column weighted normal equations can be exactly singular, and this is expected, not a bug.** With a one-hot design (e.g. a categorical covariate with no intercept) and real zero-inflated data, it is common for some design level to have *zero* non-zero-inflated observations for a given variable: both $X^\top W X$ and $X^\top W y$ are then exactly zero on that coordinate (no information to estimate that regression coefficient at all). `solve()` in base R errors on this (`Lapack routine dgesv: system is exactly`

singular), whereas arma::solve() on the C++ side automatically falls back to a pseudo-inverse and only emits a (non-fatal) runtime warning – silenced via PKG_CPPFLAGS = -DARMA_WARN_LEVEL=1 in src/Makevars/Makevars.win, since it would otherwise print once per singular column, per EM iteration, per model in a collection (hundreds of thousands of lines on real microbiome-scale data). zi_weighted_fit() on the R side mirrors this by using MASS::ginv() instead of solve(), which returns the same minimum-norm solution (effectively a 0 coefficient for the indeterminate design level) instead of erroring. A related, separate failure mode: a variable with very few non-zero observations relative to X's degrees of freedom (e.g. a rare species seen at a single station) can be fit *exactly* by $\widehat{B}$ on one iterate, driving its weighted residual sum of squares to exact 0 and so $\widehat{d} = nY/0 = \infty$; that ∞ weight then propagates into the next iterate's $X^\top W X$, and ginv()'s SVD errors on the non-finite input. zi_weighted_fit() floors the weighted SSQ away from exact zero (.Machine$double.eps) to keep $\widehat{d}$ large but finite instead – this is what initially surfaced the issue, on the real onema dataset (R/data.R), before that dataset's rarest species were filtered out for the unrelated, purely statistical reason of carrying too little information to cluster on anyway.

- **Quantities shared by E_step()/M_step() within one (V)EM iteration are computed once, not re-derived.** In the two zero-inflated variants, the masked weight matrix (dm1_mat_/DM1_, equal to repmat(dm1, n) % zeros_bar) is computed once in E_step() and reused, identically, in M_step() (dm1_ only changes at the very end of M_step()), instead of being recomputed a second time.

- $XB = XB$ is cached across calls, invalidated only when B itself changes. Every concrete E_step()/M_step()/objective() needs $Y - XB$ at least once; left uncached, the *same* $O(npd)$ product gets recomputed up to three times per settled B: once inside M_step() (for its own residual), once more in the *next* E_step() (B hasn't changed since), and – for the two known-clusters classes, whose objective() is the general criterion below and needs R explicitly – a third time there too (called right after M_step(), same B again). NormalBlockBase::XB() memoizes the product lazily; subclasses update B exclusively through set_B() (never B_ = ... directly), which invalidates the cache, as do set_state()/copy_tracked_state_from() (the only other two places B changes, both used by the SQUAREM machinery below). This is unrelated to – and cannot reintroduce – the staleness bug the next paragraph describes: that bug was about a cache going stale *without* being invalidated on every path that should have invalidated it; this one is invalidated at the single point (B's own assignment) it could ever go stale at, so it always holds exactly XB for the *current* B, never a preceding iterate's.

- **objective() always recomputes every other input fresh from** (B, d, Ω), rather than reading back anything cached from the preceding E_step(). This used to matter for performance only (an earlier revision cached $\widehat{\Sigma}$ and Γ's log-determinant – a free byproduct of the Cholesky factorization E_step() already performs to invert Γ – and read them back in objective()), but those caches held *the previous iterate's* μ/Γ, one E_step() behind the (B, d, Ω) objective() was being asked to evaluate right after M_step().

