RFAMFLOW: NESTED GUIDANCE FOR RNA FAMILY GENERATION WITH DISCRETE FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing RNA sequences that belong to a target family is central to RNA engineering. Existing generators either train a separate model per family, failing when a family has few known sequences, or condition a single autoregressive model on the family label, still degrading on the many small and medium families of the highly imbalanced Rfam database. We present RfamFlow, a conditional discrete flow-matching model that generates RNA sequences for any Rfam family from its tag alone, with no structural input. RfamFlow couples a Diffusion Transformer (DiT) with a nested classifier-free guidance scheme that follows the Rfam taxonomy, allowing a data-poor family to borrow a prior from its coarser taxonomic levels. The single trained model supports *de novo* generation, masked reconstruction, and zero-shot fitness prediction without task-specific fine-tuning. RfamFlow outperforms the comparable family-conditioned baseline models across large, medium, and small families, with its nested guidance helping most where family data are scarcest. Furthermore, it matches far larger RNA foundation models on reconstruction and fitness prediction.

1 INTRODUCTION

RNA molecules not only carry genetic information but also perform a remarkable range of functions. They catalyze reactions as ribozymes (Doudna & Cech, 2002; Flores-García et al., 2026) and, as riboswitches, sense small molecules and metabolites to regulate gene expression (Mandal & Breaker, 2004; Chen & Liu, 2025). This functional versatility makes RNA a powerful substrate for engineering, with applications spanning gene regulation (Tong & Tong, 2026), biosensors (Zhu et al., 2025), and therapeutics (Chen et al., 2025). Realizing these applications hinges on *RNA design*: generating novel sequences that reliably fold into a target structure and carry out a desired function. Yet the space of possible sequences is astronomically large while functional sequences are exceedingly rare (Calvanese et al., 2024), making automated RNA design a central challenge in computational biology.

Deep learning has been introduced into RNA sequence design and has progressed rapidly. One classical direction is *inverse folding*, which searches for sequences that fold into a given target structure (Rubio-Largo et al., 2023; Obonyo et al., 2024; Kubaney et al., 2025). It presupposes that structure, yet as of June 2026 the Protein Data Bank (PDB) (Burley et al., 2017) holds only about 2,200 RNA structures, and RNA's conformational flexibility makes any single static structure an unreliable target (Zhou & Chen, 2024). In contrast, sequence data are easier to collect, and large RNA sequence databases such as Rfam (Griffiths-Jones et al., 2003) and RNAcentral (The RNAcentral Consortium, 2026) have been established, shifting research attention toward sequence-based design and bypassing the low-reliability problem of structural data. Conditional generation, which constrains the output by an attribute such as family or function, is more useful in practice than unconditional generation. Rfam organizes the RNA universe into families whose members share similar functions (e.g., RF00174, the cobalamin riboswitch, binds cobalamin/vitamin B_{12}), providing fertile ground for training conditional generative models. Two paradigms have emerged: *family-specific* (Sumi et al., 2024; Gao & Lu, 2024) and *family-conditioned* (Sun et al., 2024).

Family-specific models, however, train a separate generator from scratch on each family's sequences alone. They share no information across families and falter when a family is data-poor. On the other hand, existing family-conditioned models have so far relied on *autoregressive decoding*, whose left-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

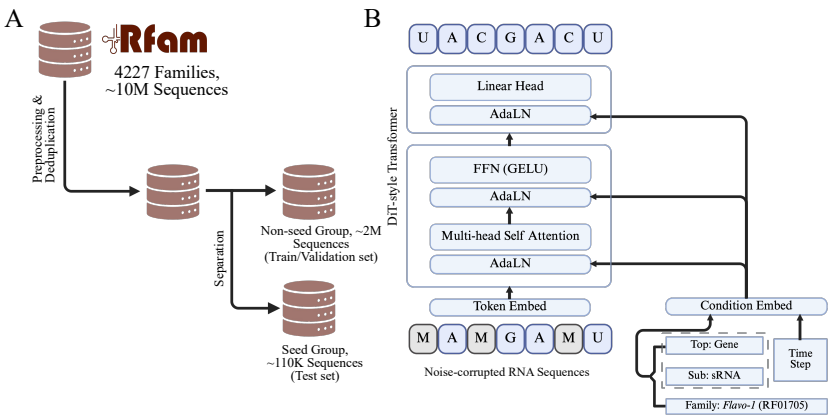


Figure 1: Overview of RfamFlow. **(a)** Data processing: sequences from each Rfam family are preprocessed, deduplicated, and separated into training/test splits. **(b)** Model and training: a DiT-style Transformer with a dedicated condition-embedding block is trained with discrete flow-matching to denoise sequences conditioned on an Rfam family tag.

to-right factorization cannot naturally support masked reconstruction or bidirectional conditioning. Both are further limited by the highly imbalanced distribution of RNA families: their generation quality depends heavily on the amount of data available per family and degrades sharply on small- and medium-sized families. Meanwhile, flow-matching (Lipman et al., 2022) has achieved remarkable success in image and text generation and has begun to be applied to biomolecule design (Morehead et al., 2026; Liang et al., 2026), offering an alternative for sequence generation.

To address these gaps, we propose RfamFlow, a conditional discrete flow-matching model for RNA sequence generation (Figure 1). Conditioned solely on an Rfam family tag, RfamFlow generates sequences across diverse RNA families without requiring any structural information. Inspired by classifier-free guidance (CFG) in image generation (Ho & Salimans, 2022), we introduce a *nested* CFG scheme that exploits the hierarchical structure of the Rfam taxonomy. We evaluate *de novo* generation on large-, medium-, and small-sized families ($\geq$ 1,000, 100–999, and 10–99 sequences), where RfamFlow outperforms the comparable single-model baseline, particularly on small families, demonstrating the value of nested CFG. In addition, we conduct two prediction tasks: masked sequence reconstruction and zero-shot fitness prediction, comparing against several baselines. Although neither is a direct training objective, RfamFlow matches much larger dedicated RNA foundation models.

In summary, our main contributions are:

- We propose RfamFlow, the first conditional *discrete* flow-matching model for family-conditioned RNA generation. A single model spans diverse RNA families and requires no structural information.
- We introduce *nested* classifier-free guidance, a guidance scheme that exploits the hierarchical structure of the Rfam taxonomy and is especially effective for small- and medium-sized families.
- Within one framework, RfamFlow unifies three tasks without task-specific training: *de novo* generation, masked sequence reconstruction, and zero-shot fitness prediction.
- Across large-, medium-, and small-sized families, RfamFlow sets a new state of the art among family-conditioned generators, and on reconstruction and fitness prediction it remains competitive with dedicated RNA foundation models that are far larger.

2 Related Work

Family-specific generation. RfamGen (Sumi et al., 2024) was among the first deep generative approaches to design sequences for different RNA families. It couples a variational autoencoder

(VAE) with a covariance model (CM) so that the family alignment and its consensus secondary structure directly guide learning, and draws new sequences from the resulting continuous latent space. The cost is that a separate model must be fit for each family from a precomputed MSA. RNACG (Gao & Lu, 2024) casts several RNA design tasks, from inverse folding to family-specific generation, as conditional flow-matching, but likewise trains one model per family and covers only three of them, each fitted to the Rfam sequences with solved 3D structures. Its inverse-folding mode further depends on scarce 3D structures, which limits its use in RNA engineering, and its generation needs the sequence length to be fixed at sampling time. As a result, these two models are *family-specific*, so they transfer nothing across families and struggle when a family offers few aligned sequences. We adopt both as baselines, RNACG on its three families (Section 4.2).

Family-conditioned generation. RfamLlama (Sun et al., 2024) was among the first to take this route: a single Llama-style autoregressive model is trained on the entire Rfam database and conditioned on a family tag to generate sequences for any family, following the control-tag conditioning popularized by protein language models such as ProGen (Madani et al., 2023) and ZymCTRL (Munsamy et al., 2022). However, its autoregressive, left-to-right factorization conditions each nucleotide only on its predecessors, whereas RNA function is largely governed by base pairing between distant positions (Tinoco Jr & Bustamante, 1999). Such bidirectional, long-range dependencies are awkward to capture one token at a time. Moreover, its family conditioning tokenizes the five digits of the Rfam accession (e.g., RF00174) individually. This suits hierarchical labels such as enzyme EC numbers, where each digit marks a level of a genuine functional hierarchy so that shared digits imply relatedness. Rfam accessions, by contrast, are assigned serially and carry no such meaning, so this digit-level scheme conveys no genuine relatedness and overlooks the clan/family hierarchy of Rfam.

Discrete flow-matching and classifier-free guidance. Flow-matching learns a continuous-time generative process by regressing the velocity field of a probability path from noise to data (Lipman et al., 2022), giving simulation-free training and fast sampling. As RNA sequences are categorical, RfamFlow builds on *discrete* flow-matching, which transports probability mass over discrete states and has proven effective for text and biomolecular generation (Gat et al., 2024; Campbell et al., 2024). For conditional control, classifier-free guidance (CFG) (Ho & Salimans, 2022), originally proposed for diffusion, interpolates between conditional and unconditional predictions without a separate classifier. The concurrent LineageFlow (Liang et al., 2026) also pairs flow-matching with family conditioning, but obtains it by swapping the noise prior for a Dirichlet prior estimated from each family's alignment and phylogeny, and it targets protein families, leaving guidance over a family taxonomy still open.

