Report: RT Regression Model (Hierarchical Ridge with Chemistry)

Bioinformatics Team

January 15, 2026

1 Introduction

Sally is Metabolon's production autocuration system for LC–MS peak assignment. For each sample set (SSID), it processes many LC tasks and determines, for each library compound, whether the compound is present and (if so) which observed peak best matches that compound. Within a single task, mass-based matching can produce multiple plausible candidate peaks, so Sally relies on additional evidence to reject decoys and to select the best remaining peak.

Retention time (RT) is a key signal in this decision. For each (task, compound) pair, Sally uses an RT regression model to predict an expected RT, expectedRT. It also attaches a window half-width, expectedRTWindow. Candidate peaks are filtered by this window:

$$|\text{apex\_rt} - \text{expectedRT}| \leq \text{expectedRTWindow}.$$

If multiple peaks remain for the same compound in a task, Sally chooses the peak closest to the predicted RT after applying a per-compound correction factor (the median RT shift across tasks within the SSID), and then removes outliers with a small fixed buffer around the corrected center. Because RT enters as a hard filter, the RT window is operationally important: windows that are too narrow remove true peaks (false negatives), while windows that are too wide allow decoys to survive and can increase false positives.

In current production, expectedRT comes from a bank of supercategory-specific lasso regression models (ESLASSO). Models are partitioned by species_matrix_type (called `species_cluster` in this report) and trained independently for each library compound using run-level covariates derived from internal standards (the `IS*`/`RS*`/`ES_` columns). Each compound model stores point-estimate coefficients and a fixed per-compound RT window derived from held-out residual variation; at scoring time the window is scaled by a global windowMultiplier and clamped to a minimum width. This design is practical and fast, and it captures broad matrix-specific RT structure and run drift. However, it is brittle in sparse settings: rare species and infrequently observed compounds receive limited statistical sharing, and the ℓ_1 penalty can behave poorly under correlated run covariates, leading to unstable coefficients across retrains. Moreover, fixed windows are heuristic and do not provide a calibrated predictive distribution, making it difficult to reason about coverage and filtering trade-offs systematically.

This report replaces the lasso baseline with a hierarchical Bayesian ridge regression with chemistry-informed sharing. The model partially pools species effects within each supercategory, regularizes compound baselines toward a chemistry-derived prior mean (from ChemBERTa embeddings), and uses ridge shrinkage to stabilize correlated run covariates. It yields a predictive distribution for RT, which we convert to task-specific

RT windows for filtering. To keep training tractable at production scale, we analytically collapse the per-group slopes and fit the remaining hierarchy with variational inference. For deployment, we export compact stage-1 coefficient summaries so that Sally can score deterministically at runtime without running Bayesian inference.

Finally, RT evidence is often strongest when aggregated across tasks. A false positive can survive the per-task RT window filter yet form an implausible across-task RT pattern for a compound. We therefore introduce an SSID-specific coherence check (peak_assignment_method = mixture_model) based on a two-component Bayesian Gaussian mixture fitted to per-compound mean RT log-likelihoods. The posterior probability of belonging to the coherent component provides a continuous, data-adaptive criterion for rejecting likely false positives without a fixed global threshold.

We evaluate both offline RT metrics and end-to-end peak assignment performance on held-out test sample sets, comparing the hierarchical model and coherence check against the production baseline.

2 Methods

This section describes the proposed RT model (hierarchical ridge with chemistry), the coherence check used for across-task rejection, and the baselines used for comparison. We start with a high-level decomposition and then give a complete probabilistic specification (likelihood, priors, and inference), followed by the production peak-assignment procedure that consumes RT predictions.

2.1 Data and notation

Each RT observation row i contains:

- response y_i (RT, minutes),
- run covariates $x_{i,j}$ for $j = 1, \dots, p$ from numeric `IS*/RS*/ES_` columns,
- identifiers: `species_cluster` $s(i)$, `species` $c(i)$, `comp_id` $k(i)$.

In this report, `species_cluster` corresponds to the production "supercategory" (matrix) grouping used to partition models in the baseline.

Each compound id k maps to a chemical identifier $h(k)$ (`compound/chem_id`), which has a fixed ChemBERTa PCA-20 embedding $e_{h(k),d}$ for $d = 1, \dots, D$ (with $D = 20$). ChemBERTa is a pretrained transformer model that reads SMILES strings (a text representation of chemical structure) and produces vector embeddings where chemically similar compounds tend to have similar representations. We use these embeddings to let the model share information across compounds when direct training support for a compound is sparse. The embeddings are built offline from SMILES strings using the pretrained ChemBERTa encoder and then projected to 20 dimensions with PCA for compactness; in this model they are treated as fixed inputs.

For numerical stability we treat the run covariates as globally centered ($x_{i,j} \leftarrow x_{i,j} - \bar{x}_{j,\text{train}}$ for each j). This is a pure reparameterization in a linear model with an intercept: it does not change the underlying fit, but it reduces intercept–slope coupling and improves conditioning.

2.2 High-level model decomposition

We model RT as:

$$\mathrm{RT} \;=\; \underbrace{\text{(species + compound baseline)}}_{\text{intercept}} \;+\; \underbrace{\text{(run adjustment)}}_{\text{slopes on run covariates}} \;+\; \underbrace{\text{noise}}_{\text{residual RT variation}} \quad .$$

Concretely, each (species, compound) pair defines a group

$$g = (c, k) = (\texttt{species}, \texttt{comp\_id}),$$

and we fit one linear regression per group:

$$y_i = b_{g(i)} + \sum_{j=1}^{p} x_{i,j}\, w_{g(i),j} + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

The model becomes useful in the long tail because we do *not* fit each $b_{c,k}$ and $w_{c,k,1:p}$ independently: we place hierarchical (partially pooled) priors on intercepts and slope means, and we tie compound effects to chemistry.

2.3 Model specification (at a glance)

For reference, this section summarizes the full generative model by separating observed quantities, the likelihood, and priors. We also spell out the indexing to make clear what is shared across supercategories, species, compounds, and observations.

We use the following indices and mappings:

- observation index $i \in \{1, \dots, N\}$ for RT rows,
- run-covariate index $j \in \{1, \dots, p\}$,
- supercategory index s (production `species_cluster`),
- species index c (nested in a supercategory),
- compound index k (`comp_id`),
- chemical id $h = h(k)$ (`chemical_id`),
- embedding dimension $d \in \{1, \dots, D\}$ (with $D = 20$).

Row i belongs to a group $g(i) = (c(i), k(i))$ and to a supercategory $s(i)$. The mapping $s(c)$ returns the supercategory containing species c, and the mapping $h(k)$ returns the chemical id for compound k. For chemistry, $z_{h,d} = e_{h,d} - \bar{e}_{d,\text{train}}$ denotes the centered ChemBERTa embedding for chemical id h.

We observe $(y_i, x_{i,1:p}, s(i), c(i), k(i))$ for $i = 1, \dots, N$, and we treat centered ChemBERTa embeddings $z_{h(k),1:D}$ as fixed inputs. Hyperparameters such as λ_{slopes} and prior scales ($\sigma_\bullet$) are fixed.

The main model parameters are:

- σ_y: global residual noise scale for RT (shared across all observations),
- t_0: global RT baseline (supercategory-wide intercept),
- μ_s: supercategory-level intercept offset, and μ_c: species-level offset within supercategory,
- α_k: compound baseline effect (shared across species) with a chemistry-informed prior mean,
- $b_{c,k}$: group intercept for each (species, compound) pair, which combines t_0, μ_c, and α_k,
- $w_{c,k,1:p}$: group slopes for run covariates (per (species, compound)), centered around hierarchical means,
- w_0, m_s, m_c: global, supercategory, and species slope means, respectively,
- τ scale parameters (e.g. $\tau_{\mu,s}$, $\tau_{w,s}$, τ_{comp}, τ_b) that control the strength of partial pooling (smaller τ means stronger shrinkage toward the parent level).

These parameters are shared at different levels:

- global: t_0, σ_y, $w_{0,1:p}$, $\theta_{\alpha,1:D}$, λ_{slopes}, and the scale parameters $\tau_{\mu,\text{supercat}}$, $\tau_{w,\text{supercat}}$, τ_{comp}, and τ_b,
- per-supercategory: μ_s, $m_{s,1:p}$, $\tau_{\mu,s}$, and $\tau_{w,s}$,
- per-species: μ_c and $m_{c,1:p}$,
- per-compound: α_k (with residual δ_k),
- per-(species, compound) group: $b_{c,k}$ and $w_{c,k,1:p}$.

For each row i, we model the observed RT y_i as a noisy observation of a group-specific linear predictor. The group is the (species, compound) pair $g(i) = (c(i), k(i))$, and $w_{g(i),j}$ are the group slopes on centered run covariates $x_{i,j}$. Conditional on the group coefficients, observations are independent.

$$y_i \mid b_{g(i)}, w_{g(i),1:p}, \sigma^2 \sim \mathcal{N}\left(b_{g(i)} + \sum_{j=1}^{p} x_{i,j}\, w_{g(i),j}, \sigma^2\right). \tag{1a}$$

We assume a single residual noise scale σ_y shared across all observations. We use a HalfNormal prior to enforce positivity and to discourage implausible noise values. We write $\sigma^2 = \sigma_y^2$ for convenience.

$$\sigma_y \sim \text{HalfNormal}(\sigma_{y,\text{prior}}), \qquad \sigma^2 = \sigma_y^2. \tag{1b}$$

We model systematic baseline RT differences across supercategories and species using additive offsets. Each supercategory has an offset μ_s, and each species has an offset μ_c that is shrunk toward its supercategory mean $\mu_{s(c)}$. This implements partial pooling: species with little data are pulled toward the supercategory mean. We allow each supercategory to have its own pooling strength through $\tau_{\mu,s}$. In implementation we center these offsets to have mean zero so that t_0 remains the overall RT baseline.

$$\begin{aligned}\tau_{\mu,\text{supercat}} &\sim \text{HalfNormal}(\sigma_{\tau_\mu}),\\ \tau_{\mu,s} &\sim \text{HalfNormal}(\sigma_{\tau_\mu}),\\ \mu_s \mid \tau_{\mu,\text{supercat}} &\sim \mathcal{N}(0, \tau^2_{\mu,\text{supercat}}),\\ \mu_c \mid \mu_{s(c)}, \tau_{\mu,s(c)} &\sim \mathcal{N}(\mu_{s(c)}, \tau^2_{\mu,s(c)}).\end{aligned} \tag{1c}$$

The compound effect α_k is decomposed into a chemistry-driven prior mean plus a residual. The embedding weights $\theta_{\alpha,1:D}$ learn a global mapping from ChemBERTa space to RT baselines, and δ_k allows each compound to deviate from the chemistry-predicted mean when the data support it. The residual scale τ_{comp} controls how strongly we trust the chemistry prior mean.