That mismatch is harmless for the profiled shortcut (which only needs μ/Γ and (B, d, Ω) to be M-step-consistent *with each other*, not with the same calendar iteration) but would silently break the general criterion's correctness for *any* θ (§1, §2, §6, §7's main use case: a SQUAREM-extrapolated candidate, where there is no "preceding E-step" to reuse at all). `NormalBlockVarKnownClusters/ZINormalBlockVarKnownClusters::objective()` therefore caches nothing beyond XB above; the extra $q \times q$ (or, for the ZI variant, n row-wise $q \times q$) factorization per call is negligible next to the $O(np)$ work already done elsewhere in a (V)EM step.

17. The graphical lasso solver

The penalized $\widehat{\Omega}$ of §13 is computed by an in-package solver (`src/graphical_lasso.h`, reached through `nb_omega::estimate()`), a port of the block coordinate descent of Friedman et al. (2008) in the bookkeeping of Sustik and Calderhead (2012). It replaced a call back into R to `glassoFast::glassoFast()` made once per M-step – thousands of R round trips from inside a single `.Call` for one sparsity path. Three things came out of bringing it in-house.

- **The callback was a memory-safety liability.** One bug on that boundary had already been found and fixed (a `static Rcpp::Function` holding an R object outside R's protection discipline), and CI still crashed intermittently with heap corruption (`malloc(): unsorted double linked list corrupted`) while rebuilding the sparsity-path vignette – about 10% of runs, across R versions and platforms. The interaction is now simply gone rather than made less likely; that is the basis for the confidence, not the clean runs since.
- **Two bugs in `glassoFast` are fixed here.** When S carries no off-diagonal mass the problem separates and Θ is diagonal; `glassoFast` returns $1/\max(\rho_{ii}, \varepsilon)$ there, dropping S_{ii}, so with an unpenalized diagonal it hands back $\approx 9.09 \cdot 10^{15}$ instead of $1/S_{ii}$ – and **every** $q = 1$ problem takes that branch. Separately, its inner coordinate descent is an unbounded loop whose only exit is `dlx < thrLasso`, never true once `dlx` is `NaN`: a non-finite input hangs the process rather than failing. We return the correct diagonal solution, and reject non-finite input and non-positive $S_{ii} + \rho_{ii}$ up front. Agreement with `glassoFast` elsewhere was checked on 477 random problems (worst relative difference $2.2 \cdot 10^{-13}$) and, more sharply, on 40 (Σ, ρ) pairs captured from a real sparsity path, where it also takes the *identical* number of inner passes.
- **Each M-step warm-starts from the previous one.** Consecutive M-steps solve nearly the same problem, and the solver stops on how much a sweep moved W rather than on distance to the optimum – so resuming both converges in fewer sweeps and, at the tightened threshold that affords (10^{-6} instead of 10^{-4}; `nb_omega::kGlassoThreshold`, mirrored by `NB_GLASSO_THRESHOLD` in R for the reference recursion), lands *closer* to the optimum than a cold start at the looser one. On 124 consecutive M-steps of a real path: 1.355s at $6.1 \cdot 10^{-5}$ from the exact solution cold, 0.589s at $8.9 \cdot 10^{-7}$ warm.

18. SQUAREM: a generic accelerator wrapped around (V)EM, not a different algorithm

Principle. Any EM or VEM recursion can be written as a fixed-point iteration $\theta_{t+1} = T(\theta_t)$, where T is "one full E-step+M-step (or VE-step+M-step) cycle" – a map that is only ever guaranteed to be an *ascent* step on the objective J (the log-likelihood for EM, the ELBO for VEM), never told how fast. Near a fixed point $\theta^\star$, standard EM's local convergence rate is governed by the largest eigenvalue of T's Jacobian there; when that eigenvalue is close to 1 (typically, the more "mixing"/overlap there is between blocks, the more this happens as q grows here), consecutive EM steps barely move θ and convergence becomes painfully linear, regardless of how good the rest of the algorithm is. SQUAREM (Varadhan and Roland 2008) treats this purely as a generic fixed-point acceleration problem – the same idea behind Aitken's Δ^2 or Anderson acceleration – without using any second-order or model-specific information: from two successive applications of T alone, it extrapolates a step *longer* than T itself would take, then leans on T once more to repair whatever the extrapolation overshot. Crucially, T is the model's own, already-implemented update; SQUAREM only ever calls it, it never needs to know what's inside.