3 METHOD

3.1 PROBLEM FORMULATION

We represent an RNA sequence as a token sequence $x = (x^1, \ldots, x^L)$ over a fixed alphabet $\mathcal{V} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$ augmented with special tokens (padding, boundary markers, and a mask symbol that serves as the absorbing state) and padded to a common length L. Each sequence is annotated with the Rfam family it belongs to. Because Rfam organizes families into a taxonomy, we encode a family as a hierarchical label $c = (c_{\text{top}}, c_{\text{sub}}, c_{\text{fam}})$, giving its coarse type, finer subtype, and family identity, respectively. The training set $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^{N}$ spans K families whose sizes are highly uneven. Our goal is to learn a *single* conditional generative model $p_\theta(x \mid c)$ that captures the sequence distribution of every family, including small ones, by sharing parameters both across families and across the levels of the taxonomy. Once trained, the same model supports three tasks without task-specific retraining (Figure 2): (i) *de novo* generation by sampling $x \sim p_\theta(\cdot \mid c)$ for a target family; (ii) masked reconstruction by infilling masked positions of a partially observed sequence; and (iii) zero-shot fitness prediction by scoring variants with the model likelihood $p_\theta(x \mid c)$.

3.2 DATA CURATION

We build our corpus from Rfam 15.1 (Ontiveros-Palacios et al., 2025), which contains $4{,}227$ families and roughly 10M sequences, and reduce redundancy within each family by clustering at 90% identity with MMseqs2, keeping one representative per cluster (Appendix A). Each family has a small,

manually curated *seed* alignment that defines it and builds its covariance model (CM), and a much larger *non-seed* set collected by searching the database with that CM. We train and validate on the non-seed sequences, as RfamGen does (Sumi et al., 2024), and hold out the seed sequences for testing. The family-size distribution is highly imbalanced (Figure 4). To probe how generation quality depends on data size, we group families by non-seed size into *large* ($\geq 1{,}000$ sequences), *medium* (100–999), and *small* (10–99), and randomly sample 100 families from each group for the downstream tasks.

3.3 Discrete Flow-Matching

Discrete flow-matching (Campbell et al., 2024; Gat et al., 2024) learns to transport a simple source distribution p_0 to the data distribution p_1 over the discrete space $\mathcal{V}^L$ along a path $\{p_t\}_{t\in[0,1]}$. We use an *absorbing* source, in which every token of $x_0 \sim p_0$ is a special mask symbol, together with a linear masking schedule applied independently per position. Given a clean sequence $x_1 \sim p_1(\cdot \mid c)$, the intermediate state x_t is obtained at each position i by

$$x_t^i = \begin{cases} x_1^i & \text{with probability } t, \\ x_0^i & \text{with probability } 1-t, \end{cases} \tag{1}$$

so each position is independently unmasked (set to its clean value) with probability t; hence x_0 is the fully masked sequence at $t = 0$ and p_1 recovers the data at $t = 1$. This path coincides with that of masked diffusion models (Appendix B.6). Our denoiser is a DiT-style Transformer (Peebles & Xie, 2023) that injects the conditioning signal through adaptive layer normalization (AdaLN): in each block, the signal predicts channel-wise scale and shift that modulate the normalized activations, and a learned gate scales the residual branch. Unlike AdaLN-Zero, we do *not* zero-initialize these residual gates; since our base model is shallow (8 layers), random initialization trains readily and converges faster in our setting. The conditioning signal fuses the family label c and the time t. Each is mapped by its own embedding to a half-width vector, and the two are concatenated into a single conditioning vector. A standard sinusoidal embedding is used for t. For *nested* CFG we maintain three embedding tables, one for the top, sub, and family levels, and sum their embeddings to form the class signal. The flat-CFG baseline (standard CFG over a single family label) instead uses a single embedding table over families. Full model configurations and the per-block forward pass are given in Appendix B.

3.4 Training and Sampling

The denoiser network f_θ predicts the clean tokens from the corrupted state. At each position, the network outputs a distribution $p_\theta(x^i \mid x_t, c, t)$ over $\mathcal{V}$, conditioned on the family label c and time t.

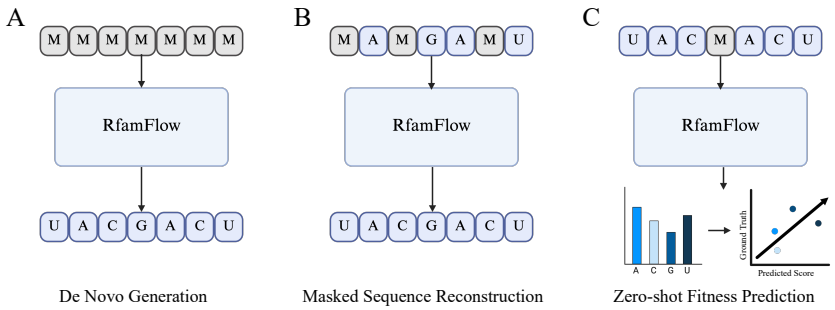


Figure 2: The three tasks a single trained RfamFlow model supports. **(a)** *De novo* generation: sampling new sequences for a target family. **(b)** Masked reconstruction: infilling masked positions of a partially observed sequence. **(c)** Zero-shot fitness prediction: scoring sequence variants by the model likelihood. M denotes the masked token, the absorbing state of the discrete flow.

We train it with a denoising cross-entropy loss,

$$\mathcal{L}(\theta) = \mathbb{E}\left[-\sum_{i=1}^{L} \log p_\theta(x_1^i \mid x_t, c, t)\right], \tag{2}$$

where the expectation is over $(x_1, c) \sim \mathcal{D}$, $t \sim \mathcal{U}[0,1]$, $x_0 \sim p_0$, and x_t drawn from the path above.

To enable CFG, we randomly replace conditioning labels with a null symbol $\varnothing$ during training. For *nested* CFG, the labels are dropped as a suffix of the hierarchy: the top level is always kept, and the family and sub levels are dropped with probability 0.15 and 0.10, coupled so that dropping the sub level always drops the family level too. This lets us guide each level separately at sampling time. The flat-CFG baseline instead drops its single family label with probability 0.15. Full training hyperparameters are given in Appendix B.3.

For inference we start from the fully masked prior $x_0 \sim p_0$ and decode with confidence-based iterative unmasking (Chang et al., 2022), which suits the absorbing path. Over N steps a cosine schedule sets the cumulative fraction of committed positions after step k to $\tau_k = t_{\max}(1 - \cos\frac{\pi k}{2N})$, revealing few positions early and many later. At each step the denoiser scores every masked position by its peak predicted probability $\max_j p_\theta(j \mid x_t, c, t)$, commits the highest-scoring positions to their predicted tokens, and freezes them. A Gumbel perturbation of the score, annealed as more positions are revealed, keeps samples diverse. Freezing lets each successive forward pass condition on more correct context, where the denoiser is most accurate. We also compare against the CTMC jump (Campbell et al., 2024) and Euler (Gat et al., 2024) solvers in Appendix C.2.

At every sampling step, guidance combines the model's log-probabilities under different conditionings and renormalizes. We consider $p(a) \equiv p_\theta(\cdot \mid a)$ for the distribution under conditioning a. The *flat-CFG* baseline anchors on the global null $\varnothing$ and takes a single step of size γ toward the family label,

$$\log \tilde{p}_\theta \propto \log p(\varnothing) + \gamma\left[\log p(c) - \log p(\varnothing)\right]. \tag{3}$$

Nested CFG maintains this increment form but replaces the single step with a telescoping sum over the hierarchy, using each coarser level as the anchor for the next. The full label is a triple $(c_{\text{top}}, c_{\text{sub}}, c_{\text{fam}})$, and setting a level to $\varnothing$ drops it. Starting from the top-only anchor, we add the sub and family levels one at a time,

$$\begin{aligned}\log \tilde{p}_\theta \propto\ & \log p_1 + w_{\text{sub}}\,(\log p_2 - \log p_1) \\ & + w_{\text{fam}}\,(\log p_3 - \log p_2),\end{aligned} \tag{4}$$

where $p_1 = p(c_{\text{top}}, \varnothing, \varnothing)$, $p_2 = p(c_{\text{top}}, c_{\text{sub}}, \varnothing)$, and $p_3 = p(c_{\text{top}}, c_{\text{sub}}, c_{\text{fam}})$ keep the top level, the top and sub levels, and all three levels, respectively. The two increments add the information contributed by the sub and family level, weighted by their own guidance scales w_{sub} and w_{fam}. Setting $w_{\text{sub}} = w_{\text{fam}} = \omega$ collapses the sum to a single top-anchored guidance step. Additionally, to attribute the gain either to weighting the levels separately or to fusing the three level embeddings, we evaluate that collapsed form as a baseline in its own right, the *diagonal* variant,

$$\log \tilde{p}_\theta \propto \log p_1 + \omega\,(\log p_3 - \log p_1), \tag{5}$$

which is the diagonal of the nested guidance grid swept in Appendix C.1. It keeps the three level embeddings but amplifies everything below the top level with one scale, and because the top level is never dropped in training, its anchor p_1 is a conditional the model has been trained on, so no untrained all-null state is needed. The scales γ and $(w_{\text{sub}}, w_{\text{fam}})$ are selected by a sweep, reported in the experiments.