Chemistry only helps when a compound can be mapped from `comp_id` to a `chem_id` with a ChemBERTa embedding (derived from a SMILES string). When this mapping or embedding is missing, we fall back to the mean embedding; after centering, this is equivalent to setting $z_{h(k),1:D} = 0$, so the chemistry-derived part of α_k is 0. In that case, compounds seen in training can still deviate through the residual δ_k, while unseen compounds have a zero chemistry backoff.

$$\begin{aligned}\tau_{\text{comp}} &\sim \text{HalfNormal}(\sigma_{\tau_{\text{comp}}}),\\ \theta_{\alpha,d} &\sim \mathcal{N}(0, \sigma^2_\theta), \qquad d = 1, \dots, D,\\ \delta_k \mid \tau_{\text{comp}} &\sim \mathcal{N}(0, \tau^2_{\text{comp}}),\\ \alpha_k &= \sum_{d=1}^{D} z_{h(k),d}\,\theta_{\alpha,d} + \delta_k.\end{aligned} \tag{1d}$$

The intercept $b_{c,k}$ is the baseline RT for compound k in species c. We draw it around the additive baseline $t_0 + \mu_c + \alpha_k$ with residual scale τ_b, which captures remaining between-(species, compound) variation not explained by the species and compound terms.

$$\begin{aligned}t_0 &\sim \mathcal{N}(0, \sigma^2_{t0}),\\ \tau_b &\sim \text{HalfNormal}(\sigma_{\tau_b}),\\ b_{c,k} \mid t_0, \mu_c, \alpha_k, \tau_b &\sim \mathcal{N}(t_0 + \mu_c + \alpha_k, \tau^2_b).\end{aligned} \tag{1e}$$

Run covariates (e.g., internal-standard intensities) shift RT, and these effects can vary by species. We model this by placing hierarchical priors on slope means: a global mean $w_{0,j}$, a supercategory mean $m_{s,j}$, and a species mean $m_{c,j}$. This is another form of partial pooling: species slope means are pulled toward their supercategory means, which stabilizes estimates for rare species and for correlated covariates.

$$\begin{aligned}\tau_{w,\text{supercat}} &\sim \text{HalfNormal}(\sigma_{\tau_w}),\\ \tau_{w,s} &\sim \text{HalfNormal}(\sigma_{\tau_w}),\\ w_{0,j} &\sim \mathcal{N}(0, \sigma^2_{w0}), \qquad j = 1, \dots, p,\\ m_{s,j} \mid w_{0,j}, \tau_{w,\text{supercat}} &\sim \mathcal{N}(w_{0,j}, \tau^2_{w,\text{supercat}}),\\ m_{c,j} \mid m_{s(c),j}, \tau_{w,s(c)} &\sim \mathcal{N}(m_{s(c),j}, \tau^2_{w,s(c)}).\end{aligned} \tag{1f}$$

Within each (species, compound) group, the slope $w_{c,k,j}$ is shrunk toward the species mean $m_{c,j}$ with strength controlled by λ_{slopes} (larger values mean stronger shrinkage). We use a ridge (Gaussian) prior because run covariates are often highly correlated: ridge shrinkage tends to distribute weight smoothly across correlated predictors and gives more stable fits, whereas lasso-style sparsity can arbitrarily pick one covariate and lead to unstable coefficients.

$$w_{c,k,j} \mid m_{c,j}, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}(m_{c,j}, \sigma^2/\lambda_{\text{slopes}}), \qquad j = 1, \dots, p. \tag{1g}$$

2.4 Generative process

Conditioned on fixed inputs $(x_{i,1:p}, z_{h,1:D})$, the model defines a generative process: it specifies a step-by-step way one could simulate RT observations by drawing parameters and then generating y_i.

- draw global scales and baselines $(\sigma_y, t_0, w_{0,1:p})$ and chemistry weights $\theta_{\alpha,1:D}$,
- for each supercategory s, draw offsets and slope means $(\mu_s, m_{s,1:p})$,
- for each species c, draw offsets and slope means around its supercategory $(\mu_c, m_{c,1:p})$,
- for each compound k, compute a chemistry-informed baseline α_k,
- for each (species, compound) pair (c, k), draw group coefficients $(b_{c,k}, w_{c,k,1:p})$,
- for each observation i, generate y_i from a Normal distribution with mean $b_{g(i)} + \sum_j x_{i,j} w_{g(i),j}$ and variance σ^2.

We assume RT is an additive linear function of run covariates plus noise, and that observations are independent once we condition on the group coefficients. We use one shared noise scale σ_y. Hierarchical priors provide the main information sharing: species offsets and slope means are pulled toward their supercategory means, with the τ pooling scales controlling how strong that pooling is. For slopes we use a ridge (Gaussian) prior so correlated run covariates behave stably. Chemistry enters only through α_k: embeddings define a prior mean, and δ_k allows the data to override that mean when there is enough support.

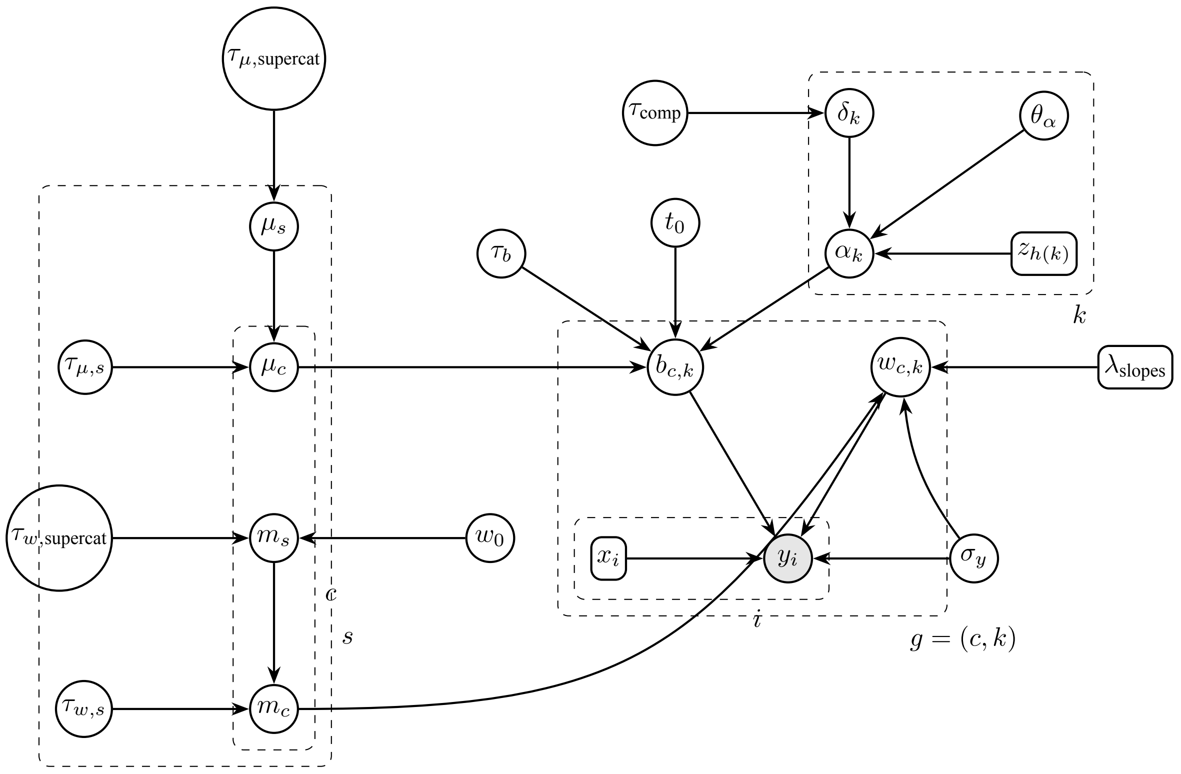

Figure 1: Plate diagram for the partial pooling RT ridge model. Shaded nodes are observed. Vector-valued nodes implicitly range over run covariates $(1:p)$ or embedding dimensions $(1:D)$. In production, the per-group slopes $w_{c,k}$ are integrated out (collapsed ridge) during inference and then recovered in closed form after fitting the hyperparameters.

2.5 Inference and predictive uncertainty

We fit the model with variational inference (ADVI), which approximates the posterior over the hierarchical parameters with a tractable distribution (rather than running full MCMC). This makes training feasible at production scale.

The dominant parameter count in a fully explicit model would be the per-group slope coefficients $w_{c,k,1:p}$ (one set per `(species, comp_id)` pair). Instead of representing all of these slopes as latent nodes in the PyMC graph, we integrate them out analytically. Conditional on the ridge prior and the run covariates, the slopes can be integrated out, yielding a collapsed (marginal) likelihood. In practice we precompute small per-group sufficient statistics, such as $S^{xx}_{g,j\ell} = \sum_{i:g(i)=g} x_{i,j} x_{i,\ell}$ and $S^{xy}_{g,j} = \sum_{i:g(i)=g} x_{i,j} y_i$, and use the resulting collapsed (marginal) likelihood inside ADVI. This replaces a very large set of per-row computations with compact per-group summaries, which is the main speedup. Appendix B derives the collapsed likelihood and the corresponding closed-form posterior for $w_{g,1:p}$. We still represent per-group intercepts $b_{c,k}$ explicitly because they are tied to the hierarchical prior mean $t_0 + \mu_c + \alpha_k$.

After ADVI fits global and hierarchical parameters (including the intercept hierarchy and the slope-mean hierarchy), we recover a Gaussian posterior for each group's coefficients in closed form and export per-group summaries for $\beta_{c,k,0} = b_{c,k}$ and $\beta_{c,k,j} = w_{c,k,j}$ for $j = 1, \ldots, p$ (posterior mean and covariance). This exported artifact is what we use in production scoring: it avoids running Bayesian inference at runtime while still providing an explicit uncertainty estimate for RT filtering.

These group-wise posteriors are computed conditional on fitted hyperparameters (typically at their posterior means under ADVI), rather than fully integrating over hyperparameter uncertainty.

The group-wise coefficient posterior depends on global and hierarchical parameters such as t_0, the pooling scales ($\tau_{\mu,\text{supercat}}$, $\tau_{\mu,s}$, $\tau_{w,\text{supercat}}$, $\tau_{w,s}$), the residual scales (τ_{comp}, τ_b), chemistry weights $\theta_{\alpha,1}, \ldots, \theta_{\alpha,D}$, and the noise scale σ_y. A fully Bayesian prediction would average over uncertainty in these quantities; we instead "plug in" their fitted values and compute the conditional Gaussian posterior for each group in closed form. This is a plug-in (point-estimate) approximation (sometimes called empirical Bayes): it makes scoring simple and deterministic, but it can slightly understate uncertainty if hyperparameters are themselves uncertain.

At scoring time, for a new row i we select the coefficient-summary group $g(i)$, use posterior mean coefficients for a point prediction, and propagate coefficient uncertainty into a predictive variance:

$$\hat{y}_i = b_{g(i)} + \sum_{j=1}^{p} x_{i,j}\, w_{g(i),j}, \qquad \widehat{\text{Var}}(y_i \mid x_{i,1:p}) \approx \sigma^2_{g(i)} + \sum_{a=0}^{p}\sum_{b=0}^{p} \tilde{x}_{i,a}\tilde{x}_{i,b}\, \widehat{\text{Cov}}(\beta_{g(i),a}, \beta_{g(i),b}),$$

where $\tilde{x}_{i,0} = 1$ and $\tilde{x}_{i,j} = x_{i,j}$ for $j = 1, \ldots, p$. In the hierarchical ridge model σ_g^2 is shared across groups, while the sklearn supercategory ridge baseline estimates a separate σ_g^2 per `(species_cluster, comp_id)` pair. With $\hat{\sigma}_i = \sqrt{\widehat{\text{Var}}(y_i \mid x_{i,1:p})}$, a nominal 95% RT window is $\hat{y}_i \pm k_{0.95}\, \hat{\sigma}_i$ (width $2 \cdot k_{0.95} \cdot \hat{\sigma}_i$), where $k_{0.95} = \Phi^{-1}(0.975) \approx 1.96$ under a Normal approximation. This window can vary by group and by covariates because it includes both residual noise and coefficient uncertainty. Downstream peak assignment uses this interval as a filter: a candidate peak is retained when its observed RT falls within the model's window around $\hat{y}_i$.