Pseudo-code (one outer iteration, matching `try_squarem_step()`):

```
Input: theta_0 (current iterate), T (one E/M or VE/M cycle), objective J
theta_1 <- T(theta_0)
theta_2 <- T(theta_1)           # two plain (V)EM steps, as usual

r <- theta_1 - theta_0
v <- (theta_2 - theta_1) - r
if ||v|| is ~0: return theta_2  # no curvature to extrapolate from -> plain EM

alpha <- -||r|| / ||v||         # alpha <= -1; alpha = -1 <=> no extrapolation

repeat up to 10 times:          # SQUAREM's quadratic extrapolation
    theta_star  <- theta_0 - 2*alpha*r + alpha^2*v
    if theta_star is feasible:
        theta_new <- T(theta_star) # one stabilizing (V)EM cycle
                                   # from the extrapolated point
        if J(theta_new) >= J(theta_2):
            return theta_new    # accept: replaces theta_2 as the next iterate

    alpha <- (alpha - 1) / 2    # backtrack steplength towards alpha = -1
return theta_2                  # every attempt rejected: fall back
```

How this extends standard EM. SQUAREM is not a competing algorithm: it is a wrapper around the model's own T, and it degenerates *exactly* to two ordinary (V)EM steps whenever

every extrapolation attempt is rejected (the `return theta_2` fallback above) – which is also why it can never do worse than plain EM: every accepted replacement is itself produced by one more application of T (the "stabilize" call), so it is guaranteed to be at least as good a (V)EM iterate as θ_2 was, just reached after a longer effective step. This is also why no per-model derivation is needed beyond what each class already implements (`E_step()`/`M_step()`/`objective()`) – the qualifier "for any EM algorithm" in Varadhan and Roland (2008)'s title is the point: the acceleration only ever touches $\theta = (B, d, \Omega)$ and calls existing code, nothing about it is specific to the Normal-Block model.

Every variant runs this SQUAREM-accelerated (V)EM, not plain EM, whenever `sparsity <= 0` (`NormalBlockBase::run_em()`/`supports_acceleration()`, `src/normal_block_em_base.h`), in both families. Motivation: as q grows, plain (V)EM's convergence rate approaches 1 (the ratio of consecutive objective increments tends to 1), requiring far more iterations than the package's iteration cap is set to by default – real fits with $q \gtrsim$ 15-20 would not converge within thousands of iterations of plain (V)EM. Concretely, only the parameters with a closed-form M-step are extrapolated: $\theta = (B, d, \Omega)$ for the variance-block family ($\tau/\alpha/M/S/\Gamma/\mu$ are never extrapolated), $\theta = (B, \Omega)$ for the mean-block one ($\tau/\alpha/\Lambda/\Psi/\Phi$ likewise) – in every case, only ever refreshed by the model's own E-step or VE-step inside T.

This acceptance test needs `objective()` to be a sound comparison between θ_2's and the stabilized candidate's value, but *not* for the same reason in every variant. For the known-clusters criteria (§1, §2, §6, §7), `objective()` is the *general* marginal log-likelihood, valid at any θ – the comparison is a real one of $\log p(Y; \theta)$, regardless of how either point was reached. For the unknown-clusters (variational) criteria (§3, §4, §8, §9), there is no such general, theta-only marginal likelihood available (the VE-step is a genuine mean-field approximation, not exact conditioning) – but no general form is needed either: every comparison happens right after a fresh VE-step+M-step pair, exactly the regime their existing *profiled* ELBO shortcut (the formula given in each section above) is already valid in, and that shortcut was checked algebraically to carry no sign bug (unlike the known-clusters case before its fix) – the M-step-optimality collapse to a constant checks out term for term. Measured speedup on real data (iterations to the package's convergence threshold): for the known-clusters classes, roughly 2-3x for $q \leq 10$ and qualitative (plain (V)EM does not converge within thousands of iterations where the accelerated version converges within a few hundred) for $q \gtrsim 15$; for the unknown-clusters classes, a similar pattern, growing from ~1.5-3x at small q to 5-9x by $q \approx$ 25-30.

