# Report: RT Regression Model (Hierarchical Ridge with Chemistry)

Bioinformatics Team

January 5, 2026

## 1 Introduction

Retention time (RT) is a strong auxiliary signal for peak assignment. For a given compound, we compare an observed peak RT to a model prediction and retain candidates that fall within an uncertainty window. This makes two properties critical: accurate point predictions, and *calibrated* uncertainty so that a nominal "95% window" contains the true RT about 95% of the time.

In current production, the baseline is a bank of **supercategory-specific lasso regressions**. For each `species_cluster` (supercategory / matrix), the baseline predicts RT from run covariates (`IS*`/`RS*`/`ES_`) using an $\ell_1$ penalty (lasso), and associates each prediction with a hand-tuned window rather than a probabilistic predictive distribution. The implementation and model bundle live in the Hippopotamus pipeline (`external_repos/Hippopotamus`).

This baseline is useful, but it has limitations that matter for peak assignment:

- **No partial pooling in the long tail.** Rare `species` and sparse compound histories are fit largely independently, which can produce noisy predictions and inconsistent calibration when training support is low.
- **No chemistry-informed sharing across compounds.** Compound effects are treated as unrelated parameters, so the model cannot borrow strength across chemically similar compounds when compound history is sparse.
- **Heuristic uncertainty.** Window widths are not derived from a probabilistic model, so "coverage" is not a nominal probabilistic statement and is hard to tune systematically.

An $\ell_1$ (lasso) prior is also a poor default for our run covariates. `IS*` and `ES_` predictors are often strongly correlated because they capture overlapping aspects of run drift and instrument state. With correlated predictors, lasso tends to select an arbitrary subset and set the rest to zero, making coefficients unstable across retrains and creating brittle behavior on sparse groups. A ridge (Gaussian) prior is a better default here: it shrinks correlated covariates smoothly instead of forcing hard selection, leading to more stable predictions and uncertainty estimates.

We address these issues with a hierarchical Bayesian ridge model with chemistry-informed sharing. The model improves long-tail stability via partial pooling across `species` nested in `species_cluster`,

regularizes compound baselines via a chemistry-informed prior that shares information across compounds, and provides calibrated uncertainty via an explicit predictive distribution.

For deployment, we export coefficient summaries (posterior means and variances), so scoring stays fast and simple: compute a linear prediction from the run covariates and attach a Normal predictive interval, without running Bayesian inference at runtime.

This report trains models for lib208 and lib209 and evaluates on the corresponding realtest splits, comparing against a per-supercategory ridge baseline and the production lasso baseline. On these evaluations, the partial pooling model improves RMSE relative to the ridge baseline while keeping coverage close to the nominal 95% target.

# 2 Methods: hierarchical ridge with chemistry

This section describes the proposed model in full, starting from a high-level decomposition and then giving a complete probabilistic specification (likelihood, priors, and inference).

## 2.1 Data and notation

Each RT observation row $i$ contains:

- response $y_i$ (RT, minutes),
- run covariates $x_{i,j}$ for $j = 1, \ldots, p$ 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 $d = 1, \ldots, D$ (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:

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

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.

**Interpretation.** The intercept $b_{c,k}$ represents the baseline RT of compound $k$ in species $c$; we expect related species in the same supercategory to have similar baselines, and we expect chemically similar compounds to have similar baselines, so we pool both species and compound effects. The slope coefficients $w_{c,k,j}$ capture how run conditions shift RT; because run covariates are correlated, we use ridge shrinkage for stability and we pool slope means at the species and supercategory levels.

## 2.3 Likelihood

Conditioned on group-level coefficients:

$$p(\{y_i\} \mid \{b_{c,k}\}, \{w_{c,k,j}\}, \sigma^2) = \prod_i \mathcal{N}\left(y_i \mid b_{c(i),k(i)} + \sum_{j=1}^{p} x_{i,j}\, w_{c(i),k(i),j}, \sigma^2\right).$$

## 2.4 Priors (partial pooling + chemistry)

**What pooling means (and what "partial pooling" is).** Pooling is how a model shares information across related groups when some groups have little data. In a regression with many groups, there are three common extremes:

- **No pooling:** fit each group completely independently (high variance in sparse groups).
- **Full pooling:** force groups to share the same parameter (low variance, but can underfit real between-group differences).

- **Partial pooling:** learn group-specific parameters while shrinking them toward a shared mean, with the amount of shrinkage learned from the data.

Hierarchical priors implement partial pooling: each group parameter is drawn from a distribution centered at a population-level mean, and the population scales (e.g., $\tau_{\mu,\cdot}$ and $\tau_{w,\cdot}$ below) determine how strongly groups are pulled together.

**Where partial pooling happens in this model.** Partial pooling occurs in two places. First, the intercept hierarchy pools baseline RTs across `species` within a `species_cluster` (supercategory / matrix) (via $\mu_c \leftarrow \mu_{s(c)}$), stabilizing rare species. Second, the slope-mean hierarchy pools the run-effect means $m_{c,j}$ across `species` within each `species_cluster`, improving stability of run-covariate effects under correlated predictors. Separately, the compound prior pools across compounds through a chemistry-informed mean (ChemBERTa embeddings), which is especially helpful when a compound has limited history.

**Intercepts: species + compound baseline with partial pooling.** We shrink each group intercept toward an additive species and compound baseline:

$$b_{c,k} \sim \mathcal{N}\big(t_0 + \mu_c + \alpha_k,\ \tau_b^2\big)\,.$$

In this prior, $t_0$ denotes a global intercept shared across all groups, $\mu_c$ is a species-specific intercept offset, $\alpha_k$ is a compound baseline shared across species, and $\tau_b$ controls residual group-level variation around the hierarchical mean.