For contrast, the production lasso baseline attaches a fixed window estimated from held-out residual variation (described in Section 2.8).

2.6 Alternative model: supercategory ridge (sklearn)

As a fast baseline, we fit an independent ridge regression for each group (`species_cluster`, `comp_id`) using the same run covariates. This model has no hierarchy across `species` and no chemistry: each supercategory–compound pair is fit essentially independently, with an ℓ_2 penalty for stability under correlated covariates. We attach an approximate Normal predictive distribution by combining an estimated noise scale with an approximate coefficient covariance.

For each `(species_cluster, comp_id)` group this baseline stores a fitted coefficient mean, an approximate coefficient covariance, and an estimated residual variance σ_g^2. For a new row, we compute $\hat{y}_i$ and $\hat{\sigma}_i$ and use the nominal 95% window $\hat{y}_i \pm k_{0.95}\, \hat{\sigma}_i$ (as defined above). This yields a row-specific window rather than a fixed width per model.

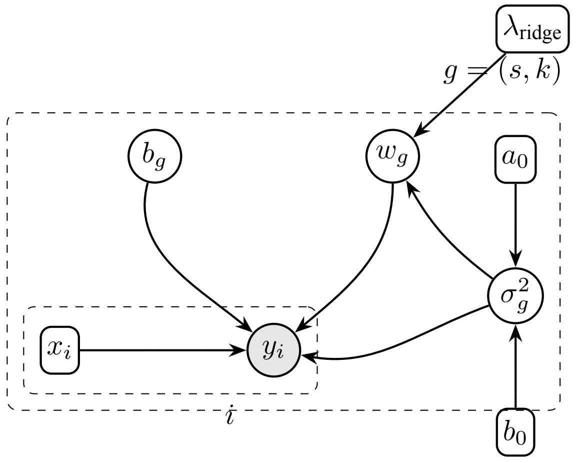

Figure 2: Plate diagram for the sklearn supercategory ridge baseline. Each group $g = (s, k)$ (species_cluster, comp_id) is fit independently with ridge shrinkage on slopes w_g (intercept b_g unpenalized). The exported artifact stores a coefficient mean, an approximate coefficient covariance, and a per-group residual variance estimate σ_g^2 (from an inverse-gamma noise model with hyperparameters a_0, b_0).

2.7 Baseline currently in production: supercategory lasso

The production baseline is a bank of lasso regressions fit per `(species_cluster, comp_id)` pair. Lasso is attractive for sparse solutions, but its ℓ_1 prior is a poor match to correlated run covariates and it does not support partial pooling or chemistry-informed sharing across compounds. Its "interval" is a fixed window, not a probabilistic prediction interval.

Each model stores point-estimate coefficients and a per-compound window scale derived from held-out residual variation. At scoring time, Sally uses this to form an RT window half-width expectedRTWindow and then applies it as an RT filter inside peak selection (described in the next subsection). Because this baseline does not expose a calibrated predictive distribution, the window is best viewed as a heuristic tolerance rather than a nominal prediction interval.

2.8 Baseline peak assignment in Sally: RT window filter + correction factor

Sally uses RT predictions in a deterministic per-task peak selection procedure. For a task t and compound c, the RT model provides an expected RT $\hat{\mu}_{t,c}$ and an RT window half-width $w_{t,c}$. A candidate peak with apex RT r passes the RT filter when

$$|r - \hat{\mu}_{t,c}| \le w_{t,c}.$$

For the production lasso baseline (ESLASSO), each compound model stores point-estimate coefficients and a base window $w_{\text{base},c}$ computed from held-out residuals as

$$w_{\text{base},c} = 6 \cdot \widehat{\text{sd}}(\hat{y} - y),$$

and the effective half-width used for filtering is scaled and clamped:

$$w_{t,c} = m \cdot \max(w_{\text{base},c}, w_{\text{min}}),$$

with default $m = 4$ and $w_{\text{min}} = 0.001$ minutes. For COMPASSIGN_PP_RIDGE, $w_{t,c}$ is derived from a predictive distribution and varies by task (Section 1 and the predictive-variance formula in the inference section).

After RT filtering, some tasks can still contain multiple candidate peaks for the same compound. Sally resolves this by computing an SSID-specific correction factor and then choosing the peak closest to the corrected center. For a retained peak in task t for compound c, define the signed residual

$$\Delta_{t,c} = \hat{\mu}_{t,c} - r.$$

The correction factor is the median shift across tasks where the compound is observed:

$$\widehat{\text{CF}}_c = \text{median}_{t \in \mathcal{T}_c} \Delta_{t,c}.$$

Within each task, Sally keeps the peak with smallest corrected distance $|\Delta_{t,c} - \widehat{\text{CF}}_c|$ and rejects the others. Finally, it removes outliers by keeping peaks within a fixed buffer

$$|\Delta_{t,c} - \widehat{\text{CF}}_c| \leq b, \qquad b = 0.02 \text{ minutes.}$$

The optional iqr method modifies only the centering step by restricting to a tighter "central" cloud when computing $\widehat{\text{CF}}_c$, but it applies the same fixed buffer after centering.

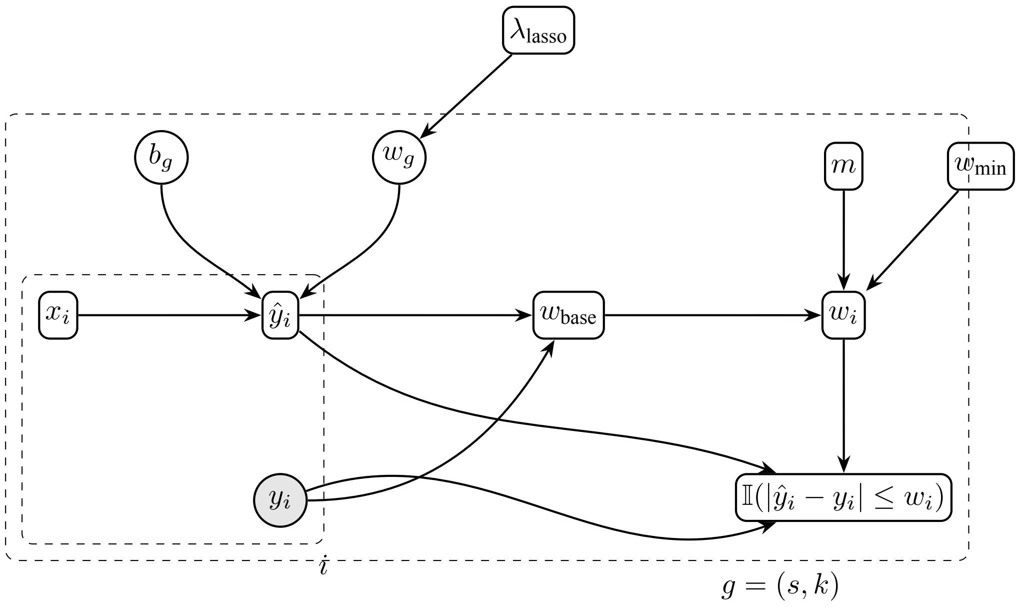

Figure 3: Schematic for the production supercategory lasso baseline (ESLASSO): each group $g = (s, k)$ (species_cluster, comp_id) stores point-estimate coefficients (b_g, w_g) and a base window w_{base} from held-out residuals; scoring scales and clamps it to $w_i = m \max(w_{\text{base}}, w_{\text{min}})$ and filters candidate peaks by the RT window.

2.9 RT coherence rejection via Bayesian Gaussian mixture

Sally's per-task selection is intentionally local: it applies a hard RT window filter, then picks the peak closest to the (corrected) expected RT. This removes gross decoys, but some false positives still look plausible on individual tasks and only become clearly inconsistent when aggregated across tasks for the same compound. We therefore add a *compound-level* coherence check for COMPASSIGN_PP_RIDGE that uses a continuous RT likelihood signal across tasks.

The default setting is peak_assignment_method = mixture_model. Conceptually, it is a two-component Bayesian Gaussian mixture model fitted *within each SSID* to a single RT evidence score per compound.

The coherence rejection can be applied to any RT model that provides an expected mean and an RT window. In this report we evaluate it with both COMPASSIGN_PP_RIDGE and ESLASSO. For ESLASSO, the score should be interpreted as a pseudo-likelihood because the window is heuristic rather than a calibrated predictive sd.

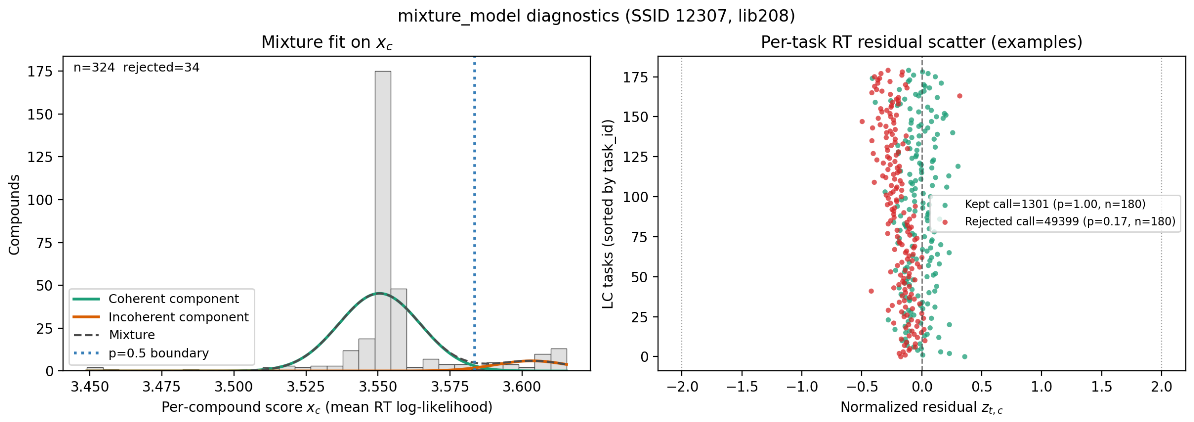

Figure 4: Example mixture_model coherence check for SSID 12307 (lib208). Left: histogram of per-compound scores x_c with the fitted two-component mixture and the $p(\text{coherent} \mid x_c) = 0.5$ decision boundary. Right: per-task normalized residuals $z_{t,c}$ for an example *kept* compound and an example *rejected* compound, illustrating coherent vs. incoherent across-task behavior.

1. *Per-peak RT log-likelihood.* After selecting a peak for compound c in task t, let $r_{t,c}$ be the observed apex RT and let the RT model provide an expected mean $\hat{\mu}_{t,c}$ and window half-width $w_{t,c}$. We convert the half-width to an sd-like scale using the nominal 95% relationship $\sigma_{t,c} = w_{t,c}/z_{0.975}$ (after applying any global window multiplier). We also apply Sally's per-compound correction factor $\widehat{\mathrm{CF}}_c$ (median RT shift across tasks within the SSID). Define the signed residual $\Delta_{t,c} = \hat{\mu}_{t,c} - r_{t,c}$ and the standardized residual and RT log-likelihood score as

$$z_{t,c} = \frac{\Delta_{t,c} - \widehat{\mathrm{CF}}_c}{\sigma_{t,c}}, \qquad \ell_{t,c} = \log p(r_{t,c} \mid \hat{\mu}_{t,c}, \sigma_{t,c}) = -\tfrac{1}{2}\left(z_{t,c}^2 + \log \sigma_{t,c}^2\right),$$

   where $\ell_{t,c}$ is the Normal log predictive density up to an additive constant. In code this is computed as an internal feature (stored as `_rt_loglik` during the mixture fit).