Excluded in every variant: `sparsity > 0` (`estimate_omega()`'s graphical lasso is an approximate iterative solver, not an exact M-step argmax – even plain (V)EM's own objective trace is not strictly non-decreasing here, a numerical property of the solver, not of the acceleration; letting SQUAREM extrapolate into regions plain (V)EM would never reach was tested and found to amplify that instability for a much more modest speedup than the unpenalized case, since the regularization already smooths out most of the slow-convergence regime this targets – not worth it).

- **Code map.** (table on the next page, in landscape orientation – the full C++ template aliases are too long to fit a portrait page legibly; the Normal-Block rows below omit their common prefix, `norm_block_var_cov_` for the non-ZI variants and `zi_norm_block_var_cov_` for the ZI ones. The Mean-Block rows are given in full: unlike the Normal-Block family, `noise_covariance` is a runtime string there, not a `NoisePolicy` template parameter – see below – so there is one alias per known/unknown-clusters case, not one per noise shape.)

Variant	C++ class (suffix, after the common prefix)	Rcpp export
§1 known, diagonal	`diag_noise_known_clusters`	`NormalBlockVarKnownClusters_fit`
§2 known, spherical	`spherical_noise_known_clusters`	`NormalBlockVarKnownClusters_fit`
§3 unknown, diagonal	`diag_noise_unknown_clusters`	`NormalBlockVarUnknownClusters_fit`
§4 unknown, spherical	`spherical_noise_unknown_clusters`	`NormalBlockVarUnknownClusters_fit`
§6 ZI known, diagonal	`diag_noise_known_clusters`	`ZINormalBlockVarKnownClusters_fit`
§7 ZI known, spherical	`spherical_noise_known_clusters`	`ZINormalBlockVarKnownClusters_fit`
§8 ZI unknown, diagonal	`diag_noise_unknown_clusters`	`ZINormalBlockVarUnknownClusters_fit`
§9 ZI unknown, spherical	`spherical_noise_unknown_clusters`	`ZINormalBlockVarUnknownClusters_fit`
§10.1 mean, known	`norm_block_mean_known_clusters`	`NormalBlockMeanKnownClusters_fit`
§10.2 mean, unknown	`norm_block_mean_unknown_clusters`	`NormalBlockMeanUnknownClusters_fit`
§11 ZI mean, known	`zi_norm_block_mean_known_clusters`	`ZINormalBlockMeanKnownClusters_fit`
§11 ZI mean, unknown	`zi_norm_block_mean_unknown_clusters`	`ZINormalBlockMeanUnknownClusters_fit`

Both noise variants of a given clustering mode share one C++ template, parameterized by `NoisePolicy` (`NormalBlockVarKnownClusters`, `NormalBlockVarUnknownClusters`, `ZINormalBlockVarKnownClusters`, `ZINormalBlockVarUnknownClusters`), instantiated with `DiagonalNoise`/`SphericalNoise` or `ZIDiagonalNoise`/`ZISphericalNoise`; only the d/ξ update differs between the two. The Mean-Block classes (`NormalBlockMeanKnownClusters`, `NormalBlockMeanUnknownClusters`, and their ZI counterparts) are not templated this way: `cov_structure` (`"diagonal"`/`"spherical"`/`"full"`, §10.3) is a constructor argument checked at runtime, since it only changes how $\widehat{\Omega}$ is computed from the residuals (`omega_from_residuals()`), not the shape of the recursion itself.