Species offsets are pooled within supercategory:

$$\mu_s \sim \mathcal{N}(0, \tau^2_{\mu,\text{supercat}}), \qquad \mu_c \sim \mathcal{N}\Big(\mu_{s(c)}, \tau^2_{\mu,s(c)}\Big)\,.$$

The mapping $s(c)$ assigns each species $c$ to its enclosing `species_cluster` $s$. $\mu_s$ is a supercategory-level offset, and $\tau_{\mu,\text{supercat}}$ and $\tau_{\mu,s}$ control pooling strength at the supercategory and within-supercategory levels, respectively. We allow supercategory-specific pooling strength by learning $\tau_{\mu,s}$ per supercategory (with a shared prior).

**Compound effects: chemistry-informed prior mean.** We model the compound effect as a chemistry-driven mean plus a residual:

$$z_{h,d} = e_{h,d} - \bar{e}_{d,\text{train}}, \qquad \alpha_k = \sum_{d=1}^{D} z_{h(k),d}\,\theta_{\alpha,d} + \delta_k,$$

$$\theta_{\alpha,d} \sim \mathcal{N}(0, \sigma_\theta^2) \quad (d = 1, \ldots, D), \qquad \delta_k \sim \mathcal{N}(0, \tau^2_{\text{comp}}).$$

Let $e_{h,d}$ be the fixed ChemBERTa embedding for chemical identifier $h$ and dimension $d$, and let $\bar{e}_{d,\text{train}}$ be the training mean for dimension $d$; then $z_{h,d}$ is the centered embedding. The weights $\theta_{\alpha,d}$ map embeddings to a prior mean for the compound baseline $\alpha_k$ (with prior scale $\sigma_\theta$), while $\delta_k$ is a compound-specific residual with scale $\tau_{\text{comp}}$. Centering $z_{h,d}$ by the training mean $\bar{e}_{d,\text{train}}$ and enforcing mean-zero constraints on compound offsets are convenient reparameterizations for identifiability with $t_0$.

**Slopes on run covariates: ridge with hierarchical slope means.** We use a ridge prior on per-group slopes around a species-level slope mean. For each run covariate $j = 1, \ldots, p$:

$$w_{c,k,j} \sim \mathcal{N}\big(m_{c,j},\ \sigma^2/\lambda_{\text{slopes}}\big)\,,$$

where $w_{c,k,j}$ is the slope for group $(c, k)$ on covariate $j$, $m_{c,j}$ is a species-level slope mean, $\sigma^2$ is the residual variance from the likelihood, and $\lambda_{\text{slopes}}$ is a fixed ridge penalty (precision) that controls shrinkage toward $m_{c,j}$. We pool $m_{c,j}$ through the supercategory:

$$w_{0,j} \sim \mathcal{N}(0, \sigma_{w0}^2), \qquad m_{s,j} \sim \mathcal{N}(w_{0,j}, \tau_{w,\text{supercat}}^2), \qquad m_{c,j} \sim \mathcal{N}(m_{s(c),j}, \tau_{w,s(c)}^2).$$

In this hierarchy, $w_{0,j}$ is a global slope mean for covariate $j$ (with prior scale $\sigma_{w0}$), $m_{s,j}$ is a supercategory-level slope mean, and $\tau_{w,\text{supercat}}$ and $\tau_{w,s}$ control pooling strength across supercategories and within each supercategory. As with the intercept hierarchy, we allow supercategory-specific pooling strength by learning $\tau_{w,s}$ per supercategory (with a shared prior).

**What "ridge" means (and why not lasso).** A ridge prior is a Gaussian prior on coefficients. In a linear model, this corresponds to an $\ell_2$ shrinkage penalty: large coefficients are discouraged, but coefficients are not driven exactly to zero. This behavior is a good fit for run covariates that are highly correlated (common for `IS*`/`ES_`): ridge shrinkage distributes weight smoothly across correlated predictors and yields more stable fits. By contrast, lasso corresponds to a Laplace prior ($\ell_1$ penalty) and tends to select an arbitrary subset of correlated covariates, producing unstable coefficients across retrains and brittle behavior in sparse groups.

**Noise and scale priors.** We model residual noise with:

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

and use HalfNormal priors for positive scales such as $\tau_b$, $\tau_{\mu,\text{supercat}}$, $\tau_{\mu,s}$, $\tau_{w,\text{supercat}}$, $\tau_{w,s}$, and $\tau_{\text{comp}}$.

## 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.

**Collapsed slopes (why it is fast).** 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 **collapse** them: conditional on the ridge prior and the run covariates, the slopes can be integrated out analytically. 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 A 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 $t_0 + \mu_c + \alpha_k$.

**Stage-1 coefficient summaries.** 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. **Approximation note:** we compute these group-wise posteriors 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 ($\tau_{\mu,\cdot}$, $\tau_{w,\cdot}$, $\tau_{\text{comp}}$), chemistry weights $\theta_{\alpha,1}, \ldots, \theta_{\alpha,D}$, and the noise scale $\sigma_y$. 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 $\sigma^2_g$ is shared across groups, while the sklearn supercategory ridge baseline estimates a separate $\sigma^2_g$ 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 $\hat{y}_i \pm 1.96\,\hat{\sigma}_i$ (width $2 \cdot 1.96 \cdot \hat{\sigma}_i$). 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 a worksheet-level hold-out split (described in the lasso baseline section below).