2. *One scalar score per compound.* For each compound c in an SSID, we aggregate over LC tasks with a selected peak and define

$$x_c = \frac{1}{n_c} \sum_{t \in \mathcal{T}_c} \ell_{t,c},$$

   where $\mathcal{T}_c$ is the set of contributing tasks and $n_c = |\mathcal{T}_c|$. This compresses the across-task RT evidence for a compound into a single scalar that can be modeled across compounds.

3. *Two-component Gaussian mixture within an SSID.* Within a fixed SSID, we model the set of compound scores $\{x_c\}$ as coming from a mixture of two Gaussian populations:

$$z_c \sim \text{Categorical}(\pi), \qquad x_c \mid z_c = k \sim \mathcal{N}(\mu_k, \sigma_k^2), \qquad k \in \{1, 2\}.$$

   The mixture is over compounds within the SSID: each observation is a compound-level summary statistic x_c (already aggregated across tasks). We use conjugate priors

$$\pi \sim \text{Dirichlet}(\alpha_0, \alpha_0), \qquad \tau_k \equiv \sigma_k^{-2} \sim \text{Gamma}(a_0, b_0), \qquad \mu_k \mid \tau_k \sim \mathcal{N}\big(m_0,\ (\kappa_0 \tau_k)^{-1}\big),$$

   which is the 1D Normal–Gamma form of the Normal–Wishart prior used by variational Bayesian Gaussian mixture models. In code we set $\alpha_0 = 0.5$ (encourages, but does not force, an imbalanced mixture) and use a small covariance regularizer; remaining hyperparameters follow the library defaults.

4. *Inference outputs.* We infer an approximate posterior over component parameters and compound memberships. In particular, variational Bayes produces responsibilities

$$r_{c,k} \equiv p(z_c = k \mid x_c), \qquad \sum_k r_{c,k} = 1,$$

   along with posterior summaries for π, μ_k, and σ_k^2. Implementation uses scikit-learn's variational Bayesian Gaussian mixture implementation.

5. *Identifying the coherent component.* The fitted mixture produces two clusters with weights π_k and means μ_k. After the per-task RT window filter, most compounds are expected to be coherent, so we label the coherent cluster as the one with the larger fitted weight:

$$k^\star = \arg\max_k \pi_k.$$

   The corresponding mean $\mu_{k^\star}$ summarizes typical coherent RT evidence in that SSID, while the other component captures a lower-likelihood population. Together $(\pi_k, \mu_k, \sigma_k^2)$ define an implicit, data-driven boundary for "coherent enough" without a fixed global threshold.

6. *Rejection rule.* For each compound we compute its posterior coherence probability $p_c = r_{c,k^\star}$. If $p_c < 0.5$ we reject the compound (MAP under equal misclassification costs) by removing all of its selected peaks from `bundle.peaks`; the standard Sally pipeline then marks the compound `IGNORE`.

The score x_c is a *likelihood* under a Normal model whose scale $\sigma_{t,c}$ is derived directly from the RT window $w_{t,c}$. For COMPASSIGN_PP_RIDGE, $w_{t,c}$ is computed from a probabilistic predictive variance and varies by task (2D window matrix), so $z_{t,c}$ and $\ell_{t,c}$ behave like a calibrated standardized residual and log predictive density. This makes x_c a meaningful continuous coherence signal. For ESLASSO, the RT window is a per-compound heuristic (1D; constant across tasks) rather than an inferred predictive sd, so after centering by $\widehat{\text{CF}}_c$ the compound score x_c is dominated by the window term $-\frac{1}{2}\log\sigma_{t,c}^2$ rather than residual coherence (in our runs, $\text{corr}(x_c, \overline{\log\sigma^2}) \approx -1$). The resulting mixture primarily separates compounds by their heuristic window width, which can reject many true compounds and reduce recall.

3 Evaluation

We compare models trained on the capped `cap100` dataset and evaluated on the held-out test split for lib208 and lib209. This report focuses on the standard seen-compound evaluation where models score nearly all rows and the comparison is driven by accuracy and calibrated windows rather than missing-model fallbacks. To focus on *model form* rather than feature engineering, we use the same linear run covariates for all models (no polynomial expansions).

3.1 Data preparation

All offline training and evaluation inputs are built by `src/compassign/rt/prep.sh`. Starting from Pachyderm, it produces merged Parquets under `repo_export/merged_training/` and then derives the training caps and evaluation CSVs used throughout this report. In outline, the pipeline is:

- *Fetch training shards from Pachyderm.* Use pachctl get file to download create_training_data and combine_predictors outputs for each library from fixed job IDs (project autocuration_platinum) into `repo_export/pachyderm_rt_training_raw/`.
- *Assemble a combined export tree.* Copy the per-library shard trees into a stable layout under `repo_export/pachyderm_rt_training_latest/`.
- *Merge shards into a single Parquet.* Run `src/compassign/rt/data_prep/merge_pachyderm_training.py` to create `repo_export/merged_training/merged_training_all.parquet`.
- *Split by library.* Run `src/compassign/rt/data_prep/split_merged_by_lib.py` to produce per-library Parquets under `repo_export/merged_training/`.
- *Regenerate strict species mappings.* Run `src/compassign/rt/data_prep/check_rt_metadata_mapping.py` to ensure a consistent mapping from `sample_set_id` to `species` and `species_cluster`.
- *Build capped training datasets (`cap100`).* For each (supercategory, species, compound) group, keep at most 100 rows using uniform random sampling with a fixed seed (42) via `src/compassign/rt/data_prep/build_rt_cap_datasets.sh` (which calls `src/compassign/rt/data_prep/sample_per_species_compound.py`). These capped CSVs are the training inputs for all reported models.
- *Build held-out test CSVs (evaluation input).* Filter the merged Parquet to SSIDs assigned to the `test` fold in `data/split_outputs/train_test_split_all.csv` and convert each SSID to RT production-style CSVs via `src/compassign/rt/data_prep/build_rt_real_test_csvs.py`.

3.2 Metrics

Let y_i be observed RT and $\hat{y}_i$ the point prediction (minutes).

- RMSE: $\sqrt{\frac{1}{n}\sum_i(\hat{y}_i - y_i)^2}$.
- MAE: $\frac{1}{n}\sum_i |\hat{y}_i - y_i|$.
- Cov95 (ridge models): define predicted variance as

$$\widehat{\mathrm{Var}}(y_i \mid x_{i,1:p}) \approx \sigma^2_{g(i)} + \sum_{a=0}^{p}\sum_{b=0}^{p} \tilde{x}_{i,a}\tilde{x}_{i,b}\,\widehat{\mathrm{Cov}}\big(\beta_{g(i),a}, \beta_{g(i),b}\big),$$

with predicted standard deviation $\hat{\sigma}_i = \sqrt{\widehat{\mathrm{Var}}(y_i \mid x_{i,1:p})}$ and $k_{0.95} = \Phi^{-1}(0.975) \approx 1.96$. We define the nominal 95% interval as $\hat{y}_i \pm k_{0.95}\,\hat{\sigma}_i$ and count row i as "covered" if $|\hat{y}_i - y_i| \le k_{0.95}\,\hat{\sigma}_i$; Cov95 is the empirical fraction of covered rows. The terms $\sigma^2_{g(i)}$ and $\widehat{\mathrm{Cov}}(\beta_{g(i)})$ come from the exported stage-1 coefficient summaries: for the sklearn ridge baseline they are computed analytically from ridge normal equations with an inverse-gamma noise prior, while for the partial pooling model they are computed in closed form conditional on fitted hierarchical hyperparameters (an empirical-Bayes plug-in approximation).

- Width95 (ridge models): $\frac{1}{n}\sum_i 2 \cdot k_{0.95} \cdot \hat{\sigma}_i$.

In the expressions above, $g(i)$ denotes the coefficient-summary group for row i, and we use the augmented covariates $\tilde{x}_{i,0} = 1$ and $\tilde{x}_{i,j} = x_{i,j}$ for $j = 1, \ldots, p$. For the production lasso baseline, the "window" is a fixed filtering half-width derived from held-out residuals and then scaled by a global multiplier (see Section 2.7). This window is not calibrated to a nominal 95% target, so we do not report it as Cov95/Width95.

4 Results: offline RT regression

4.1 Global performance

Table 1 summarizes global metrics on the held-out test split for lib208 and lib209. Lasso baselines do not score all rows; the Rows scored column reflects applicability.

Table 1: Global held-out test metrics for cap100-trained models (linear features). Cov95/Width95 report empirical coverage and mean width of the nominal 95% Normal interval $\hat{y}_i \pm k_{0.95}\,\hat{\sigma}_i$ and are only defined for ridge models with an explicit predictive variance. The lasso baseline uses a heuristic fixed filtering window; its Cov95/Width95 entries are omitted (–) to avoid implying nominal calibration.

Lib	Model	RMSE	MAE	Cov95	Width95	Rows scored	Train (min)
208	Ridge (supercategory)	0.009050	0.004684	0.944	**0.023573**	**100.0%**	**0.0**
208	Ridge (partial pooling)	**0.007846**	**0.003817**	**0.958**	0.028744	**100.0%**	131.0
208	Lasso (supercategory)	0.015081	0.007956	–	–	94.7%	–
209	Ridge (supercategory)	0.008295	0.004631	0.919	**0.020994**	**100.0%**	**0.0**
209	Ridge (partial pooling)	**0.007589**	**0.004203**	**0.957**	0.027094	**100.0%**	242.3
209	Lasso (supercategory)	0.009439	0.004954	–	–	97.8%	–

Relative to Ridge (supercategory), Ridge (partial pooling) improves RMSE while moving coverage closer to the nominal 0.95 target. This is desirable for peak assignment because under-coverage translates directly into missed true peaks when the window is used as a hard filter. The lasso baseline is included for context: it does not score all rows and its uncertainty is a fixed window rather than a probabilistic interval.