4 Experiments

4.1 Experimental Setup

The full family lists for the three cohorts are given in Appendix A. For *de novo* generation, we sample 100 sequences per family and score each against the family covariance model with Infernal `cmsearch` (Nawrocki & Eddy, 2013), whose bit score is the standard measure of family membership and serves as our gold standard for generation quality. For masked reconstruction, we use the held-out seed sequences as the test set and randomly mask nucleotides at three rates (0.15, 0.3, and

0.5), measuring how well the model recovers the masked positions. For zero-shot fitness prediction we use three deep mutational scanning (DMS) assays curated by RNAGym (Arora et al., 2025) whose Rfam family falls in our large cohort, so that every assay is scored by a family the model has seen in quantity: tRNA (RF00005) (Li et al., 2016), the glmS ribozyme (RF00234) (Andreasson et al., 2020), and the CPEB3 ribozyme (RF00622) (Zhang et al., 2020). Each is a substitution scan, so all variants within an assay share one length, and every method is given the assayed sequences as they are. Details are in Appendix C.5.

4.2 Main Results

For *de novo* generation we compare RfamFlow against the family-specific RfamGen (Sumi et al., 2024) and the family-conditioned RfamLlama (Sun et al., 2024). We also generate sequences with Infernal's own sampler, cmemit (Nawrocki & Eddy, 2013), which samples directly from each family's covariance model, so it serves as an additional family-specific baseline. For every family in the large, medium, and small groups we generate 100 sequences and score each with `cmsearch`. We report the macro-average bit score, i.e. the mean over families of each family's average bit score, together with each method's family coverage, the fraction of families for which it produces any output, which exposes methods that fail on data-poor families. Results are summarized in Table 1.

Table 1: *De novo* generation quality as macro-average `cmsearch` bit score, with family coverage out of 100 in parentheses. Best results are in bold throughout this paper.

Method	Large	Medium	Small
Natural sequences			
Seed	109.8	106.4	110.4
Non-seed	77.5	74.9	80.0
Family-specific			
RfamGen	**120.6** (91)	77.0 (74)	19.4 (77)
cmemit	59.8 (100)	51.5 (100)	52.2 (100)
Family-conditioned			
RFamLlama-Large	54.2 (100)	33.4 (100)	15.0 (100)
RfamFlow-base (*nested*)	78.9 (100)	57.5 (100)	24.4 (100)
RfamFlow-large (*nested*)	83.3 (100)	**77.0** (100)	**53.9** (100)

RfamFlow outperforms the family-conditioned RfamLlama at every resource level, and notably, the gap grows as families shrink. Against the family-specific RfamGen, the outcome depends on family size. RfamGen trains a dedicated CM-guided VAE on each family, so it works well only when a family has enough aligned sequences to fit its own model, and it leads on the large cohort. On the medium cohort RfamFlow-large matches it while covering 26 more families. On the small cohort, where many families have too few sequences to align and train, RfamGen reaches only about a third of RfamFlow-large's score, because RfamFlow shares parameters across families. Sampling from the scoring model itself does not give an upper bound, as cmemit samples the covariance model's entire emission distribution, so nearly all of its samples are hits but most score only moderately. Overall, apart from trailing RfamGen on the large cohort, RfamFlow-large matches or outperforms every baseline, with the clearest margin on the small cohort. Moving from the base to the large model raises scores on every cohort and more than doubles the small-cohort score. Baseline reproduction details are provided in Appendix C.3.

Beyond family membership, we ask whether the generated sequences reproduce the biophysical hallmarks of their target family, measuring GC content and the minimum free energy (MFE) predicted by RNAfold (Lorenz et al., 2011). In Figure 3, which plots each generator's per-family median against the natural non-seed median for five families per cohort taken at evenly spaced ranks of that natural median, the generators follow the diagonal rather than collapsing to a global average, and RfamFlow-large stays consistently close to it on every cohort, indicating that its samples track each family's natural non-seed distribution and lie about as close to the held-out seed as natural non-seed sequences do (Appendix C.3, which also gives the full comparison over all 100 families of each cohort). Over all families, RfamFlow-large correlates with the natural medians at $r \geq 0.97$ for GC

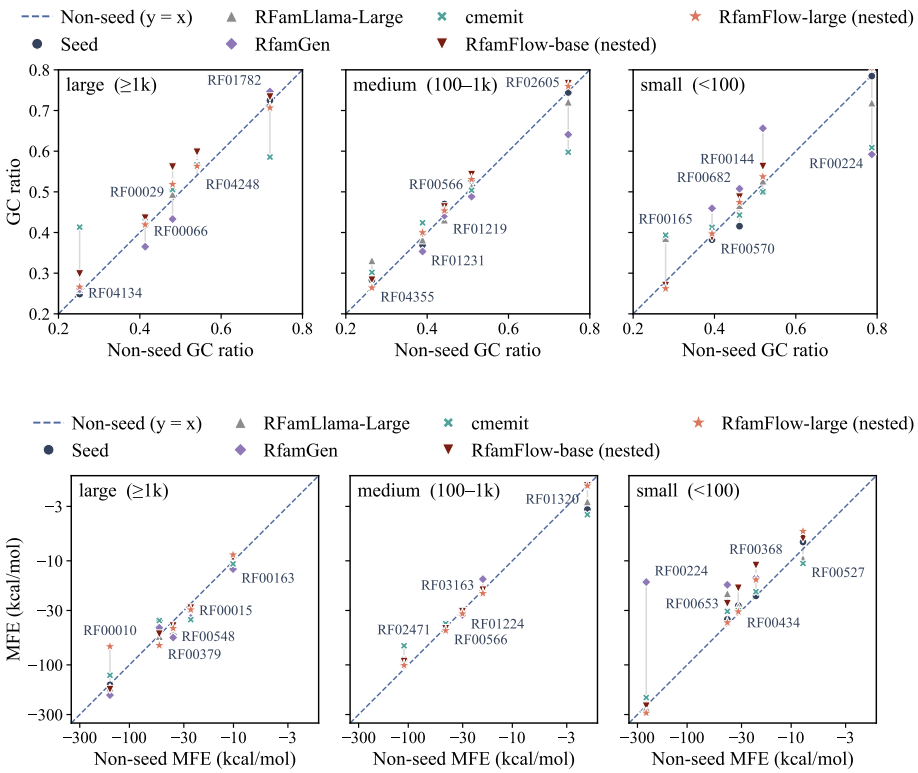


Figure 3: Per-family median GC content (top) and MFE (bottom) of each generator against the natural non-seed median, on five representative families per cohort chosen to span the natural range. MFE axes are log-scaled.

on every cohort, whereas RfamGen drops to $r = 0.34$ on the small cohort and cmemit reaches only 0.87–0.91.

RNACG (Gao & Lu, 2024) is trained only on family sequences with solved 3D structures, so we compare on the three families for which it releases checkpoints (Table 2). Even on its own benchmark families, and although it is given natural sequence lengths at sampling time, both RfamFlow models score markedly higher on average, most of all on RF00002 and RF00005. Details are in Appendix C.3.2.

Table 2: Comparison with RNACG on its three released families: average `cmsearch` bit score, with hit rate (%) in parentheses.

Group	RF00001	RF00002	RF00005
Seed	89.5 (100)	134.6 (100)	55.7 (100)
Non-seed	58.2 (99.9)	77.1 (100)	38.4 (99.9)
RNACG	62.1 (93)	65.6 (86)	13.2 (83)
RfamFlow-base (*nested*)	61.5 (98)	**125.0** (99)	34.7 (100)
RfamFlow-large (*nested*)	**62.3** (99)	111.9 (100)	**38.2** (100)

4.3 Ablation Study

To compare the two guidance schemes fairly, we tune each scheme independently. We sweep the scale γ for flat CFG and the pair $(w_{\text{sub}}, w_{\text{fam}})$ for nested CFG on each cohort (full sweeps in Appendix C.1), pick the best setting per variant, and report those in Table 3. As a reference, we also

include the no-guidance configurations ($\gamma = 1$ and $w_{\text{sub}} = w_{\text{fam}} = 1$) and the *diagonal* variant of equation 5.

Table 3: Ablation of the hierarchical embedding and guidance on RfamFlow-base, as macro-average `cmsearch` bit score.

Configuration	Large	Medium	Small
Flat, no guidance ($\gamma = 1$)	53.2	33.2	8.8
Flat, best γ	76.5	50.7	15.5
Nested, no guidance	57.6	38.6	15.2
Diagonal, best ω	78.0	53.6	22.9
Nested, best $(w_{\text{sub}}, w_{\text{fam}})$	**78.9**	**57.5**	**24.4**

Guidance is essential for both schemes. Without it, pure conditional generation is only comparable to the autoregressive RfamLlama-large baseline, and enabling guidance improves the macro-average bit score on large families for either variant. More importantly, nested CFG improves on flat CFG at their best settings on every cohort, with by far the largest gain on small families, where it yields a $1.6\times$ improvement. On the grid we sweep, the *diagonal* variant falls between the two on every cohort, so the shared embedding and the level-wise weights each contribute. This shows that the taxonomic hierarchy helps most when a family's own data is scarce. Finally, our best configuration is the same nested setting ($w_{\text{sub}} = 2.0$, $w_{\text{fam}} = 1.5$) on every cohort, showing that a strong sub-level prior with moderate family-level guidance generalizes across the resource spectrum.

4.4 Masked Sequence Reconstruction

Table 4: Masked reconstruction on the large cohort.