# 3 Evaluation

We compare models trained on the cap100 dataset (capped at 100 observations per (species, comp_id) pair) and evaluated on realtest for lib208 and lib209. This report focuses on **seen-compound performance**: the standard realtest 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).

**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**: $\frac{1}{n}\sum_i |\hat{y}_i - y_i|$.

- **Coverage@95%** (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 $\hat{\sigma}_i = \sqrt{\widehat{\mathrm{Var}}(y_i \mid x_{i,1:p})}$. We define the nominal 95% interval as $\hat{y}_i \pm 1.96\,\hat{\sigma}_i$ and compute the fraction covered.

- **Mean interval width** (ridge models): $\frac{1}{n}\sum_i 2 \cdot 1.96 \cdot \hat{\sigma}_i$.

- **Coverage/width (lasso)**: computed from the baseline window $w_i$ via $|\hat{y}_i - y_i| \le w_i$ (width $2w_i$). This is a useful diagnostic but is not a nominal probabilistic interval.

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 $j = 1, \ldots, p$.

We also stratify by **training support bin** of each (`group_id`, `comp_id`) group using bins:

$$\{\le 1,\ 2,\ 3\text{–}5,\ 6\text{–}10,\ 11\text{–}20,\ 21\text{–}50,\ 51\text{–}100,\ > 100\}.$$

Note that `group_id` depends on the model: for the hierarchical ridge model it is `species`, while for the supercategory ridge baseline it is `species_cluster`. Therefore the support-bin plots should be interpreted as "tail vs head" behavior within each model's native grouping.

### 3.1 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 $\ell_2$ penalty for stability under correlated covariates. We attach an approximate Normal predictive distribution by combining an estimated noise scale with an approximate coefficient covariance.

**Windowing.** For each `(species_cluster, comp_id)` group this baseline stores a fitted coefficient mean, an approximate coefficient covariance, and an estimated residual variance $\sigma_g^2$. For a new row, we compute $\hat{y}_i$ and $\hat{\sigma}_i$ and use the nominal 95% window $\hat{y}_i \pm 1.96\,\hat{\sigma}_i$ (as defined above). This yields a row-specific window rather than a fixed width per model.

### 3.2 Baseline currently in production: supercategory lasso

The production baseline is a bank of lasso regressions fit per `species_cluster`. Lasso is attractive for sparse solutions, but its $\ell_1$ 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.

**Windowing in Hippopotamus.** In Hippopotamus, lasso coefficients are fit on a worksheet-level train split, and then a fixed window is attached using a hold-out split (commonly an 80/20 split by worksheet). The window is computed empirically from the held-out 20% as a high percentile of absolute prediction errors (residuals) and stored in the model bundle as a per-model `window`. At scoring time, the interval is $\hat{y}_i \pm w$ (width $2w$), where $w$ is the same for every row scored by that `(species_cluster, comp_id)` model. This is a pragmatic way to set windows, but it is not derived from an explicit predictive distribution and it does not provide row-specific uncertainty; the window depends on the chosen split and the chosen percentile.

# 4 Results (seen compounds)

## 4.1 Global performance

Table 1 summarizes global metrics on realtest for lib208 and lib209. Lasso baselines do not score all rows; **Rows scored** reflects applicability.

Table 1: Global realtest metrics for cap100-trained models (linear features).

| Lib | Model | RMSE | MAE | Cov95 | Width95 | Rows scored | Train (min) |
|---|---|---|---|---|---|---|---|
| 208 | Ridge (supercategory, sklearn) | 0.009023 | 0.004673 | 0.943 | **0.023534** | **100.0%** | **0.0** |
| 208 | Ridge (partial pooling, chem-linear) | **0.007874** | **0.003821** | **0.953** | 0.026681 | **100.0%** | 107.8 |
| 208 | Lasso (supercategory) | 0.015081 | 0.007956 | 0.954 | 0.058908 | 94.7% | – |
| 209 | Ridge (supercategory, sklearn) | 0.008317 | 0.004677 | 0.920 | **0.021393** | **100.0%** | **0.0** |
| 209 | Ridge (partial pooling, chem-linear) | **0.007707** | **0.004285** | **0.956** | 0.027518 | **100.0%** | 182.9 |
| 209 | Lasso (supercategory) | 0.009439 | 0.004954 | 0.993 | 0.050886 | 97.8% | – |

**Interpretation.** Relative to Ridge (supercategory, sklearn), Ridge (partial pooling, chem-linear) 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.