19. Initialization and what a collection shares

All initialization (heuristic parameters, clustering, the upfront zero-inflation logistic regression) is done in R; the C++ classes only run the (V)EM recursion from parameters passed in – see the R6 classes (`NormalBlockVarKnownClusters`, `NormalBlockVarUnknownClusters`, `ZINormalBlockVarKnownClusters`, `ZINormalBlockVarUnknownClusters`) for the R-side heuristic initialization that supplies the starting values.

- **Initializing the unknown clustering (**C/τ, §§3-4, 8-9). A registry of cheap clustering heuristics, run once on the OLS/ZI residuals (on the fitted mean trajectory $X\widehat{B}$ for the mean-block family) before the first VE-step, supplies the starting τ: `kmeans`, `ward2` (hierarchical, Ward.D2 on $1 - \mathrm{cor}(R)$), `sbm` (a Gaussian SBM fit on $\mathrm{cov}(R)$, the closest theoretical match to the model's actual clustering target – a *shared block covariance*, not residual values), and `spectral` (k-means on the row-normalized top-$\min(q, \mathrm{rank})$ eigenvectors of $\mathrm{cov}(R)$, capped at $\mathrm{cov}(R)$'s numerical rank rather than q – eigenvectors past the rank are an arbitrary completion of the null space, not derived from the data, and $R = X\widehat{B}$ for the mean-block family is routinely rank $\leq d \ll q$; a cheap proxy for `sbm` otherwise). Selectable via `NB_control(clustering_init = ...)` (`private$clustering_methods` in `R/NormalBlockBase.R`); also accepts an explicit clustering directly, or a list of one per q for a collection. **The default is family-specific**, resolved from `NULL` in `NormalBlockVarBase$initialize()`/`NormalBlockMeanBase$initialize()` (and mirrored at the collection level by `clustering_path_for_family()`, `R/utils.R`): `ward2` for the variance-block family, `kmeans` for the mean-block one – the two families' tuning does not transfer, since they cluster genuinely different quantities (residual covariance vs. mean trajectory, §10's table).

  For the **variance-block family**, a benchmark across three real datasets found `ward2` gives the best balance of BIC rank and deviance-monotonicity reliability ($-2\times$ log-lik non-increasing in q, at fixed sparsity) – `kmeans` has a marginally better raw BIC rank, but far more and larger monotonicity violations. A fifth heuristic, `kmeansvar` (`ClustOfVar::kmeansvar`), ranked and performed worst on every dataset tested and was dropped from the registry entirely, removing the package's dependency on `ClustOfVar` – the same benchmark's hidden `ward2` fallback (used whenever a chosen heuristic collapses

to fewer than q clusters) used to be `ClustOfVar::hclustvar` for the same reason, also retired.

For the **mean-block family**, a separate benchmark across three datasets (`brca_rppa`, `onema`, and an $n = 300, p = 150, q = 10$ simulation) found `kmeans` consistently lands in a better ELBO basin than `ward2` or `spectral` for this family – not a quirk of one dataset.