Model	15%	30%	50%
ERNIE-RNA	**78.3**	72.3	45.0
RiNALMo	69.1	65.3	55.1
RNA-FM	66.6	60.2	40.8
RNAErnie	45.8	42.5	35.8
SpliceBERT	34.2	33.5	31.6
RNA-MSM	26.1	26.0	25.8
RfamFlow-large (*nested*)	77.0	**75.3**	**73.6**
RfamFlow-base (*nested*)	69.6	68.4	67.5
RfamFlow-base (*flat*)	68.6	67.7	66.6

Although RfamFlow is trained for generation, its denoising objective also allows it to infill masked nucleotides directly, as an additional capability that needs no task-specific fine-tuning. To evaluate this capacity, on the large cohort we mask a fraction of the positions of each held-out sequence, run a single forward pass, take the most likely nucleotide at each masked position, and report the per-family averaged recovery rate at mask rates of 15%, 30%, and 50%. We compare against six discriminative RNA language models trained with masked language modeling: ERNIE-RNA (Yin et al., 2024), RiNALMo (Penić et al., 2024), RNA-FM (Chen et al., 2022), RNAErnie (Wang et al., 2024a), SpliceBERT (Chen et al., 2024), and RNA-MSM (Zhang et al., 2024).

Although our large model, at 58.5M parameters, is far smaller than these RNA foundation models (e.g. 650M for RiNALMo), RfamFlow-large matches or outperforms all of them at every mask rate in Table 4, except a narrow gap against ERNIE-RNA at 15%. ERNIE-RNA is structure-enhanced and injects base-pairing priors, which help most when few positions are missing and the surrounding context is rich. As more of the sequence is hidden, every discriminative model degrades sharply, whereas RfamFlow-large remains stable and is the most accurate method at both 30% and 50%. The 10M-parameter base model already matches most baselines, and the large model overtakes them. We report the plain (unguided) single forward pass, because reconstruction is a one-step point prediction rather than iterative sampling, so guidance has little effect here.

Table 5: Zero-shot fitness prediction: Spearman ρ on the three DMS assays. WT: WT-marginal; Mut: mutant-marginal; LL: family-tagged log-likelihood; ELBO: evidence lower bound of the per-family model. Best RfamFlow variant per column underlined, best overall in bold.

Model	Score	tRNA	glmS	CPEB3
RfamFlow-base (*flat*)	WT	<u>0.61</u>	0.06	<u>0.45</u>
	Mut	0.56	0.04	0.12
RfamFlow-base (*nested*)	WT	0.60	<u>0.24</u>	0.42
	Mut	0.57	-0.04	0.12
RfamFlow-large (*nested*)	WT	0.61	0.21	0.44
	Mut	0.59	0.23	0.28
RFamLlama-Large	LL	0.57	**0.60**	0.38
RfamGen	ELBO	0.52	0.53	0.46
ERNIE-RNA	WT	0.63	0.07	**0.51**
	Mut	0.61	-0.08	0.36
RiNALMo	WT	**0.63**	-0.05	0.39
	Mut	0.60	-0.19	0.03
RNA-FM	WT	0.63	0.05	0.44
	Mut	0.60	-0.02	0.30

4.5 Zero-Shot Fitness Prediction

As fitness prediction is central to RNA engineering, we evaluate whether RfamFlow can rank sequence variants zero-shot, without ever seeing a fitness label. Following the marginal-likelihood scoring introduced for protein language models (Meier et al., 2021), we score each variant in two ways: a *WT-marginal* score, which reads off the model's log-likelihood of the mutated positions relative to the wild type, and a *mutant-marginal* score, which masks and re-scores those positions. We correlate each score with the measured fitness by Spearman's ρ and report both schemes (Table 5). Full details of the assays and the two scoring schemes are given in Appendix C.5.

RfamFlow reaches performance comparable to that of the much larger foundation models across the three assays, without any task-specific training. On the data-rich tRNA assay every sequence model lands within a few points of the same value and RfamFlow sits among them; on CPEB3 the methods spread further apart and RfamFlow again ranks with the leaders, behind only the structure-enhanced ERNIE-RNA and the family-specific RfamGen. The glmS assay separates the two kinds of baseline: all three foundation models collapse on it while RfamFlow stays clearly positive, and the two generative baselines lead. As with reconstruction, scoring is a single forward pass rather than an iterative sampling trajectory, so guidance barely moves the ranking and we report the plain conditional scores without CFG.

5 Conclusion

We presented RfamFlow, a conditional discrete flow-matching model that generates RNA sequences for any Rfam family from its tag alone, without structural input. We introduced nested CFG, a multi-level guidance scheme that leverages the Rfam hierarchy to steer generation from coarse taxonomic context to family-specific sequence patterns. Across our experiments, nested CFG closes a key gap in low-resource RNA generation. When family-specific data are scarce, it draws on broader taxonomic structure to provide a useful prior and markedly improve generation quality. Our model also performs masked reconstruction and zero-shot fitness prediction with no task-specific training, matching much larger RNA foundation models.

However, RfamFlow depends on Rfam's curated taxonomy. Nested guidance borrows a prior from a family's coarser levels, so it applies to RNA families that a curated hierarchy already organizes, and its generalization to new RNA types outside Rfam is limited. Our evaluation of the generated sequences also remains computational, resting on `cmsearch` homology and folding statistics rather than measured function, and we see coupling RfamFlow with wet-lab screening as the most promising route to turning generated candidates into validated functional RNAs.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the tasks with required disclosure. Specifically, we have not used generative AI tools to help develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat datasets, or interpret results. The remaining tasks with required disclosure, namely generating synthetic data sets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, and supporting qualitative and thematic data analysis, are not applicable to this work. Additionally, we used generative AI tools for sourcing and searching for information and identifying relevant literature, for editing software code (limited to troubleshooting and suggesting fixes for bugs in code written by the authors), and for editing the paper to improve readability (grammar and clarity of text written by the authors). We have reviewed all AI-assisted work, and all authors took part in this review. For literature, AI tools were used only to surface candidate papers; the authors read every suggested paper in its original source, confirmed that it exists and that our description of it is accurate, and decided independently whether to cite it. For example, AI-assisted search pointed us to the arXiv preprint of RNACG (Gao & Lu, 2024), which we had not known of before; we read the paper and evaluated it as a baseline using the authors' released code. For code, every AI-suggested fix was checked with unit tests and by comparing outputs before and after the change before being adopted, and all reported results were produced by the authors' own pipeline. For writing, the authors compared each edited passage against the original text, confirming that no number, claim, or conclusion was changed. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Neil Tenenholtz, Robert Strome, Alan M. Moses, Alex X. Lu, Nicolò Fusi, Ava P. Amini, and Kevin K. Yang. Protein generation with evolutionary diffusion: Sequence is all you need. *bioRxiv*, 2023. doi: 10.1101/2023.09.11.556673.

Johan O L Andreasson, Andrew Savinov, Steven M Block, and William J Greenleaf. Comprehensive sequence-to-function mapping of cofactor-dependent RNA catalysis in the glmS ribozyme. *Nature Communications*, 11(1):1663, 2020.

Rohit Arora, Murphy Angelo, Christian Andrew Choe, Courtney A Shearer, Aaron W Kollasch, Fiona Qu, Ruben Weitzman, Artem Gazizov, Sarah Gurev, Erik Xie, Debora S Marks, and Pascal Notin. RNAGym: Large-scale benchmarks for RNA fitness and structure prediction. *bioRxiv*, pp. 2025–06, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Stephen K Burley, Helen M Berman, Gerard J Kleywegt, John L Markley, Haruki Nakamura, and Sameer Velankar. Protein Data Bank (PDB): the single global macromolecular structure archive. *Protein crystallography: methods and protocols*, pp. 627–641, 2017.

Francesco Calvanese, Camille N Lambert, Philippe Nghe, Francesco Zamponi, and Martin Weigt. Towards parsimonious generative modeling of RNA families. *Nucleic Acids Research*, 52(10): 5465–5477, 2024.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi S Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning (ICML)*, 2024.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11315–11325, 2022.

Dian Chen and Yu Liu. Riboswitch dynamics and their expanding biotechnological applications. *Health and Metabolism*, 2(4), 2025.

Jiayang Chen, Zhihang Hu, Siqi Sun, Qingxiong Tan, Yixuan Wang, Qinze Yu, Licheng Zong, Liang Hong, Jin Xiao, Tao Shen, et al. Interpretable RNA foundation model from unannotated data for highly accurate RNA structure and function predictions. *arXiv preprint arXiv:2204.00300*, 2022.

Ken Chen, Yue Zhou, Maolin Ding, Yu Wang, Zhixiang Ren, and Yuedong Yang. Self-supervised learning on millions of primary RNA sequences from 72 vertebrates improves sequence-based RNA splicing prediction. *Briefings in Bioinformatics*, 25(3):bbae163, 2024.

Shuying Chen, Qian Chen, Xinru You, Zhuoming Zhou, Na Kong, Fabrisia Ambrosio, Yihai Cao, Reza Abdi, and Wei Tao. Using RNA therapeutics to promote healthy aging. *Nature Aging*, 5(6): 968–983, 2025.

Jennifer A Doudna and Thomas R Cech. The chemical repertoire of natural ribozymes. *Nature*, 418 (6894):222–228, 2002.

Alexis A Flores-García, Aura Hernandez-Rodriguez, E Isabel López-Brito, Dafne F Nava-Domínguez, Sebastian P Pasapera, R Franchesca Pazos-Murillo, Tonatzin Pineda-Diaz, and Alberto Vázquez-Salazar. Brave new RNA world (s): from prebiotic chemistry to gene regulation and RNA technology. *Frontiers in Genetics*, 17:1813517, 2026.