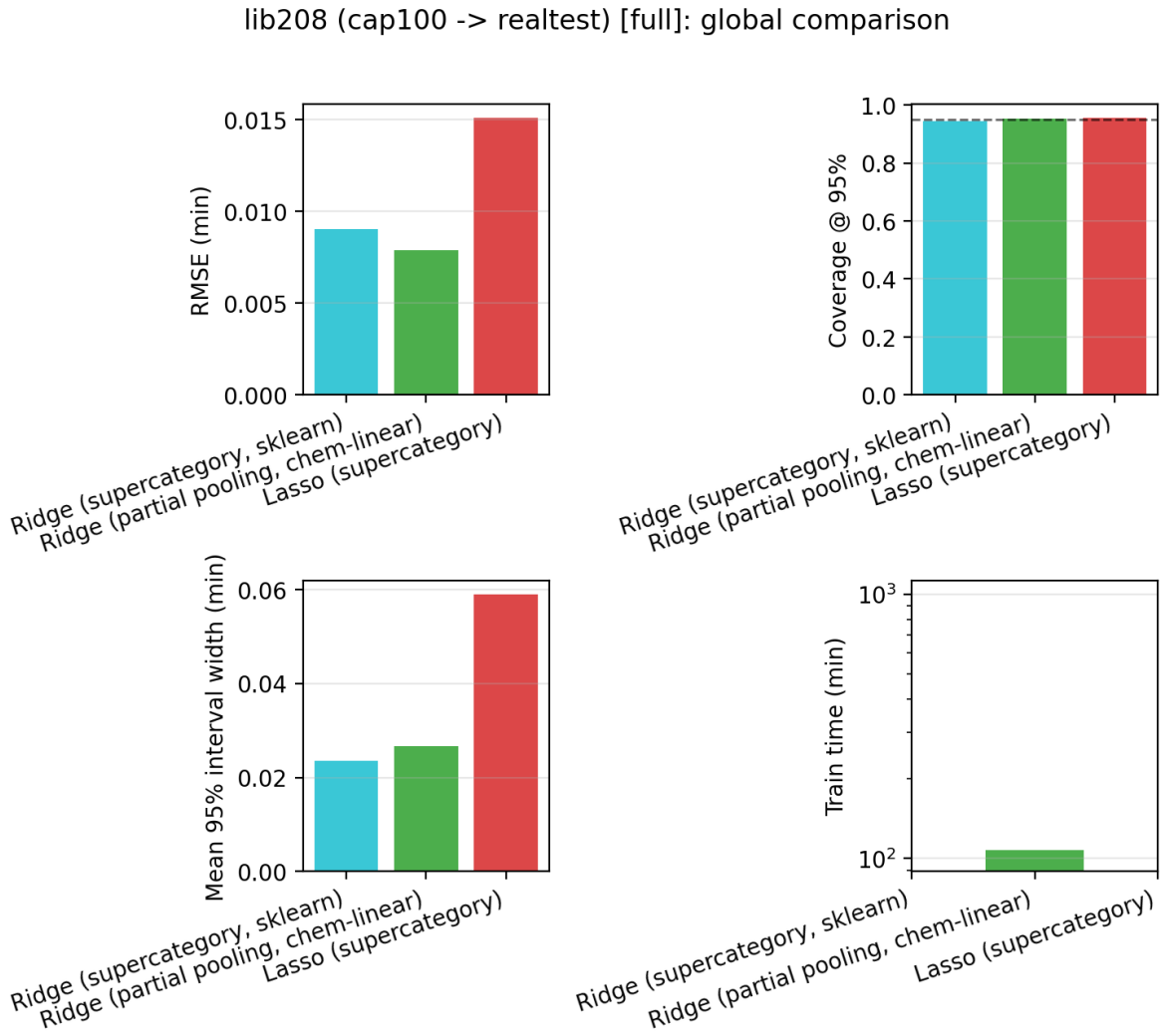


Figure 1: Global comparison across report baselines for lib208 (cap100 training, realtest evaluation). Panels show RMSE, coverage@95%, mean interval width, and training time.