- **A collection computes each heuristic's** q-independent part once, not once per q. `clustering_path_for_family()` (`R/utils.R`) resolves the family's heuristic and returns one clustering per q from a single shared computation: one hierarchical tree cut at each q for `ward2`, one eigendecomposition for `spectral`, one lossless row compression (`compress_columns()`, exact for `kmeans` since it preserves the inter-column distances that is all k-means sees) for `kmeans`, and – the original case – a single `sbm::estimateSimpleSBM()` exploration over the whole `q_list` range instead of once per q: a wide SBM exploration already visits every intermediate block count on its way to fitting the largest one, so its full path of clusterings comes "for free" with the cost of its single most expensive point. Benchmarked at ~15-90x cheaper than one independent exploration per q (depending on the range), for the same or statistically equivalent ICL after the (V)EM polish (`inst/clustering_initialization_benchmark/`). `estimateSimpleSBM()`'s own model selection is adaptive and can stop short of every q in `q_list` (or land on a clustering with an empty block, same general EM degeneracy as §0); any q it doesn't cover falls back to a single shared `ward2` clustering rather than a dedicated (and re-expensive) SBM call (and, more generally, any q whose shared computation fails to yield exactly q groups is handed back to that model's own `heuristic_clustering()`) – re-running one would silently reintroduce the cost this function exists to avoid, for no measured quality benefit.
- Y is rescaled column-wise by default before any of this. `NormalBlockData$new(Y, X, scale = TRUE)` (the default) divides each column of Y by its own standard deviation (no centering – the regression intercept/group-indicator columns a user is expected to include in X already absorb each variable's mean). Every clustering heuristic above therefore operates on a common scale across variables by default, which matters because the model's identifying assumption ($\mathrm{Cov}(Y_j, Y_{j'}) = \Omega_{kk}^{-1}$ for j, j' in the same block k, a single value shared by the whole block) is only meaningful when same-block variables don't have wildly different baseline variances. `B`, `dm1` and `fitted` are converted back to Y's original units (`B_original`/`dm1_original` active bindings, and the `fitted` binding directly, in every leaf class); $\widehat{\Omega}$ and the clustering itself are not (and structurally cannot be: the per-column scaling factors differ *within* a block, so there is no single way to "unscale" one shared block covariance value back to a $p \times p$ matrix on the original axis).
- **A post-hoc local search can refine a whole collection's clustering after the fact.** `NormalBlockCollectionClusters$refine()` tries, for every q in the collection (beyond its extremes), a single short split-and-reoptimize trial seeded from its already-fitted $q - 1$ neighbor and/or a single short merge-and-reoptimize trial seeded from its $q + 1$ neighbor (`split()`/`merge()`/`candidates_split()`/`candidates_merge()` in

`R/NormalBlockBase.R`), keeping a candidate only if it strictly improves the deviance. Motivation: every model in such a collection is cold-started independently (just above), which on real data can settle into a milder local optimum than an incremental neighbor-seeded search would – but, empirically, not in a way a cheap "did deviance increase" (or even a relative local-slope) diagnostic can reliably flag, since the gap can be spread uniformly across the whole q range rather than concentrated at an identifiable point; `refine()` therefore tries unconditionally everywhere and discards whatever doesn't help. `NB_control(refine = TRUE)` runs it automatically after `optimize()` (default `FALSE`, since it adds real cost); called directly, it matches or beats a from-scratch chained split/merge search (an earlier exploration strategy, retired, that grew a single collection by splitting/merging from one starting q) at a fraction of the cost, with no equivalent to that chained approach's failure mode of an early bad split/merge propagating to every larger q explored from it.

References

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2008. "Sparse Inverse Covariance Estimation with the Graphical Lasso." *Biostatistics* 9 (3): 432–41. https://doi.org/10.1093/biostatistics/kxm045.

Nglala Manguitini, Nestor, Jeanne Tous, and Julien Chiquet. 2026. *Modèle de Régression Log-Normale Multivarié Avec Clustering de Variables Intégré Dans La Moyenne*. MsC manuscript, available upon request.

Sustik, Mátyás A., and Ben Calderhead. 2012. *GLASSOFAST: An Efficient GLASSO Implementation*. TR-12-29. University of Texas at Austin.

Tous, Jeanne. 2026. "Graphical and Latent Variable Models for Species Community Modelling in Ecology." PhD thesis, Université Paris-Saclay.

Tous, Jeanne, and Julien Chiquet. 2026. "An Integrated Method for Clustering and Association Network Inference." *Computational Statistics & Data Analysis* 219: 108347. https://doi.org/10.1016/j.csda.2026.108347.

Varadhan, Ravi, and Christophe Roland. 2008. "Simple and Globally Convergent Methods for Accelerating the Convergence of Any EM Algorithm." *Scandinavian Journal of Statistics* 35 (2): 335–53. https://doi.org/10.1111/j.1467-9469.2007.00585.x.