Letian Gao and Zhi John Lu. RNACG: A universal RNA sequence conditional generation model based on flow-matching. *arXiv preprint arXiv:2407.19838*, 2024.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Sam Griffiths-Jones, Alex Bateman, Mhairi Marshall, Ajay Khanna, and Sean R Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 2003.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Andrew Kubaney, Andrew Favor, Lilian McHugh, Raktim Mitra, Robert Pecoraro, Justas Dauparas, Cameron Glasscock, and David Baker. RNA sequence design and protein–DNA specificity prediction with NA-MPNN. *bioRxiv*, pp. 2025–10, 2025.

Chuan Li, Wenfeng Qian, Calum J Maclean, and Jianzhi Zhang. The fitness landscape of a trna gene. *Science*, 352(6287):837–840, 2016.

Langzhang Liang, Ming Yang, Yi Feng, Junfan Li, Shirui Pan, Yinghui Xu, Tianlei Ying, Yizhen Zheng, and Zenglin Xu. LineageFlow: Flow matching for high-fidelity family-aware protein sequence generation. *arXiv preprint arXiv:2605.22252*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*, 2024.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41 (8):1099–1106, 2023.

Maumita Mandal and Ronald R Breaker. Gene regulation by riboswitches. *Nature Reviews Molecular Cell Biology*, 5(6):451–463, 2004.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:29287–29303, 2021.

Alex Morehead, Lazar Atanackovic, Akshata Hegde, Yanli Wang, Frimpong Boadu, Joel Selvaraj, Alexander Tong, Aditi Krishnapriyan, and Jianlin Cheng. Flow matching for generative modelling in bioinformatics and computational biology. *Nature Machine Intelligence*, pp. 1–18, 2026.

Geraldene Munsamy, Sebastian Lindner, Philipp Lorenz, and Noelia Ferruz. ZymCTRL: a conditional language model for the controllable generation of artificial enzymes. In *NeurIPS machine learning in structural biology workshop*. NeurIPS, 2022.

Eric P Nawrocki and Sean R Eddy. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935, 2013.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *International Conference on Learning Representations (ICLR)*, 2025.

Stephen Obonyo, Nicolas Jouandeau, and Dickson Owuor. Self-playing RNA inverse folding. *SN Computer Science*, 5(4):403, 2024.

Nancy Ontiveros-Palacios, Emma Cooke, Eric P Nawrocki, Sandra Triebel, Manja Marz, Elena Rivas, Sam Griffiths-Jones, Anton I Petrov, Alex Bateman, and Blake Sweeney. Rfam 15: RNA families database in 2025. *Nucleic Acids Research*, 53(D1):D258–D267, 2025.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations (ICLR)*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Rafael Josip Penić, Tin Vlašić, Roland G Huber, Yue Wan, and Mile Šikić. RiNALMo: General-purpose RNA language models can generalize well on structure prediction tasks. *arXiv preprint arXiv:2403.00043*, 2024.

Álvaro Rubio-Largo, Nuria Lozano-García, José M Granado-Criado, and Miguel A Vega-Rodríguez. Solving the RNA inverse folding problem through target structure decomposition and multiobjective evolutionary computation. *Applied Soft Computing*, 147:110779, 2023.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shunsuke Sumi, Michiaki Hamada, and Hirohide Saito. Deep generative design of RNA family sequences. *Nature Methods*, 21(3):435–443, 2024.

Jinyuan Sun, Han Li, and Yifan Deng. RFamLlama: An efficient conditional language model for RNA sequence generation across diverse structural families. In *ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery*, 2024.

The RNAcentral Consortium. RNAcentral in 2026: genes and literature integration. *Nucleic Acids Research*, 54(D1):D303–D313, 2026.

Ignacio Tinoco Jr and Carlos Bustamante. How RNA folds. *Journal of Molecular Biology*, 293(2): 271–281, 1999.

Shengkun Tong and Yaojun Tong. Engineering cis- and trans-acting RNA regulators for next-generation prokaryotic synthetic biology. *Biotechnology Advances*, pp. 108882, 2026.

Ning Wang, Jiang Bian, Yuchen Li, Xuhong Li, Shahid Mumtaz, Linghe Kong, and Haoyi Xiong. Multi-purpose RNA language modelling with motif-aware pretraining and type-guided fine-tuning. *Nature Machine Intelligence*, 6(5):548–557, 2024a.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. In *International Conference on Machine Learning (ICML)*, 2024b.

Weijie Yin, Zhaoyu Zhang, Liang He, Rui Jiang, Shuo Zhang, Gan Liu, Xuegong Zhang, Tao Qin, and Zhen Xie. ERNIE-RNA: An RNA language model with structure-enhanced representations. *bioRxiv*, pp. 2024–03, 2024.

Yikun Zhang, Mei Lang, Jiuhong Jiang, Zhiqiang Gao, Fan Xu, Thomas Litfin, Ke Chen, Jaswinder Singh, Xiansong Huang, Guoli Song, et al. Multiple sequence alignment-based RNA language model and its application to structural inference. *Nucleic Acids Research*, 52(1):e3, 2024.

Zhe Zhang, Peng Xiong, Tongchuan Zhang, Junfeng Wang, Jian Zhan, and Yaoqi Zhou. Accurate inference of the full base-pairing structure of RNA by deep mutational scanning and covariation-induced deviation of activity. *Nucleic Acids Research*, 48(3):1451–1465, 2020. doi: 10.1093/nar/gkz1192.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. In *International Conference on Learning Representations (ICLR)*, 2025.

Yuanzhe Zhou and Shi-Jie Chen. Harnessing computational approaches for RNA-targeted drug discovery. *RNA Nanomedicine*, 1(1):1–15, 2024. doi: 10.59566/isrnn.2024.0101001.

Tao Zhu, Weiwei Jiang, Yingyu Wu, Rong Fang, Fei Deng, and Danting Yang. Advances in CRISPR/Cas13a-based biosensors for non-coding RNA detection. *Talanta*, 294:128223, 2025.

A Data Curation and Cohorts

We take Rfam 15.1 (Ontiveros-Palacios et al., 2025) as our data source, which spans $4{,}227$ RNA families and roughly 10M sequences. For preprocessing, we capitalize all sequences, convert `T` to `U`, and discard any sequence containing characters outside $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. Within each family we then reduce redundancy with MMseqs2 `easy-cluster` (`--min-seq-id 0.9 -c 0.8 --cov-mode 0`), keeping one representative per cluster. Each Rfam family provides two sequence sets, a small, manually curated *seed* set and a larger *non-seed* set. We train and validate on the non-seed sequences and hold out the seed sequences as a test set.

After preprocessing and deduplication, we categorize the remaining families into three groups according to their scale, measured by the number of non-seed sequences a family retains after deduplication (Figure 4). We define *large* families as those with at least $1{,}000$ sequences, *medium* families as those with 100–999, and *small* families as those with 10–99. We set the lower limit of the small group at 10, as a family with only a handful of sequences yields a signal too weak to be meaningful. From each group we then randomly sample 100 families for the downstream tasks, and keep this selection fixed across all experiments. The complete large, medium, and small cohorts, with each family's Rfam accession, taxonomy, and RNA type, are listed in Tables 15, 16, and 17.

B Model Architecture and Implementation

B.1 Tokenization and Conditioning

An RNA sequence is tokenized at single-nucleotide resolution over the four bases $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. We add four special tokens, namely padding (`<pad>`), the start- and end-of-sequence markers

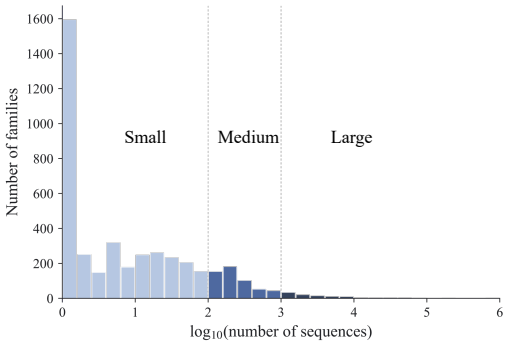


Figure 4: Distribution of Rfam family sizes. Each family is binned by the base-10 logarithm of its number of deduplicated non-seed sequences (x-axis); the y-axis counts the families falling in each bin.

```
Rfam
|-- Gene                <- c_top
|   |-- tRNA            <- c_sub
|   |   `-- RF00005     <- c_fam
|   |-- rRNA
|   |   `-- RF00001     (5S rRNA)
|   `-- ribozyme
|       `-- RF03160     (twister)
|-- Cis-reg
|   `-- riboswitch
|       |-- RF00174     (Cobalamin)
|       `-- RF00234     (glmS)
`-- Intron
    `-- RF00028     (group I)
```

Figure 5: The Rfam taxonomy rendered as a tree. Each family (a leaf, given by its RF accession) sits under a subtype (c_{sub}) and one of three coarse types (c_{top}), namely Gene, Cis-reg, or Intron. The subtype is the family's Rfam type string (e.g. "Gene; tRNA"), shown here by its leaf label; the three levels form the hierarchical label $c = (c_{\text{top}}, c_{\text{sub}}, c_{\text{fam}})$ used by nested guidance.

(<sos>, <eos>), and a mask token (<mask>) that serves as the absorbing state of the discrete flow, so the vocabulary comprises 8 tokens. Each sequence is wrapped with <sos>/<eos> and padded to a common length $L = 512$, and padded positions are not scored by the loss.

We encode each family as a three-level label $c = (c_{\text{top}}, c_{\text{sub}}, c_{\text{fam}})$ following the Rfam taxonomy, namely a coarse RNA type, a finer subtype, and the family (its Rfam accession) itself. Figure 5 illustrates this hierarchy with representative families. Each level is encoded by a learned look-up embedding indexed by its id. *Nested* guidance exploits this structure, letting a data-poor family borrow a prior from its coarser levels.