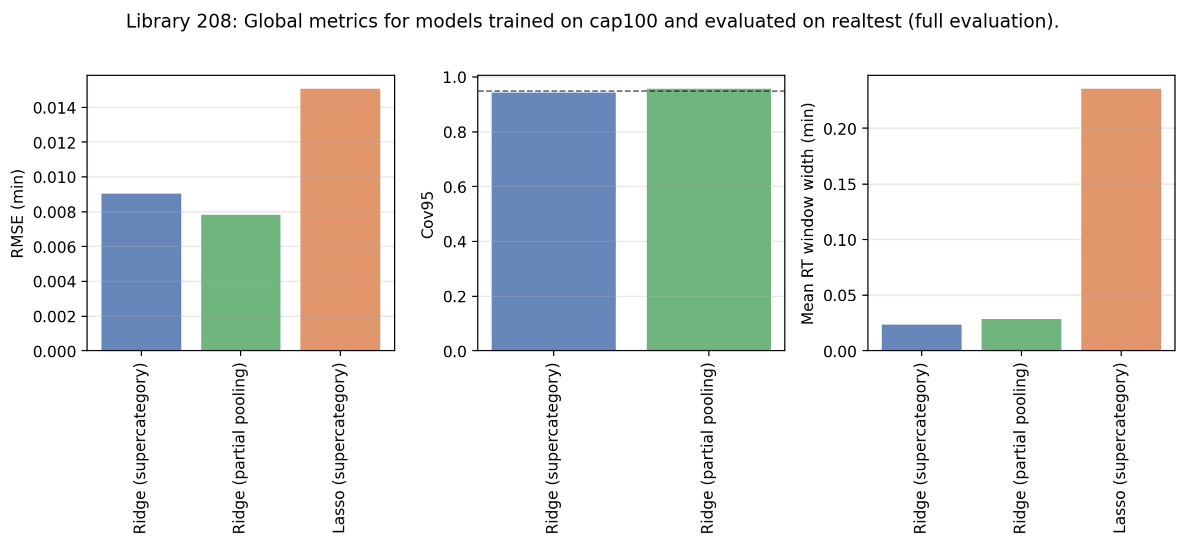

Figure 5: Global comparison across report baselines for lib208 (cap100 training, held-out test evaluation). Panels show RMSE, Cov95 (ridge models only; lasso omitted), and mean RT window width (Width95 for ridge models; production filtering window for lasso). Tighter windows are only meaningful when coverage is comparable.

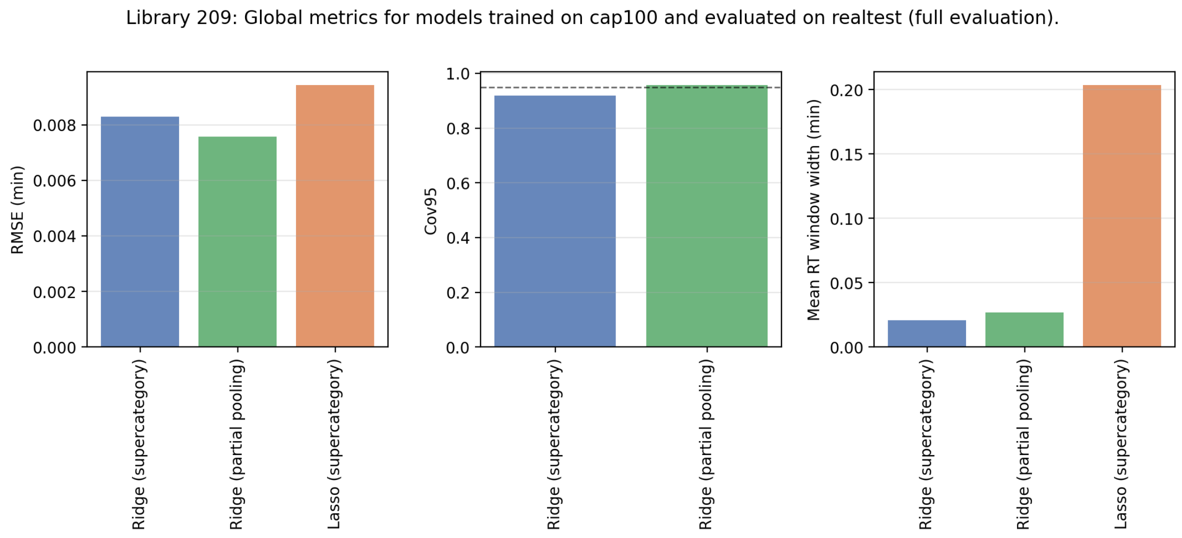

Figure 6: Global comparison across report baselines for lib209 (cap100 training, held-out test evaluation). Panels show RMSE, Cov95 (ridge models only; lasso omitted), and mean RT window width (Width95 for ridge models; production filtering window for lasso). Tighter windows are only meaningful when coverage is comparable.

4.2 Interval calibration: supercategory ridge vs partial pooling

The sklearn ridge baseline and the proposed hierarchical ridge model are both linear in the same run covariates, but they differ in how information is shared across sparse groups and in how predictive variance is formed. Empirically, the sklearn baseline produces narrower windows but under-covers the nominal 95% target (coverage ∼0.92–0.94). The hierarchical model yields coverage closer to nominal at moderately larger width.

4.3 Performance by species_cluster

Figures 7 and 8 aggregate metrics by `species_cluster`.

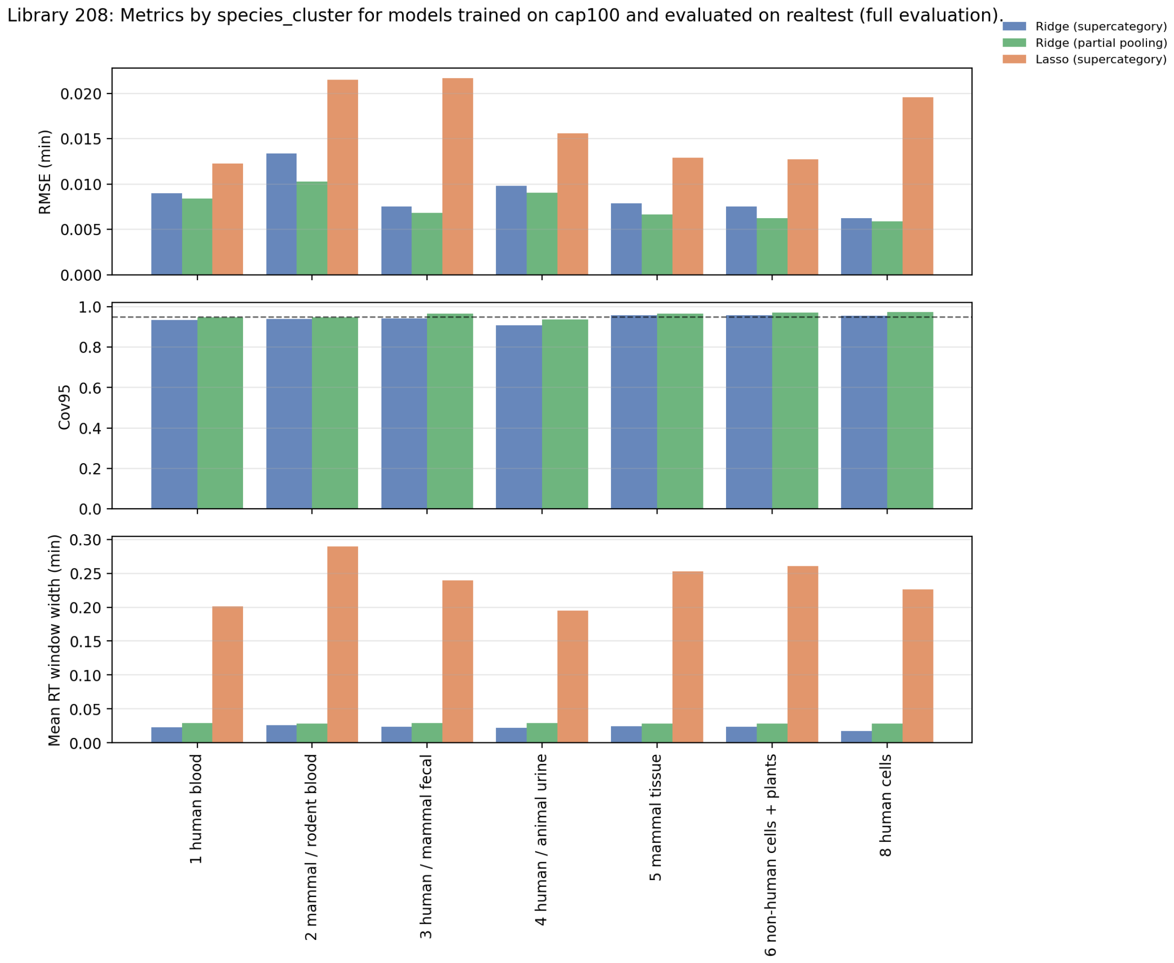

Figure 7: Metrics by `species_cluster` (supercategory) for lib208 report baselines. The Cov95 panel excludes lasso because its fixed filtering window is not a nominal probabilistic interval; mean RT window width includes the lasso filtering window for context.

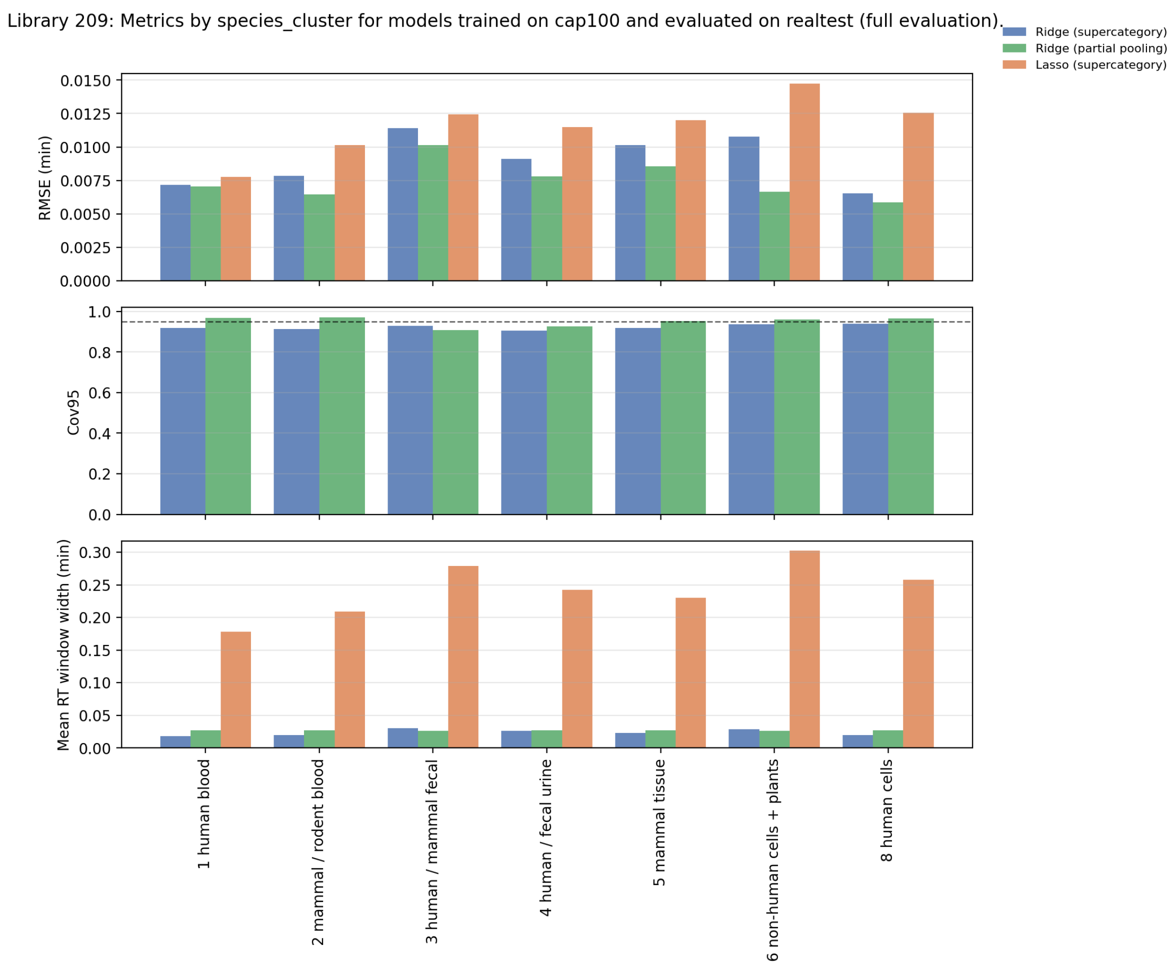

Figure 8: Metrics by `species_cluster` (supercategory) for lib209 report baselines. The Cov95 panel excludes lasso because its fixed filtering window is not a nominal probabilistic interval; mean RT window width includes the lasso filtering window for context.