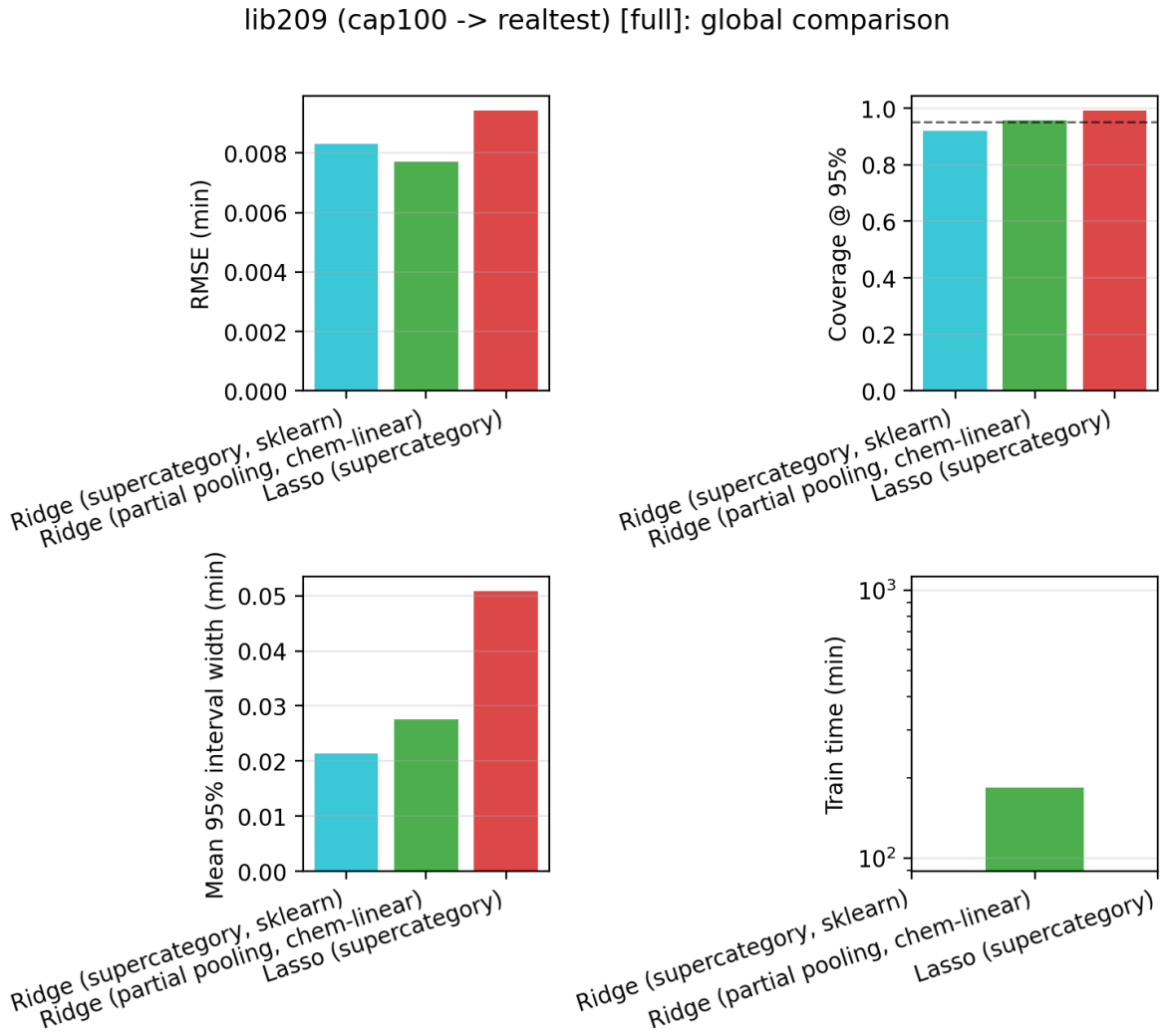


Figure 2: Global comparison across report baselines for lib209 (cap100 training, realtest evaluation). Panels show RMSE, coverage@95%, mean interval width, and training time.

### 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 $\sim$0.92–0.94). The hierarchical model yields coverage closer to nominal at moderately larger width.

### 4.3 Performance by training support bin

Figures 3 and 4 break metrics down by training support bin (long tail to head).

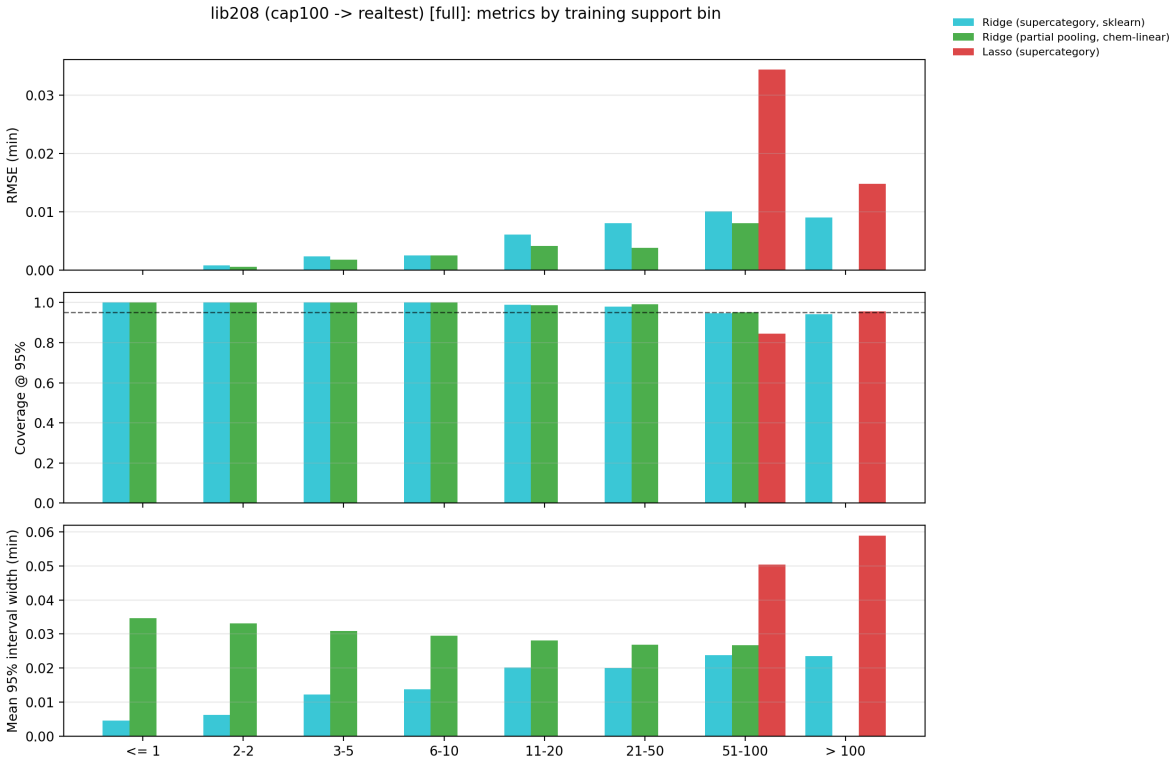


Figure 3: Metrics by training support bin for lib208 report baselines. Coverage@95% is empirical coverage of the nominal 95% prediction interval (ridge models) or window interval (lasso baseline).