For the *nested* model we keep one embedding table per level and sum their outputs into a single class signal,

$$e_{\text{cls}} = e_{\text{top}}(c_{\text{top}}) + e_{\text{sub}}(c_{\text{sub}}) + e_{\text{fam}}(c_{\text{fam}}). \tag{6}$$

The *flat* baseline keeps a single table over family ids, so its class signal is simply $e_{\text{cls}} = e_{\text{fam}}(c_{\text{fam}})$. Each level embedding, and hence e_{cls}, has width $d/2$, where d is the model width. The diffusion time t is encoded by a sinusoidal timestep embedding of width $d/2$, and the two are concatenated into the d-dimensional conditioning vector that drives the AdaLN modulation in every DiT block.

B.2 DiT Block with AdaLN

The denoiser is a DiT-style Transformer (Peebles & Xie, 2023). The input tokens are mapped by a learned token embedding, to which a fixed sinusoidal positional encoding is added. The result passes through N_L identical AdaLN blocks and a final head, all conditioned on the vector s formed in Section B.1 (the class signal concatenated with the timestep embedding).

Conditioning enters through adaptive layer normalization. For a hidden state h and conditioning s,

$$\text{AdaLN}(h; s) = \text{LayerNorm}(h) \odot \big(1 + \gamma(s)\big) + \beta(s), \tag{7}$$

where $\gamma(s)$ and $\beta(s)$ are per-block linear projections of s. Each block applies AdaLN before both the multi-head self-attention (MHSA) and the feed-forward network (FFN), and gates each residual branch, after dropout, by a further learned projection of s,

$$h \leftarrow h + g_1(s) \odot \text{MHSA}\big(\text{AdaLN}_1(h; s)\big), \tag{8}$$

$$h \leftarrow h + g_2(s) \odot \text{FFN}\big(\text{AdaLN}_2(h; s)\big), \tag{9}$$

where the gates $g_1(s), g_2(s)$ are element-wise vectors broadcast over sequence positions. Unlike AdaLN-Zero, we do not zero-initialize these residual gates. Because the base model is shallow,

Table 6: Architecture configuration of the base and large RfamFlow models.

Hyperparameter	RfamFlow-base	RfamFlow-large
Layers N_L	8	12
Model width d	256	512
Attention heads	8	8
Head dimension	32	64
FFN hidden width	1024	2048
Dropout	0.1	0.1
Vocabulary size	8	8
Max sequence length L	512	512
Parameters	10.2M	58.5M

random initialization trains readily and converges faster in our setting. The final head applies one more AdaLN and a linear projection to per-position logits over the vocabulary, with this output projection zero-initialized so generation starts from a neutral prediction. Attention uses scaled dot-product flash attention. The base and large configurations are given in Table 6.

B.3 Training Hyperparameters

Table 7: Training configuration for the base and large models.

Setting	RfamFlow-base	RfamFlow-large
Optimizer	Adam	
Learning rate	3×10^{-4}	1×10^{-4}
Warmup steps	2,000	4,000
LR schedule	linear warmup, cosine decay	
Gradient-norm clip	1.0	
Epochs	50	
Global batch size	128	

Both models are trained with the same objective and schedule (Table 7). The large model uses a lower learning rate and a longer warmup to stabilize its early training. For classifier-free guidance we drop labels during training. The *flat* variant drops the family label with probability 0.15. The *nested* variant keeps the top level, drops family with probability 0.15, and further drops sub with probability 0.10.

B.4 Computing Infrastructure

All models are trained and evaluated on a single machine with two NVIDIA GeForce RTX 3080 Ti GPUs (12 GB each), running Ubuntu 22.04 with Linux kernel 5.15. The models are implemented in PyTorch 2.4.1 (CUDA 11.8, cuDNN 9.1) under Python 3.10, with NumPy 2.2 and SciPy 1.15. Evaluation uses Infernal 1.1.5 (`cmsearch`, `cmalign`), MMseqs2 18 for clustering, and ViennaRNA 2.7.2 (`RNAfold`) for secondary-structure folding.

B.5 Sampling

At inference we start from an all-<mask> sequence and decode with MaskGIT-style confidence-based iterative unmasking (Chang et al., 2022) (Algorithm 1), run for $N = 100$ steps with $t_{\max} = 0.999$, kept just below 1 because the discrete flow's rate diverges as $t \to 1$ (Gat et al., 2024). A cosine schedule commits few high-confidence positions early and many late, freezing them so that later steps condition on more correct context; annealed Gumbel noise ($\tau = 1$) on the selection score keeps samples diverse, and each committed position is filled by sampling from the model's predicted distribution (temperature $T = 1$). Guidance scales are set per cohort (Section C.1).

Algorithm 1 MaskGIT-style confidence-based iterative unmasking (Chang et al., 2022).

Require: label c, steps N, $t_{\max}$, guidance γ, Gumbel temp τ

$\mathbf{x} \leftarrow L$ copies of <mask>
$\mathcal{R} \leftarrow \varnothing$ and $r \leftarrow 0$
for $k = 1$ **to** N **do**
  $n \leftarrow \mathrm{round}\big(L\, t_{\max}(1 - \cos \frac{\pi k}{2N})\big) - r$
  if $n \le 0$ **then**
    continue
  end if
  $p \leftarrow \mathrm{softmax}\, \tilde{f}_\theta(\mathbf{x}, c, r/L)$
  $s_i \leftarrow \log\big(\max_j p_{i,j}\big) + \tau\,(1 - r/L)\, g_i, \quad g_i \sim \mathrm{Gumbel}(0, 1)$
  $s_i \leftarrow -\infty$ for every committed position $i \in \mathcal{R}$
  $\mathcal{R}' \leftarrow$ the n positions with the largest s_i
  sample $\mathbf{x}_i \sim p_i$ for $i \in \mathcal{R}'$ {commit temperature T=1}
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$ and $r \leftarrow r + n$
end for
return $\mathbf{x}$

B.6 Relation to Masked Diffusion Models

With an absorbing source and the linear schedule, our discrete flow is a masked diffusion model (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024). The path in the main text masks each position independently with probability $1 - t$, which is the absorbing forward process of these models with $\alpha_t = t$ (in our convention, $t = 1$ is data). The denoiser in both cases learns the clean-data conditional $p_\theta(x_1^i \mid x_t, c)$ at masked positions, and this conditional also determines the concrete score learned by absorbing SEDD (Lou et al., 2024; Ou et al., 2025). The three views thus share one model and differ in training weight and sampler.

Training objective. In our time convention the negative ELBO of masked diffusion (Sahoo et al., 2024; Shi et al., 2024) is

$$\mathcal{L}_{\mathrm{ELBO}}(\theta) = \mathbb{E}\left[\frac{1}{1-t} \sum_{i:\, x_t^i = \texttt{<mask>}} -\log p_\theta(x_1^i \mid x_t, c, t)\right]. \tag{10}$$

Our loss drops the $1/(1-t)$ weight, as in discrete flow-matching (Gat et al., 2024). Positions already unmasked in x_t carry their clean token, so the denoiser reproduces them exactly and they do not change the optimum at masked positions. The two objectives therefore have the same minimizer and differ only in how training effort is spread over t.

Sampling. The ancestral sampler of masked diffusion unmasks each masked position at random with the probability set by the schedule. Our MaskGIT-style sampler (Chang et al., 2022) instead chooses the positions by model confidence. This greedy ordering is not the sampler implied by the ELBO. It is widely used with masked diffusion models (Zheng et al., 2025; Nie et al., 2025) and gives the best samples in our setting (Table 10).

Guidance and biological sequences. Guidance for discrete diffusion and flow models has been developed in both classifier-based and classifier-free forms (Nisonoff et al., 2025; Schiff et al., 2025), and masked diffusion has been applied to protein sequences (Alamdari et al., 2023; Wang et al., 2024b). RfamFlow uses the standard classifier-free form and extends it to the nested label hierarchy of Rfam.

C Additional Results

C.1 Classifier-Free Guidance Sweeps

We sweep the classifier-free guidance strength for both the *flat* and *nested* variants on the base model, and select the setting that maximizes the macro-average `cmsearch` bit score on each cohort. The *flat* variant has a single scale γ (Table 8), whereas the *nested* variant has two level scales

$(w_{\text{sub}}, w_{\text{fam}})$ that we sweep over a grid (Table 9). At $\gamma = 1$ and $w_{\text{sub}} = w_{\text{fam}} = 1$ the guidance term vanishes and each variant reduces to plain conditional generation. We swept guidance only on the base model and transfer the resulting best nested setting, $(w_{\text{sub}}, w_{\text{fam}}) = (2.0, 1.5)$, to the large model. The diagonal $w_{\text{sub}} = w_{\text{fam}}$ of Table 9 is the *diagonal* variant of equation 5, since setting the two scales equal collapses the telescoping sum to one top-anchored step; the *diagonal* row of Table 3 is its best entry per cohort. Owing to computational cost we sweep the two level scales at 0.5 spacing, coarser than the 0.1 spacing used for the single scale of flat CFG, so the true optimum of the grid, and of its diagonal, may lie between the points we evaluate. Across the points we do evaluate the ordering is consistent: on every cohort the best nested setting improves on no guidance, on flat guidance, and on the *diagonal* variant. The last of the three is the informative comparison, because the *diagonal* variant already carries all three label levels, so the advantage of nested guidance does not reduce to taking those levels into account.