5 Results: end-to-end Sally evaluation

5.1 Setup

Offline split evaluation is necessary but not sufficient for deployment: in production, the RT model is used inside Sally's peak assignment pipeline as a hard filter and interacts with other stages (candidate generation, peak selection, and post-processing). We therefore evaluate end-to-end peak assignment using `sally-dev evaluate`. We compare the current production baseline RT model bundle (supercategory lasso; ESLASSO) against the proposed CompAssign partial pooling ridge bundle exported as stage-1 coefficient summaries (COMPASSIGN_PP_RIDGE). We report precision/recall/Threat Score (TS) from Sally's `mdsutils` summary.

We also compare two peak-selection modes: Sally's standard within-task selection logic (peak_assignment_method=baseline) and the compound-level RT coherence rejection described in Section 2.9 (peak_assignment_method=mixture_model).

This yields a 2×2 ablation over (i) RT model (ESLASSO vs COMPASSIGN_PP_RIDGE) and (ii) peak assignment method (baseline vs mixture_model). In these runs, COMPASSIGN_PP_RIDGE uses a widened RT window multiplier ($m = 4$).

We evaluated eight held-out cell-type SSIDs from supercategories 6 and 8 (two SSIDs per supercategory per library), selected from the test fold of `data/split_outputs/train_test_split_all.csv`. For lib208: SSID 12307 and SSID 12725 (supercat 6), and SSID 12609 and SSID 13208 (supercat 8). For lib209: SSID 23059 and SSID 23146 (supercat 6), and SSID 20159 and SSID 20814 (supercat 8). For reproducibility, `src/compassign/rt/sally_test.sh` re-runs the exact SSIDs and extracts metrics from the `mdsutils` summary pickles. Sally integration details are listed in Appendix A.

5.2 Results

Tables 2 and 3 report the end-to-end peak-assignment metrics from Sally's `mdsutils` summary pickles. Higher is better for all three metrics. For each SSID, the baseline and COMPASSIGN_PP_RIDGE were run with the same evaluation mode (`mdsutils`).

Table 2: Preliminary end-to-end Sally evaluation on selected cell-type SSIDs (lib208; supercategories 6 and 8). Metrics are precision, recall, and Threat Score (TS) from `mdsutils`. Bold indicates the best value within each SSID.

Supercat	SSID	Setting	Precision	Recall	TS
6	12307	ESLASSO + baseline	0.8436	0.7848	0.6851
6	12307	ESLASSO + mixture_model	0.8300	0.4737	0.4318
6	12307	COMPASSIGN_PP_RIDGE + baseline	0.9390	0.8113	0.7706
6	12307	COMPASSIGN_PP_RIDGE + mixture_model	**0.9480**	**0.8122**	**0.7776**
6	12725	ESLASSO + baseline	0.6808	0.8610	0.6134
6	12725	ESLASSO + mixture_model	0.6808	0.8610	0.6134
6	12725	COMPASSIGN_PP_RIDGE + baseline	0.8285	**0.8810**	0.7451
6	12725	COMPASSIGN_PP_RIDGE + mixture_model	**0.9303**	**0.8810**	**0.8264**
8	12609	ESLASSO + baseline	0.8553	0.6513	0.5867
8	12609	ESLASSO + mixture_model	0.8593	0.5410	0.4970
8	12609	COMPASSIGN_PP_RIDGE + baseline	0.8417	0.8070	0.7006
8	12609	COMPASSIGN_PP_RIDGE + mixture_model	**0.9320**	**0.8073**	**0.7624**
8	13208	ESLASSO + baseline	0.8931	0.7930	0.7243
8	13208	ESLASSO + mixture_model	0.8931	0.7930	0.7243
8	13208	COMPASSIGN_PP_RIDGE + baseline	0.8546	0.7930	0.6987
8	13208	COMPASSIGN_PP_RIDGE + mixture_model	**0.9555**	**0.7973**	**0.7688**

Table 3: Preliminary end-to-end Sally evaluation on selected cell-type SSIDs (lib209; supercategories 6 and 8). Metrics are precision, recall, and Threat Score (TS) from `mdsutils`. Bold indicates the best value within each SSID.

Supercat	SSID	Setting	Precision	Recall	TS
6	23059	ESLASSO + baseline	0.8357	0.5940	0.5319
6	23059	ESLASSO + mixture_model	0.8357	0.5940	0.5319
6	23059	COMPASSIGN_PP_RIDGE + baseline	0.8164	**0.6868**	0.5949
6	23059	COMPASSIGN_PP_RIDGE + mixture_model	**0.8913**	0.6852	**0.6324**
6	23146	ESLASSO + baseline	0.6732	0.8257	0.5894
6	23146	ESLASSO + mixture_model	0.6732	0.8257	0.5894
6	23146	COMPASSIGN_PP_RIDGE + baseline	0.6800	0.8686	0.6166
6	23146	COMPASSIGN_PP_RIDGE + mixture_model	**0.9857**	**0.8692**	**0.8584**
8	20159	ESLASSO + baseline	0.8865	0.8697	0.7825
8	20159	ESLASSO + mixture_model	0.8865	0.8697	0.7825
8	20159	COMPASSIGN_PP_RIDGE + baseline	0.8906	**0.8848**	0.7980
8	20159	COMPASSIGN_PP_RIDGE + mixture_model	**0.9871**	**0.8848**	**0.8747**
8	20814	ESLASSO + baseline	0.9123	0.8695	0.8024
8	20814	ESLASSO + mixture_model	0.9123	0.8695	0.8024
8	20814	COMPASSIGN_PP_RIDGE + baseline	0.9194	**0.8783**	0.8155
8	20814	COMPASSIGN_PP_RIDGE + mixture_model	**0.9687**	0.8753	**0.8513**

Across these eight SSIDs, the strongest setting is consistently the combination of COMPASSIGN_PP_RIDGE with mixture_model, which improves TS relative to the production ESLASSO baseline for all eight checks. Within COMPASSIGN_PP_RIDGE, mixture_model improves TS versus baseline in every case. In contrast, applying mixture_model to ESLASSO does not help and can substantially reduce recall/TS (e.g. SSIDs 12307 and 12609 in Table 2).

5.3 Notes

The `mdsutils` TS in Tables 2 and 3 includes false negatives for curated internal standards and X-compounds, which are omitted from production outputs by design. We therefore treat these TS values as conservative when judging production impact, and interpret TS deltas in the context of the "do not call standards / X-compounds" constraint.

For these cap100-trained lib208/lib209 models, the coefficient-summary artifacts include compounds with missing chemistry metadata (missing `chemical_id` or missing ChemBERTa embeddings). For lib208, the trainer uses mean-embedding fallback for 310 compounds with missing `chemical_id` and 138 compounds with missing embeddings; for lib209, this is 990 and 540 compounds, respectively. This prevents coverage holes in the Sally integration (avoids "No RT model") at the cost of disabling the chemistry-informed prior mean for those compounds (chemistry term uses $z_c = 0$).

6 Discussion and conclusion

6.1 Summary of findings

Offline split evaluation shows that COMPASSIGN_PP_RIDGE improves RT prediction and produces RT windows with coverage close to the nominal 95% target. This matters because Sally uses RT windows as a hard filter: if the window is too narrow, true peaks are removed (false negatives); if it is too wide, decoys survive and can become false positives. Relative to the fast supercategory ridge baseline, the hierarchical model improves RMSE while reducing under-coverage.

End-to-end evaluation inside Sally confirms these improvements translate to peak assignment performance when combined with the across-task coherence check. On eight held-out SSIDs (Tables 2 and 3), COMPASSIGN_PP_RIDGE with mixture_model improves Threat Score (TS) versus the current ESLASSO baseline on all checks (lib208: approximately +0.04 to +0.21; lib209: approximately +0.05 to +0.27). Within COMPASSIGN_PP_RIDGE, the coherence check improves TS versus the baseline within-task peak picker for every SSID evaluated. This supports the central hypothesis: RT evidence is often strongest when aggregated across tasks, and a compound-level coherence score can remove false positives that survive per-task filtering.

6.2 Why the coherence score works

The mixture-based coherence check is most reliable when the underlying RT model produces a calibrated predictive sd. For COMPASSIGN_PP_RIDGE, the per-task window is derived from an explicit predictive variance, so the per-task RT log-likelihood behaves like a standardized residual score. This makes the compound score x_c a meaningful continuous signal and allows the mixture model to adapt its separation boundary to each SSID.

By contrast, for ESLASSO the window is heuristic and constant across tasks, so the same score behaves like a pseudo-likelihood and can be dominated by window width rather than true coherence. In practice this can lead to over-rejection and recall loss on some SSIDs. This difference highlights a key practical point: if RT uncertainty is used downstream as a hard filter and as a likelihood scale, then having a probabilistic RT model is not just “nicer”—it can change the behavior of compound-level aggregation in a qualitatively important way.

6.3 Limitations and future work

The main practical downside of the hierarchical model is training cost: variational inference over the hierarchy is far slower than independent ridge fits. This makes the hierarchical model best suited to offline retrains (periodic model refreshes), while the fast sklearn ridge baseline remains useful for quick iteration and regression checks. The lasso baseline remains valuable as a historical point of comparison, but its lack of hierarchy and its heuristic uncertainty windows make it a weaker fit for modern peak assignment requirements.

The end-to-end evaluation in this report is intentionally scoped: it covers a small set of SSIDs and focuses on single-library checks. Before deployment, the most important next step is to expand evaluation across more SSIDs (including multi-library sample sets) and to confirm that RT window calibration remains stable under different run conditions and post-processing settings. Separately, the mixture_model coherence probability could be exposed as a soft confidence score (instead of a hard rejection) to support explicit precision/recall trade-offs in downstream reporting.

6.4 Unseen species (exploratory holdouts)

The main results focus on seen-compound performance. To probe cold-start for new species, we ran exploratory group holdouts on a production subset with two modes:

- `species`: hold out entire species within each `species_cluster`,
- `species_comp`: hold out (`species`, `comp_id`) pairs. For each held-out pair, we keep the corresponding `species_cluster` and `comp_id` observed in training.

We evaluate the hierarchical ridge model with an explicit unseen-species backoff (aggregate fitted (`species`, `comp_id`) coefficients across training species to form a supercategory–compound coefficient), compare to the supercategory ridge baseline, and include the legacy lasso bundle for context.

Table 4: Unseen holdout metrics (lib208 production subset; seed=42; holdout_frac=0.2; clusters=4,5; top_comp_ids=100).

Holdout	Model	RMSE	MAE	Cov95	Rows scored (%)
`species`	Ridge (partial pooling)	0.014968	0.009012	1.000	**100.0**
`species`	Ridge (supercategory)	0.011090	0.005202	0.877	**100.0**
`species`	Lasso (supercategory, external)	**0.007913**	**0.004174**	–	94.8
`species_comp`	Ridge (partial pooling)	0.013474	0.009569	1.000	**100.0**
`species_comp`	Ridge (supercategory)	0.013272	0.006532	0.812	**100.0**
`species_comp`	Lasso (supercategory, external)	**0.007671**	**0.004331**	–	94.9

Table 5: Unseen holdout metrics (lib209 production subset; seed=42; holdout_frac=0.2; clusters=4,5; top_comp_ids=100).