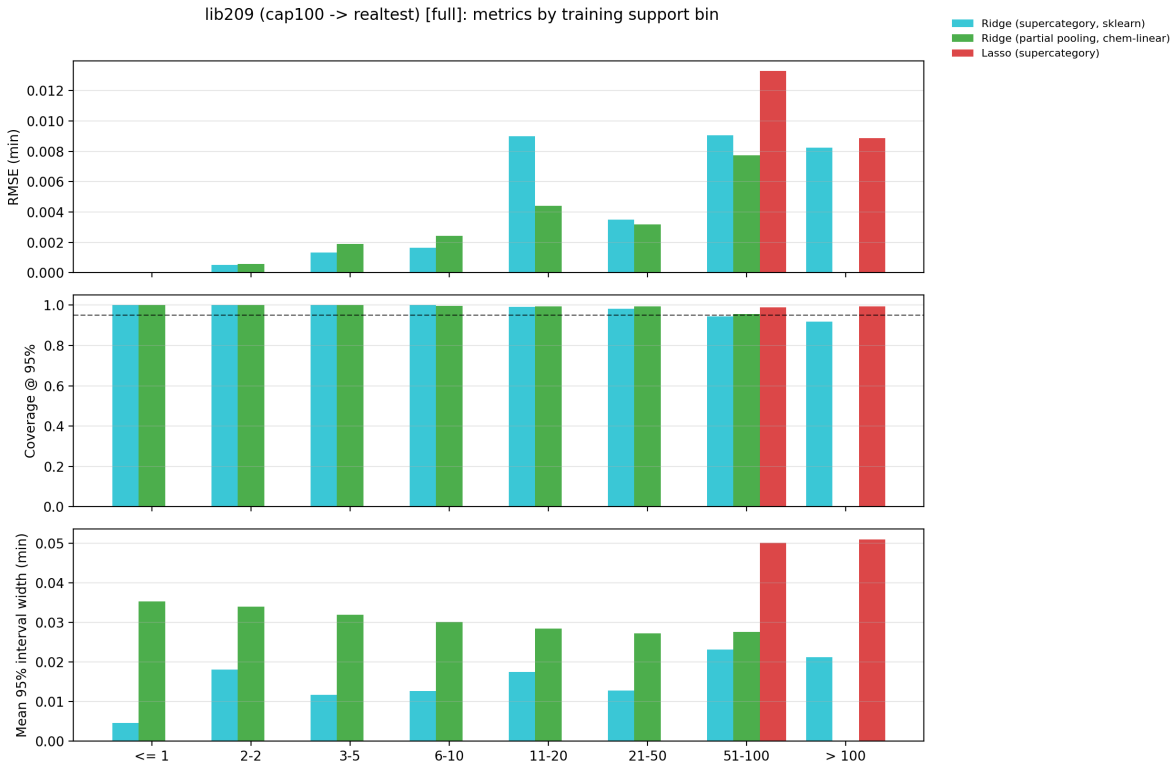


Figure 4: Metrics by training support bin for lib209 report baselines. Coverage@95% is empirical coverage of the nominal 95% prediction interval (ridge models) or window interval (lasso baseline).

## 4.4 Performance by species_cluster

Figures 5 and 6 aggregate metrics by `species_cluster`.

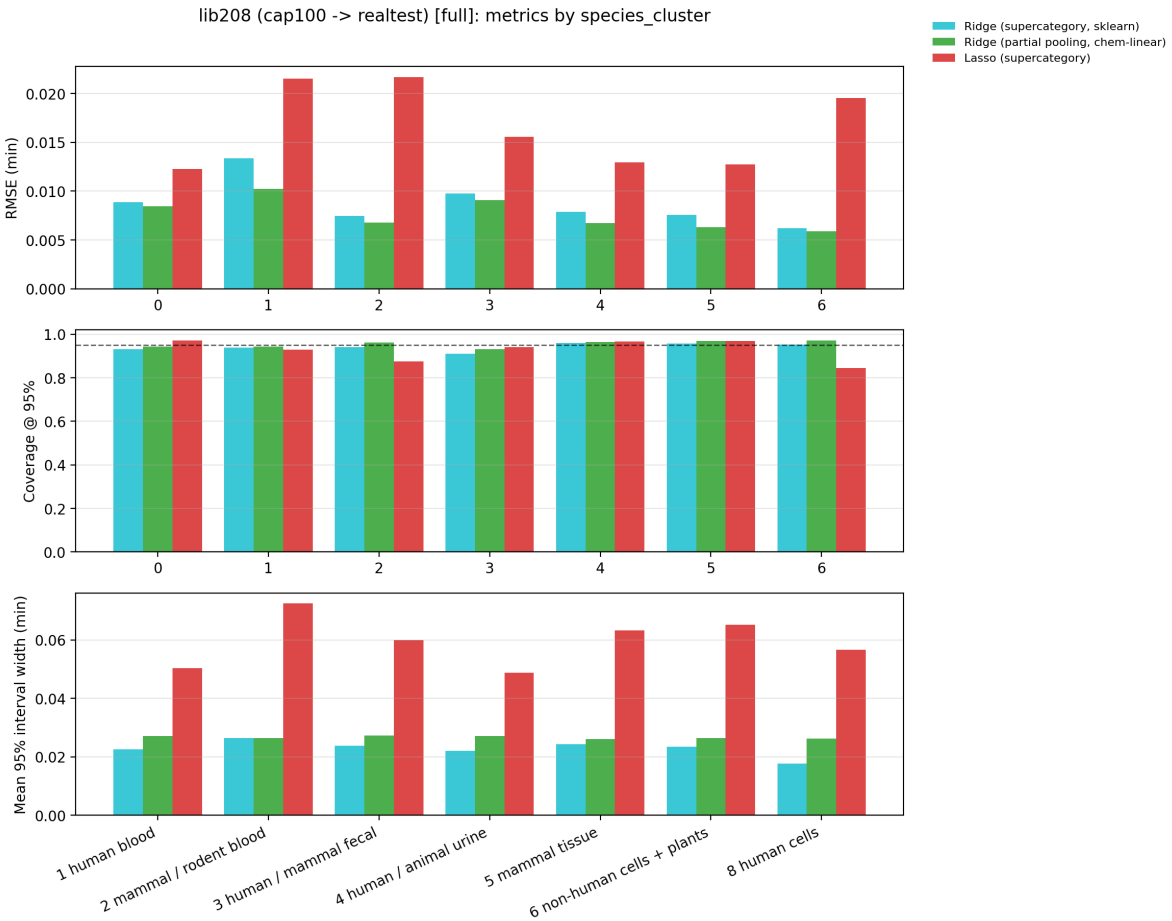


Figure 5: Metrics by `species_cluster` (supercategory) for lib208 report baselines.