Table 8: *Flat*-CFG guidance-scale sweep on the base model.

γ	Large	Medium	Small
1.0	53.2	33.2	8.8
1.1	58.7	37.5	9.9
1.2	63.7	40.9	11.1
1.3	67.9	43.8	12.2
1.4	71.0	45.9	13.0
1.5	73.5	47.7	13.6
1.6	75.0	49.2	14.4
1.7	76.0	50.0	14.9
1.8	**76.5**	**50.7**	15.2
1.9	75.0	50.4	**15.5**
2.0	73.1	49.6	15.4

Table 9: *Nested*-CFG guidance-scale grid sweep on the base model.

w_{sub}	w_{fam}	Large	Medium	Small
1.0	1.0	57.6	38.6	15.2
1.0	1.5	70.4	53.1	21.9
1.0	2.0	67.3	55.9	22.8
1.5	1.0	63.8	39.4	15.8
1.5	1.5	76.7	53.6	22.7
1.5	2.0	78.3	57.4	24.0
2.0	1.0	63.9	37.9	15.1
2.0	1.5	**78.9**	**57.5**	**24.4**
2.0	2.0	78.0	53.2	22.9

C.2 Sampler Ablation

Our default sampler is the MaskGIT-style confidence-based unmasking of Section B (Algorithm 1). We compare it against two standard discrete-flow integrators that share the same trained model, the same absorbing all-<mask> prior, and the same 100 steps. The *Euler* solver integrates the mixture-path velocity in continuous state, predicting the clean token distribution at each step and resampling every position from the current state moved along that velocity (Gat et al., 2024). The *jump* solver uses the continuous-time Markov chain formulation, where each position jumps to a freshly sampled clean token with probability $1 - e^{-h\,\rho}$ at rate $\rho = 1/(1-t)$ (Campbell et al., 2024). Both resample every position at every step and never commit, whereas the mask solver freezes each position once it is unmasked. We run all three on RfamFlow-base (*nested*) at its chosen guidance and score the generated sequences by `cmsearch` bit, macro averaged over the families in each cohort.

The mask solver is best on every cohort by a wide margin, with the gap largest on the large cohort (78.9 against 66.5 for Euler and 44.2 for the jump solver). Committing high-confidence positions and conditioning the remaining steps on them protects structure that is already correct, while the

Table 10: Sampler ablation on RfamFlow-base (*nested*), macro-averaged `cmsearch` bit score per cohort. All three use 100 steps and the same absorbing prior.

Sampler	Large	Medium	Small
Euler	66.5	47.1	20.0
Jump (CTMC)	44.2	33.9	14.8
Mask (MaskGIT)	**78.9**	**57.5**	**24.4**

Euler and jump solvers resample every position at every step and can overwrite it, the jump solver most of all because its transitions are the most stochastic. This is why we adopt the MaskGIT-style sampler as our default.

C.3 *De Novo* Generation

For every family in the large, medium, and small cohorts we generate 100 sequences with both RfamFlow-base and RfamFlow-large, each decoded from an all-<mask> sequence by the mask solver (Algorithm 1).

C.3.1 Per-Family Sequence Statistics

For every sequence we compute three descriptors. The GC content is the fraction of `G` and `C` among the four bases. The minimum free energy (MFE) is the free energy, in kcal/mol, of the MFE secondary structure predicted for the sequence by ViennaRNA RNAfold (Lorenz et al., 2011). The edit distance is the minimum normalized Levenshtein distance from a sequence to the family's held-out seed set. We compute it for both the generated sequences and the natural non-seed sequences, using the seed as a common reference. When the generated sequences are as close to the seed as the non-seed sequences are, they occupy the same region of sequence space as real family members. For each family we summarize these over the generated `cmsearch` hits and compare against the natural family sequences, treating them as qualitative fidelity checks rather than a ranking. GC content and MFE are shown per cohort in Figures 6–11, and the macro-average edit distance is reported in Table 11.

Table 11: Per-family macro-average normalized edit distance (Levenshtein) to the held-out seed set.

Method	Large	Medium	Small
Natural sequences			
Non-seed	0.245	0.236	0.206
Family-specific			
RfamGen	0.214	0.221	0.379
cmemit	0.396	0.373	0.372
Family-conditioned			
RFamLlama-Large	0.319	0.368	0.435
RfamFlow-base (*flat*)	0.244	0.278	0.395
RfamFlow-base (*nested*)	0.241	0.260	0.345
RfamFlow-large (*nested*)	0.244	0.220	0.260

Table 12 quantifies this agreement by correlating each generator's per-family median against the natural one, so a high value means the generator tracks each family's own profile rather than a global average. RfamFlow stays high on every cohort, whereas RfamGen collapses on the small cohort (0.34 for GC, 0.39 for MFE), matching its low bit scores there. cmemit, which samples each family's covariance model directly, matches or exceeds RfamFlow on MFE, as expected from a model that encodes the family's consensus secondary structure, but trails it on GC, where its samples drift toward the middle of the range.

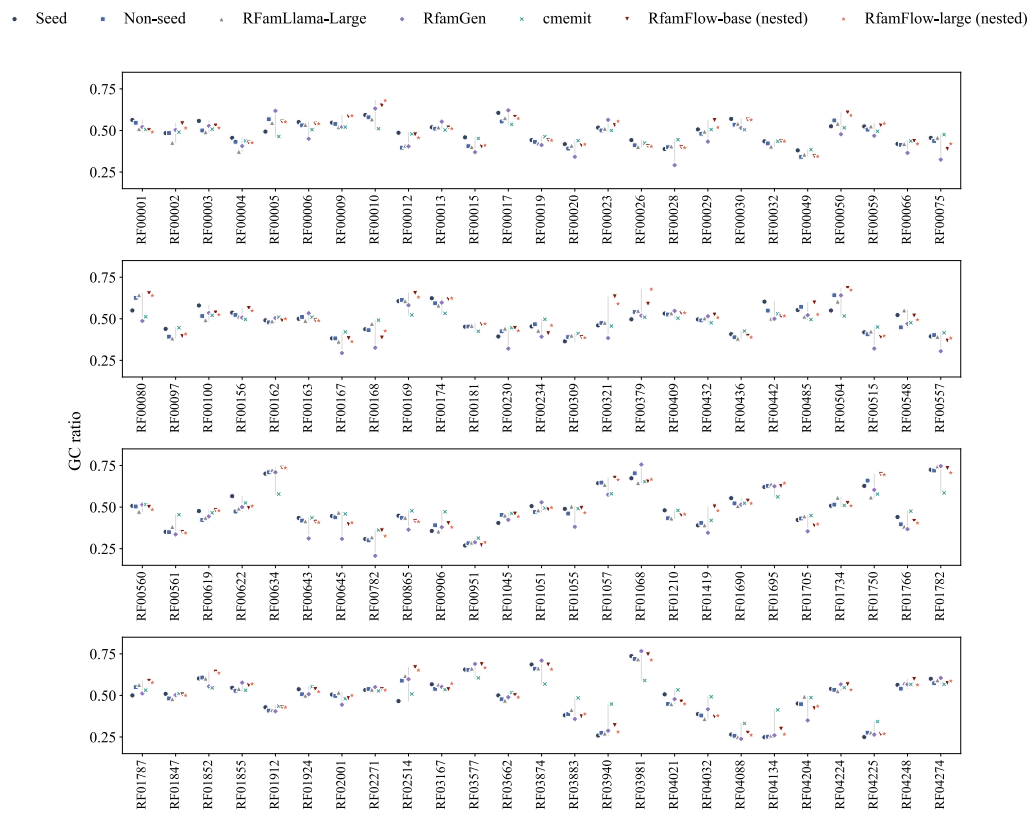


Figure 6: Per-family GC content of generated versus natural sequences on the large cohort.

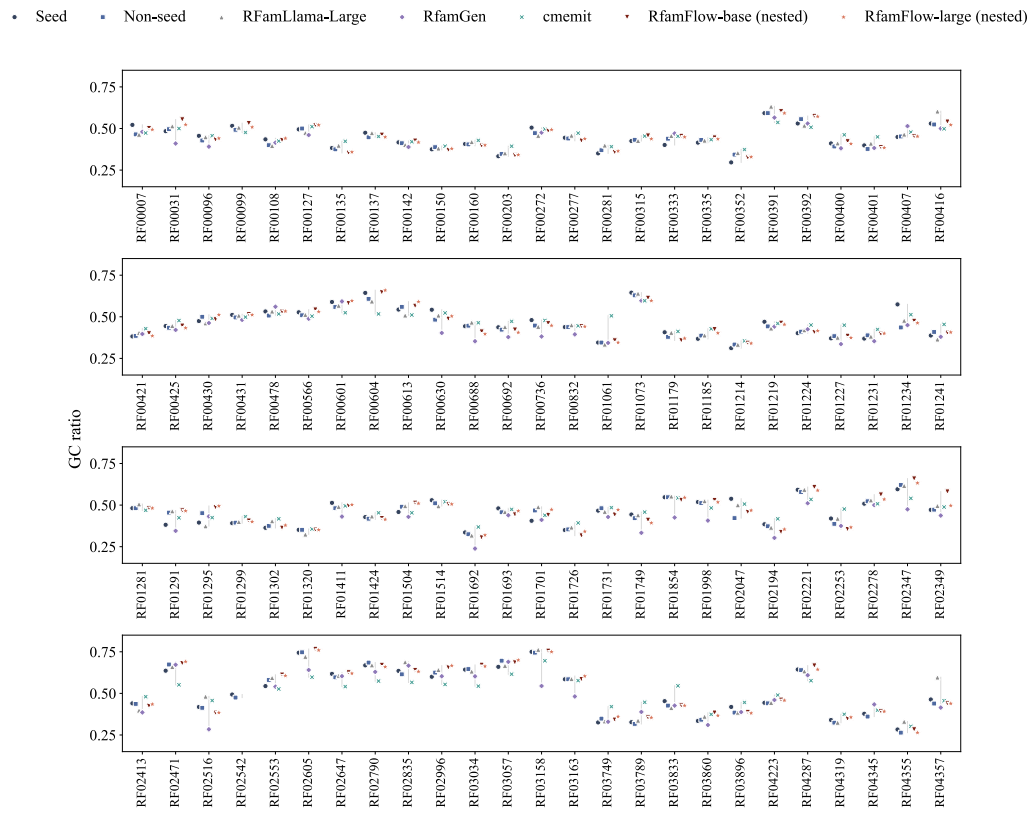


Figure 7: Per-family GC content of generated versus natural sequences on the medium cohort.