Holdout	Model	RMSE	MAE	Cov95	Rows scored (%)
`species`	Ridge (partial pooling)	0.008381	0.005281	1.000	**100.0**
`species`	Ridge (supercategory)	0.008268	0.004870	0.902	**100.0**
`species`	Lasso (supercategory, external)	**0.006779**	**0.004267**	–	92.4
`species_comp`	Ridge (partial pooling)	0.011684	0.006456	1.000	**100.0**
`species_comp`	Ridge (supercategory)	0.013582	0.006320	0.883	**100.0**
`species_comp`	Lasso (supercategory, external)	**0.009482**	**0.005677**	–	95.0

These holdouts suggest that an explicit backoff strategy can make the hierarchical model usable for unseen species as long as the supercategory remains known. The high coverage values indicate the backoff is conservative; improving precision under unseen species will likely require either a stronger species hierarchy (so truly unseen species can be handled without ad-hoc aggregation) or small amounts of new-species calibration data.

6.5 Unseen compounds (zero-shot chemistry)

To isolate true compound cold-start, we ran a held-out chemistry experiment where `chem_id` values are removed entirely from training. We compare the hierarchical ridge model (which can still score via the chemistry prior mean) to a non-linear single-model embedding baseline (MLP) that uses ChemBERTa PCA-20 features for point prediction.

Table 6: Zero-shot chemistry results on the lib208 production subset (hold out 20 `chem_id`; seed=42).

Model	RMSE	MAE	Cov95
Ridge (partial pooling)	1.049358	0.699639	**1.000**
MLP (ChemBERTa PCA-20 + cluster interactions)	**0.468125**	**0.345321**	–

Table 7: Zero-shot chemistry results on the lib209 production subset (hold out 20 `chem_id`; seed=42).

Model	RMSE	MAE	Cov95
Ridge (partial pooling)	1.523705	1.062139	**0.991**
MLP (ChemBERTa PCA-20 + cluster interactions)	**1.061293**	**0.799969**	–

Point accuracy in the true zero-shot setting is currently much better for the non-linear embedding baseline, which suggests the linear chemistry head inside the hierarchical model is not yet expressive enough. However, the hierarchical model still provides a calibrated uncertainty estimate, which is valuable for windowing. Improving the chemistry head while keeping the residual per-compound term δ_k (so seen-compound performance is maintained) is the key next step.

Two practical extensions are: (i) add a compound-class anchor term so unseen compounds can back off to a learned class mean when embeddings are weak, and (ii) increase shrinkage on the chemistry-head coefficients $\theta_{\alpha,1:D}$ (or use global shrinkage priors) to avoid high-variance extrapolation on held-out chemistries.

In conclusion, the combination of a probabilistic RT regression model and an SSID-specific coherence score provides a more principled RT pipeline for autocuration: calibrated per-task uncertainty supports reliable RT filtering, and the mixture-based compound-level probability provides a continuous, data-adaptive criterion for rejecting likely false positives without a fixed global threshold. The remaining work is to broaden end-to-end evaluation and to improve cold-start behavior for truly unseen species and chemistries while preserving seen-compound performance.

A Sally integration details

To support COMPASSIGN_PP_RIDGE end-to-end inside `sally-dev`, we made the following focused changes:

- Added modelType=COMPASSIGN_PP_RIDGE which loads CompAssign stage-1 coefficient summaries and backoff summaries (both shipped as `.npz` artifacts), and computes per-task RT windows from predictive variance.
- Generalized Sally's regression predictor so RT windows can be either per-compound (1D) or per-(task,compound) (2D), since COMPASSIGN_PP_RIDGE produces row-specific uncertainty.
- Added CompAssign RT inputs (species, species_cluster, and covariates CSV) and `--force-curate` to run the full curation pipeline for evaluation even when a sample set would normally be skipped.
- Updated `sally-dev evaluate` to fail fast when the underlying pipeline run fails (instead of continuing with stale cached pickles), and updated the pipeline wrapper and COMPASSIGN_PP_RIDGE loader so production-style RT covariates can be mounted into the Docker container and read efficiently by streaming only the required task IDs (important for multi-GB lib209 covariates CSVs).

B Collapsed-slope derivation

This appendix shows, step by step, how we integrate out the ridge-regularized slope coefficients for one group g. This yields a marginal ("collapsed") likelihood that depends only on compact per-group summary statistics, which is why the PyMC model can scale to production data.

B.1 Summary (what is being collapsed and why)

Within each group g, the RT model is a standard linear regression with a Gaussian ridge prior on slopes:

$$y_g = b_g \mathbf{1}_{n_g} + X_g w_g + \epsilon_g, \qquad \epsilon_g \sim \mathcal{N}(0, \sigma^2 I_{n_g}), \qquad w_g \sim \mathcal{N}\big(m_g,\ (\sigma^2/\lambda_{\text{slopes}})\, I_p\big)\,.$$

Because both the likelihood and the prior are Gaussian, we can integrate out w_g exactly. The marginal distribution of y_g remains Gaussian:

$$y_g \mid b_g, m_g, \sigma^2 \sim \mathcal{N}\Big(b_g \mathbf{1}_{n_g} + X_g m_g,\ \sigma^2 \left(I_{n_g} + \tfrac{1}{\lambda_{\text{slopes}}} X_g X_g^\top\right)\Big)\,.$$

This is useful computationally: the group log-likelihood can be written in terms of a $p \times p$ matrix $I_p + \frac{1}{\lambda_{\text{slopes}}} X_g^\top X_g$ (instead of an $n_g \times n_g$ covariance matrix). Since p is small (number of run covariates), this lets the PyMC model evaluate many groups efficiently while still retaining a proper Bayesian ridge effect on slopes.

B.2 Setup and notation (one group)

We work with a single group g (for this project, a group corresponds to a single `(species, comp_id)` pair). Let I_g be the index set of rows in the dataset that belong to group g, and let $n_g = |I_g|$ be the number of rows.

We will use the following symbols (with dimensions shown explicitly):

- p: number of run covariates.
- $y_g \in \mathbb{R}^{n_g}$: response vector for group g (RT values).
- $X_g \in \mathbb{R}^{n_g \times p}$: design matrix for group g (run covariates). Row i is the covariate row vector $x_i^\top$, and entry $(X_g)_{i,j} = x_{i,j}$.
- $b_g \in \mathbb{R}$: group intercept.
- $w_g \in \mathbb{R}^p$: group slope vector.
- $m_g \in \mathbb{R}^p$: prior mean for the group slopes (in the main model this comes from the slope-mean hierarchy).
- $\sigma > 0$ and σ^2: residual noise scale and variance.
- $\lambda_{\text{slopes}} > 0$: ridge penalty (a precision) for the slopes.
- $\mathbf{1}_{n_g} \in \mathbb{R}^{n_g}$: all-ones vector. I_{n_g} and I_p are identity matrices.

The likelihood is:

$$y_g \mid b_g, w_g, \sigma^2 \sim \mathcal{N}\big(b_g \mathbf{1}_{n_g} + X_g w_g,\ \sigma^2 I_{n_g}\big)\,,$$

which is equivalent to the generative equation

$$y_g = b_g \mathbf{1}_{n_g} + X_g w_g + \epsilon_g, \qquad \epsilon_g \sim \mathcal{N}(0, \sigma^2 I_{n_g}).$$

The ridge prior on slopes (conditional on m_g) is:

$$w_g \mid m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}\big(m_g,\ (\sigma^2/\lambda_{\text{slopes}}) I_p\big)\,.$$

It is often helpful to write this prior as

$$w_g = m_g + u_g, \qquad u_g \sim \mathcal{N}\big(0,\ (\sigma^2/\lambda_{\text{slopes}}) I_p\big)\,,$$

so that u_g is the zero-mean deviation of the group slopes away from their prior mean m_g.

B.3 Goal: the marginal likelihood

We want the marginal likelihood obtained by integrating out w_g:

$$p(y_g \mid b_g, m_g, \sigma^2, \lambda_{\text{slopes}}) = \int p(y_g \mid b_g, w_g, \sigma^2)\, p(w_g \mid m_g, \sigma^2, \lambda_{\text{slopes}})\, dw_g.$$

B.4 Step 1: rewrite the model in terms of a centered residual

Define the centered response

$$\tilde{y}_g \equiv y_g - b_g \mathbf{1}_{n_g},$$

and then define the centered residual

$$r_g \equiv \tilde{y}_g - X_g m_g = y_g - b_g \mathbf{1}_{n_g} - X_g m_g.$$

This r_g is what remains after subtracting the intercept term and the prior-mean slope contribution.

B.5 Step 2: express the residual as a sum of two independent Gaussian terms

Using $w_g = m_g + u_g$ in the likelihood equation:

$$\begin{aligned} y_g &= b_g \mathbf{1}_{n_g} + X_g(m_g + u_g) + \epsilon_g \\ &= b_g \mathbf{1}_{n_g} + X_g m_g + X_g u_g + \epsilon_g. \end{aligned}$$

Subtract $b_g \mathbf{1}_{n_g} + X_g m_g$ from both sides:

$$r_g = X_g u_g + \epsilon_g.$$

By construction, u_g and ϵ_g are independent and both are mean-zero Gaussians.

B.6 Step 3: compute the distribution of the residual (mean and covariance)

First compute the mean:

$$\mathbb{E}[r_g] = \mathbb{E}[X_g u_g + \epsilon_g] = X_g \mathbb{E}[u_g] + \mathbb{E}[\epsilon_g] = 0.$$

Next compute the covariance. Because u_g and ϵ_g are independent, the cross-covariance terms vanish, so:

$$\begin{aligned} \text{Cov}(r_g) &= \text{Cov}(X_g u_g + \epsilon_g) \\ &= \text{Cov}(X_g u_g) + \text{Cov}(\epsilon_g). \end{aligned}$$

Compute each term separately:

$$\text{Cov}(\epsilon_g) = \sigma^2 I_{n_g}.$$

For the slope term, use $\text{Cov}(Az) = A\,\text{Cov}(z)\,A^\top$:

$$\begin{aligned} \text{Cov}(X_g u_g) &= X_g\,\text{Cov}(u_g)\,X_g^\top \\ &= X_g\left((\sigma^2/\lambda_{\text{slopes}})I_p\right)X_g^\top \\ &= (\sigma^2/\lambda_{\text{slopes}})\,X_g X_g^\top. \end{aligned}$$

Putting the two pieces together:

$$\begin{aligned} \text{Cov}(r_g) &= \sigma^2 I_{n_g} + (\sigma^2/\lambda_{\text{slopes}})\,X_g X_g^\top \\ &= \sigma^2\left(I_{n_g} + \lambda_{\text{slopes}}^{-1} X_g X_g^\top\right). \end{aligned}$$

Define the $n_g \times n_g$ matrix

$$C_g \equiv I_{n_g} + \lambda_{\text{slopes}}^{-1} X_g X_g^\top.$$

Then we have the simple distribution statement:

$$r_g \mid b_g, m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}(0, \sigma^2 C_g).$$

Because y_g and r_g differ only by a deterministic shift, this also gives the marginal distribution of y_g given $b_g, m_g, \sigma^2, \lambda_{\text{slopes}}$.