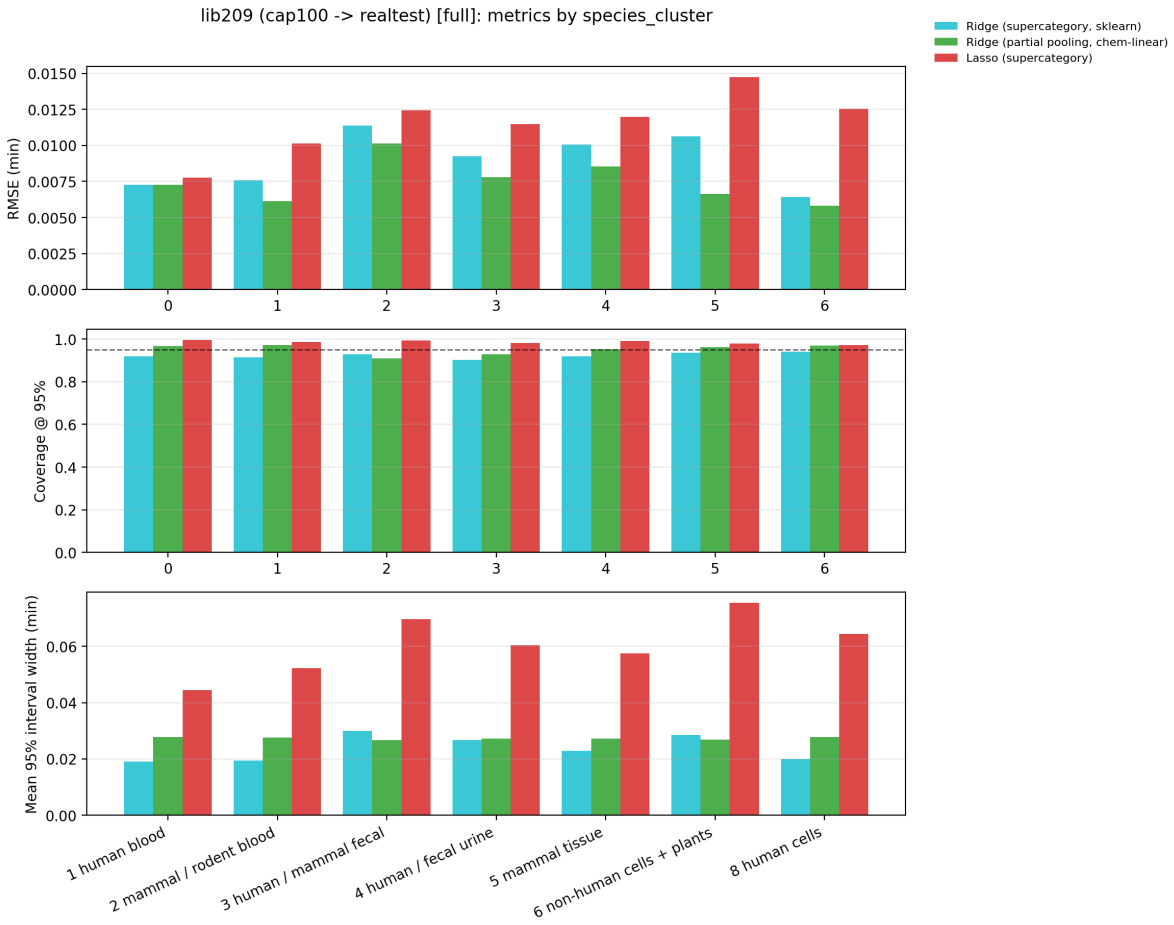


Figure 6: Metrics by `species_cluster` (supercategory) for lib209 report baselines.

# 5 Discussion

Across both libraries, the hierarchical ridge model improves point accuracy modestly but, more importantly for peak assignment, produces predictive windows with coverage close to the nominal 95% target. The fast sklearn ridge baseline is competitive in RMSE but systematically under-covers because its predictive variance is optimistic for many supercategory–compound pairs. In a pipeline that uses RT windows as a hard filter, this under-coverage is not a cosmetic issue: it directly increases the probability that the true peak is excluded from consideration.

The hierarchical model is designed around the data realities of RT regression. Species–compound history is sparse and heavy-tailed, so independent per-group fits are unstable in the tail. Partial pooling addresses this by shrinking group intercepts and slope means toward supercategory-aware priors, improving stability without sacrificing head performance. Chemistry enters as a prior mean for compound effects, which lets the model share information across chemically similar compounds and stabilizes estimates when per-compound support is limited.

The main practical downside 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.

# 6 Future extensions

This section summarizes exploratory results and follow-up work needed for true cold-start scenarios.