B.7 Step 4: write the Gaussian log density in terms of its covariance

If $r \sim \mathcal{N}(0, \sigma^2 C)$ with C symmetric positive-definite, then the density is:

$$p(r) = (2\pi)^{-n/2}\,|\sigma^2 C|^{-1/2}\,\exp\!\left(-\frac{1}{2} r^\top (\sigma^2 C)^{-1} r\right),$$

and therefore the log density is:

$$\log p(r) = -\frac{1}{2}\left[n \log(2\pi) + \log|\sigma^2 C| + r^\top (\sigma^2 C)^{-1} r\right].$$

Apply this with $n = n_g$, $r = r_g$, and $C = C_g$:

$$\begin{aligned}\log p(y_g \mid b_g, m_g, \sigma^2, \lambda_{\text{slopes}}) &= \log p(r_g \mid b_g, m_g, \sigma^2, \lambda_{\text{slopes}}) \\ &= -\frac{1}{2}\left[n_g \log(2\pi) + \log|\sigma^2 C_g| + r_g^\top (\sigma^2 C_g)^{-1} r_g\right].\end{aligned}$$

Now simplify the two terms involving σ^2:

$$\log|\sigma^2 C_g| = \log\big((\sigma^2)^{n_g}|C_g|\big) = n_g \log(\sigma^2) + \log|C_g|,$$

and

$$r_g^\top (\sigma^2 C_g)^{-1} r_g = r_g^\top \left((1/\sigma^2) C_g^{-1}\right) r_g = \frac{1}{\sigma^2} r_g^\top C_g^{-1} r_g.$$

So the log-likelihood becomes:

$$\log p(y_g \mid b_g, m_g, \sigma^2, \lambda_{\text{slopes}}) = -\frac{1}{2}\left[n_g \log(2\pi\sigma^2) + \log|C_g| + \frac{1}{\sigma^2} r_g^\top C_g^{-1} r_g\right].$$

At this point, the remaining computational problem is: how do we compute $\log|C_g|$ and $r_g^\top C_g^{-1} r_g$ efficiently, without constructing the potentially huge $n_g \times n_g$ matrix C_g?

B.8 Step 5: reduce computations to p-by-p using standard identities

Define the $p \times p$ matrix and p-vector:

$$A_g \equiv X_g^\top X_g + \lambda_{\text{slopes}} I_p,$$

$$s_g \equiv X_g^\top r_g.$$

We will show (in full) that:

$$\log|C_g| = \log|A_g| - p\log(\lambda_{\text{slopes}}) \quad \text{and} \quad r_g^\top C_g^{-1} r_g = r_g^\top r_g - s_g^\top A_g^{-1} s_g.$$

B.8.1 Step 5a: determinant reduction (matrix determinant lemma)

We start from the definition $C_g = I_{n_g} + \lambda_{\text{slopes}}^{-1} X_g X_g^\top$. Introduce the scaled matrix $\tilde{X}_g \equiv \lambda_{\text{slopes}}^{-1/2} X_g$, so that $\tilde{X}_g \tilde{X}_g^\top = \lambda_{\text{slopes}}^{-1} X_g X_g^\top$. Then:

$$|C_g| = \left| I_{n_g} + \tilde{X}_g \tilde{X}_g^\top \right|.$$

The matrix determinant lemma states that for conformable matrices U and V,

$$|I + UV| = |I + VU|.$$

Apply it with $U = \tilde{X}_g$ (an $n_g \times p$ matrix) and $V = \tilde{X}_g^\top$ (a $p \times n_g$ matrix):

$$\begin{aligned} \left| I_{n_g} + \tilde{X}_g \tilde{X}_g^\top \right| &= \left| I_p + \tilde{X}_g^\top \tilde{X}_g \right| \\ &= \left| I_p + \lambda_{\text{slopes}}^{-1} X_g^\top X_g \right|. \end{aligned}$$

Now factor out $\lambda_{\text{slopes}}^{-1}$:

$$\begin{aligned} I_p + \lambda_{\text{slopes}}^{-1} X_g^\top X_g &= \lambda_{\text{slopes}}^{-1} \left(X_g^\top X_g + \lambda_{\text{slopes}} I_p \right) \\ &= \lambda_{\text{slopes}}^{-1} A_g. \end{aligned}$$

Taking determinants:

$$|C_g| = \left| \lambda_{\text{slopes}}^{-1} A_g \right| = \lambda_{\text{slopes}}^{-p} |A_g|.$$

Finally take logs:

$$\log |C_g| = \log |A_g| - p \log(\lambda_{\text{slopes}}).$$

B.8.2 Step 5b: inverse/quadratic-form reduction (Woodbury identity)

We again start from $C_g = I_{n_g} + \lambda_{\text{slopes}}^{-1} X_g X_g^\top$. The Woodbury identity states:

$$(I + UCV)^{-1} = I - U \left(C^{-1} + VU \right)^{-1} V,$$

for conformable matrices U, C, V (with inverses defined). Apply it with:

$$U = X_g, \qquad C = \lambda_{\text{slopes}}^{-1} I_p, \qquad V = X_g^\top.$$

Then $C^{-1} = \lambda_{\text{slopes}} I_p$ and $C^{-1} + VU = \lambda_{\text{slopes}} I_p + X_g^\top X_g = A_g$. So:

$$C_g^{-1} = I_{n_g} - X_g A_g^{-1} X_g^\top.$$

Now compute the quadratic form explicitly:

$$\begin{aligned} r_g^\top C_g^{-1} r_g &= r_g^\top \left(I_{n_g} - X_g A_g^{-1} X_g^\top \right) r_g \\ &= r_g^\top r_g - r_g^\top X_g A_g^{-1} X_g^\top r_g. \end{aligned}$$

Recognize $s_g = X_g^\top r_g$ and note that $r_g^\top X_g = (X_g^\top r_g)^\top = s_g^\top$. Therefore:

$$r_g^\top X_g A_g^{-1} X_g^\top r_g = s_g^\top A_g^{-1} s_g,$$

and hence:

$$r_g^\top C_g^{-1} r_g = r_g^\top r_g - s_g^\top A_g^{-1} s_g.$$

B.9 Final collapsed log-likelihood (putting it all together)

Substitute the two identities into the log-likelihood expression from Step 4:

$$\theta_g \equiv (b_g, m_g, \sigma^2, \lambda_{\text{slopes}}).$$

$$\log p(y_g \mid \theta_g) = -\frac{1}{2}\left[n_g \log(2\pi\sigma^2) + \big(\log|A_g| - p\log(\lambda_{\text{slopes}})\big) + \frac{1}{\sigma^2}\left(r_g^\top r_g - s_g^\top A_g^{-1} s_g\right)\right].$$

This final expression depends on the full row-level data only through small per-group quantities:

- $X_g^\top X_g$ (a $p \times p$ matrix),
- $s_g = X_g^\top r_g$ (a p-vector),
- $r_g^\top r_g$ (a scalar),
- and n_g (a scalar).

These are the sufficient statistics that we precompute and feed into ADVI, instead of representing every slope coefficient for every row as an explicit latent variable.

B.10 Closed-form posterior for the slopes

We now derive (without skipping steps) the conditional posterior distribution of w_g for one group, given the hyperparameters and the intercept b_g.

1. Start from the likelihood and prior. Write the centered response as $\tilde{y}_g = y_g - b_g \mathbf{1}_{n_g}$. The likelihood is:
$$\tilde{y}_g \mid w_g, \sigma^2 \sim \mathcal{N}(X_g w_g, \sigma^2 I_{n_g}).$$
The ridge prior is:
$$w_g \mid m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}\big(m_g, (\sigma^2/\lambda_{\text{slopes}}) I_p\big).$$
2. Write the unnormalized log posterior. Up to an additive constant (that does not depend on w_g),
$$\begin{aligned}\log p(w_g \mid \tilde{y}_g, m_g, \sigma^2, \lambda_{\text{slopes}}) &\propto \log p(\tilde{y}_g \mid w_g, \sigma^2) + \log p(w_g \mid m_g, \sigma^2, \lambda_{\text{slopes}}) \\ &\propto -\frac{1}{2\sigma^2}\|\tilde{y}_g - X_g w_g\|_2^2 - \frac{\lambda_{\text{slopes}}}{2\sigma^2}\|w_g - m_g\|_2^2.\end{aligned}$$
3. Expand both squared norms. First expand the likelihood term:
$$\begin{aligned}\|\tilde{y}_g - X_g w_g\|_2^2 &= (\tilde{y}_g - X_g w_g)^\top(\tilde{y}_g - X_g w_g) \\ &= \tilde{y}_g^\top \tilde{y}_g - 2w_g^\top X_g^\top \tilde{y}_g + w_g^\top X_g^\top X_g w_g.\end{aligned}$$
Next expand the prior term:
$$\begin{aligned}\|w_g - m_g\|_2^2 &= (w_g - m_g)^\top(w_g - m_g) \\ &= w_g^\top w_g - 2w_g^\top m_g + m_g^\top m_g.\end{aligned}$$

4. Collect the terms that depend on w_g. Ignoring constants (terms not involving w_g), the exponent becomes:
$$-\frac{1}{2\sigma^2}\left[w_g^\top (X_g^\top X_g + \lambda_{\text{slopes}} I_p) w_g - 2 w_g^\top (X_g^\top \tilde{y}_g + \lambda_{\text{slopes}} m_g)\right].$$
Define $A_g \equiv X_g^\top X_g + \lambda_{\text{slopes}} I_p$ and $h_g \equiv X_g^\top \tilde{y}_g + \lambda_{\text{slopes}} m_g$. Then the exponent is:
$$-\frac{1}{2\sigma^2}\left[w_g^\top A_g w_g - 2 w_g^\top h_g\right].$$

5. Complete the square. For any symmetric positive-definite matrix A and vector h, we have:
$$w^\top A w - 2w^\top h = (w - A^{-1}h)^\top A (w - A^{-1}h) - h^\top A^{-1} h.$$
Applying this with $A = A_g$ and $h = h_g$, the posterior is a Gaussian with mean $A_g^{-1} h_g$ and covariance $\sigma^2 A_g^{-1}$:
$$w_g \mid y_g, b_g, m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}\left(A_g^{-1} h_g,\ \sigma^2 A_g^{-1}\right).$$

6. Rewrite the posterior mean using r_g and s_g. Recall that $r_g = \tilde{y}_g - X_g m_g$, so $\tilde{y}_g = X_g m_g + r_g$. Then:
$$\begin{aligned} h_g &= X_g^\top \tilde{y}_g + \lambda_{\text{slopes}} m_g \\ &= X_g^\top (X_g m_g + r_g) + \lambda_{\text{slopes}} m_g \\ &= (X_g^\top X_g + \lambda_{\text{slopes}} I_p) m_g + X_g^\top r_g \\ &= A_g m_g + s_g. \end{aligned}$$
Therefore:
$$A_g^{-1} h_g = A_g^{-1}(A_g m_g + s_g) = m_g + A_g^{-1} s_g,$$
and we recover the compact form:
$$w_g \mid y_g, b_g, m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}\left(m_g + A_g^{-1} s_g,\ \sigma^2 A_g^{-1}\right).$$

These formulas are what we use after ADVI to recover per-group slope posterior means and covariances. Combined with the intercept treatment in the main model, they produce the exported stage-1 coefficient summaries used for scoring.