## 6.1 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 while keeping the corresponding `(species_cluster 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 2: 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, chem-linear) | 0.014968 | 0.009012 | 1.000 | **100.0** |
| `species` | Ridge (supercategory, sklearn) | 0.011090 | 0.005202 | 0.877 | **100.0** |
| `species` | Lasso (supercategory, external) | **0.007913** | **0.004174** | **0.995** | 94.8 |
| `species_comp` | Ridge (partial pooling, chem-linear) | 0.013474 | 0.009569 | 1.000 | **100.0** |
| `species_comp` | Ridge (supercategory, sklearn) | 0.013272 | 0.006532 | 0.812 | **100.0** |
| `species_comp` | Lasso (supercategory, external) | **0.007671** | **0.004331** | **0.999** | 94.9 |

Table 3: 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, chem-linear) | 0.008381 | 0.005281 | 1.000 | **100.0** |
| `species` | Ridge (supercategory, sklearn) | 0.008268 | 0.004870 | 0.902 | **100.0** |
| `species` | Lasso (supercategory, external) | **0.006779** | **0.004267** | **0.996** | 92.4 |
| `species_comp` | Ridge (partial pooling, chem-linear) | 0.011684 | 0.006456 | 1.000 | **100.0** |
| `species_comp` | Ridge (supercategory, sklearn) | 0.013582 | 0.006320 | 0.883 | **100.0** |
| `species_comp` | Lasso (supercategory, external) | **0.009482** | **0.005677** | **0.997** | 95.0 |

**Interpretation.** 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.

### 6.2 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 4: Zero-shot chemistry results on the lib208 production subset (hold out 20 `chem_id`; seed=42).

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

Table 5: Zero-shot chemistry results on the lib209 production subset (hold out 20 `chem_id`; seed=42).

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

**Interpretation.** 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 $\delta_k$ (so seen-compound performance is maintained) is the key next step.

**Promising next steps.** 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.

## 7 Conclusion

For seen-compound prediction (cap100 $\rightarrow$ realtest), the hierarchical ridge model with chemistry-informed compound effects provides the best overall behavior for peak assignment: competitive point accuracy and predictive windows with coverage close to nominal. The supercategory ridge baseline remains useful for

speed, but its optimistic intervals make it risky as a filtering model without explicit calibration. The lasso baseline highlights the limitations of non-hierarchical, sparsity-driven models under correlated run covariates and heuristic uncertainty windows.

# A 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.

## A.1 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 $\sigma^2$: 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 $m_g$.

### A.2 Goal: the marginal likelihood

We want the marginal likelihood obtained by integrating out $w_g$:

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

### A.3 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.

### A.4 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 $\epsilon_g$ are independent and both are mean-zero Gaussians.

### A.5 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 $\epsilon_g$ 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:

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

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

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

Putting the two pieces together:

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

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 $b_g, m_g, \sigma^2, \lambda_{\text{slopes}}$.

### A.6 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 $\sigma^2$:

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

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 $C_g$?

### A.7 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:

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

**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:

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

**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:

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

### A.8 Final collapsed log-likelihood (putting it all together)

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

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

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

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.

### A.9 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 $b_g$.

**Start from likelihood and prior.** Write the centered response as $\tilde{y}_g = y_g - b_g \mathbf{1}_{n_g}$. Then 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:

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

**Write the unnormalized log posterior.** Up to an additive constant (that does not depend on $w_g$),

$$\begin{aligned}\log p(w_g \mid \tilde{y}_g, m_g, \sigma^2, \lambda_{\text{slopes}}) &\propto \log p(\tilde{y}_g \mid w_g, \sigma^2) + \log p(w_g \mid m_g, \sigma^2, \lambda_{\text{slopes}}) \\ &\propto -\frac{1}{2\sigma^2}\|\tilde{y}_g - X_g w_g\|_2^2 - \frac{\lambda_{\text{slopes}}}{2\sigma^2}\|w_g - m_g\|_2^2.\end{aligned}$$

**Expand both squared norms.** First expand the likelihood term:

$$\begin{aligned}\|\tilde{y}_g - X_g w_g\|_2^2 &= (\tilde{y}_g - X_g w_g)^\top (\tilde{y}_g - X_g w_g) \\ &= \tilde{y}_g^\top \tilde{y}_g - 2 w_g^\top X_g^\top \tilde{y}_g + w_g^\top X_g^\top X_g w_g.\end{aligned}$$

Next expand the prior term:

$$\begin{aligned}\|w_g - m_g\|_2^2 &= (w_g - m_g)^\top (w_g - m_g) \\ &= w_g^\top w_g - 2 w_g^\top m_g + m_g^\top m_g.\end{aligned}$$

**Collect 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:

$$-\frac{1}{2\sigma^2}\left[w_g^\top A_g w_g - 2 w_g^\top h_g\right].$$

**Complete the square.** For any symmetric positive-definite matrix $A$ and vector $h$, we have:

$$w^\top A w - 2 w^\top h = (w - A^{-1}h)^\top A (w - A^{-1}h) - h^\top A^{-1} h.$$

Applying this with $A = A_g$ and $h = h_g$, the posterior is a Gaussian with mean $A_g^{-1} h_g$ and covariance $\sigma^2 A_g^{-1}$:

$$w_g \mid y_g, b_g, m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}\big(A_g^{-1} h_g,\ \sigma^2 A_g^{-1}\big)\,.$$

**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:

$$w_g \mid y_g, b_g, m_g, \sigma^2, \lambda_{\text{slopes}} \sim \mathcal{N}\big(m_g + A_g^{-1} s_g,\ \sigma^2 A_g^{-1}\big)\,.$$

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.