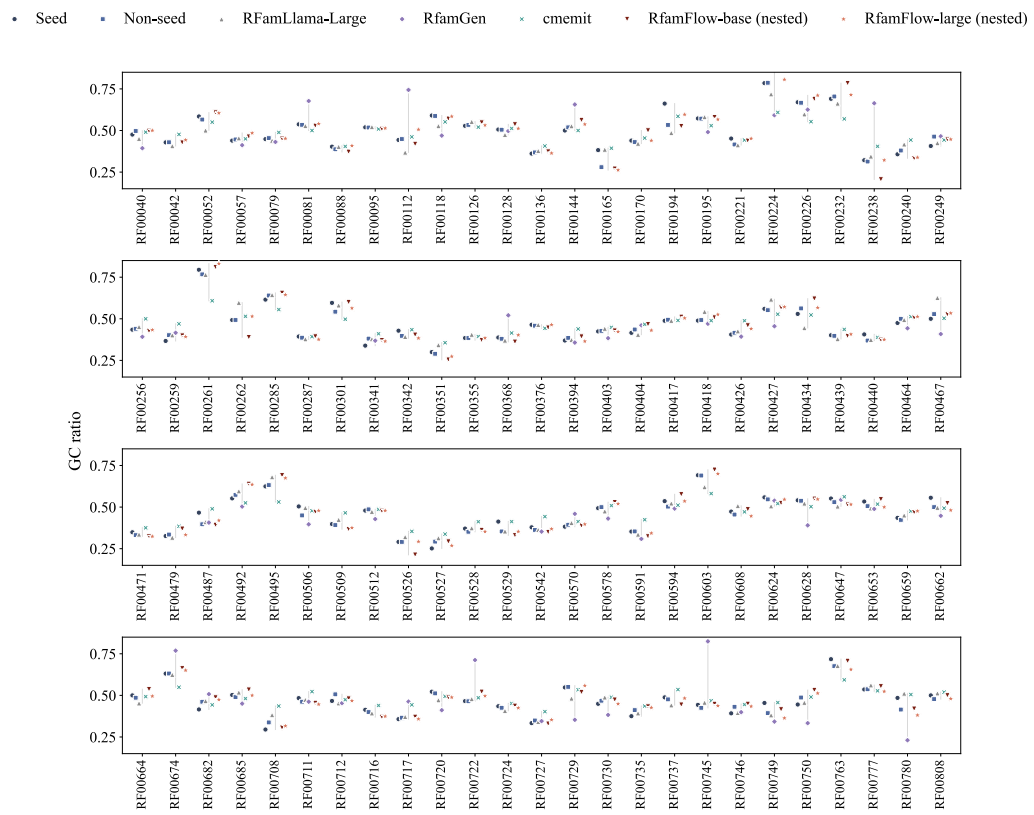


Figure 8: Per-family GC content of generated versus natural sequences on the small cohort.

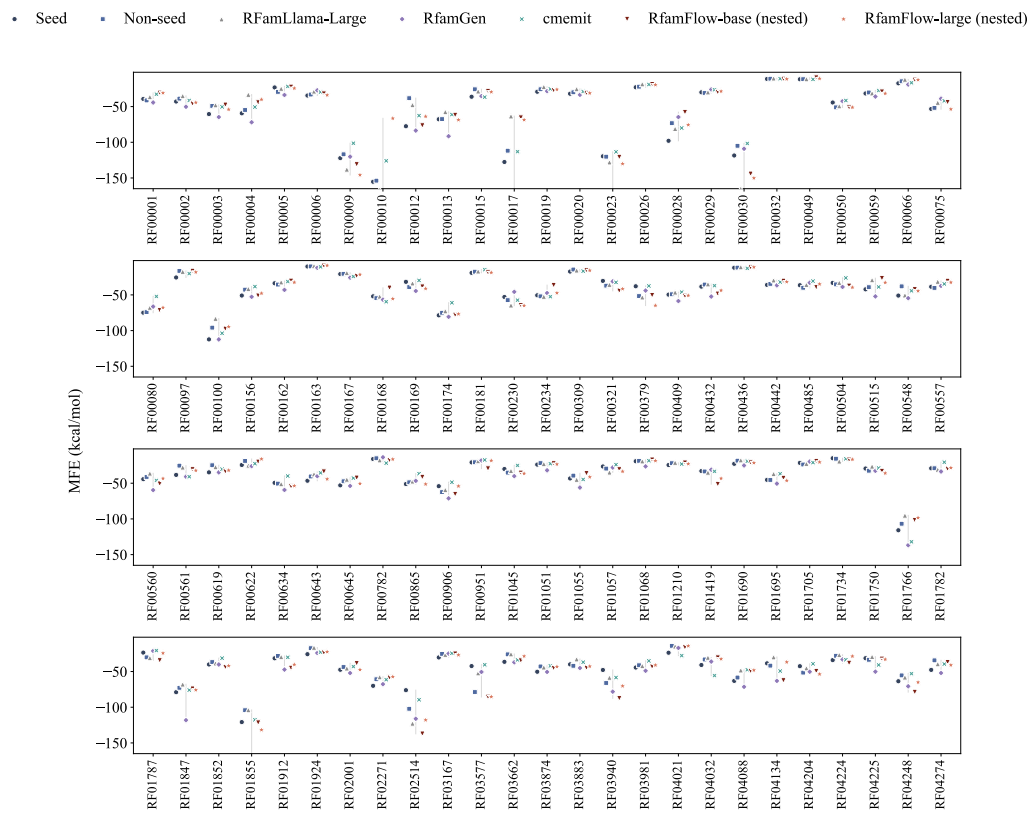


Figure 9: Per-family MFE of generated versus natural sequences on the large cohort.

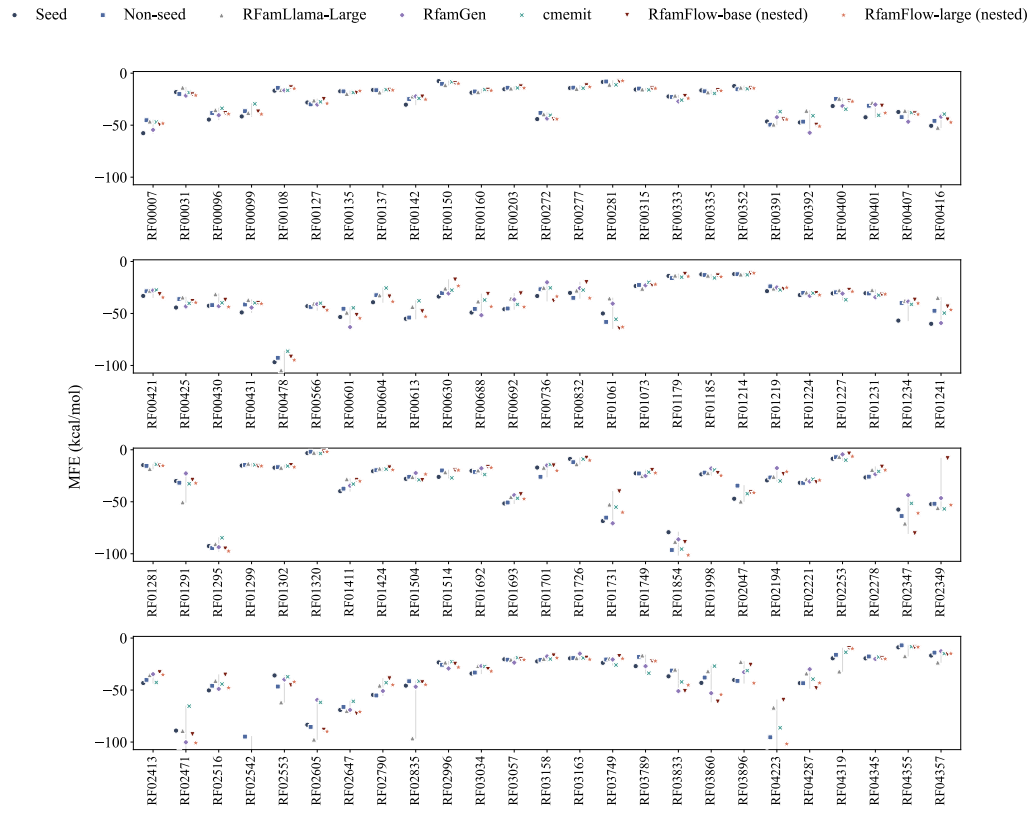


Figure 10: Per-family MFE of generated versus natural sequences on the medium cohort.

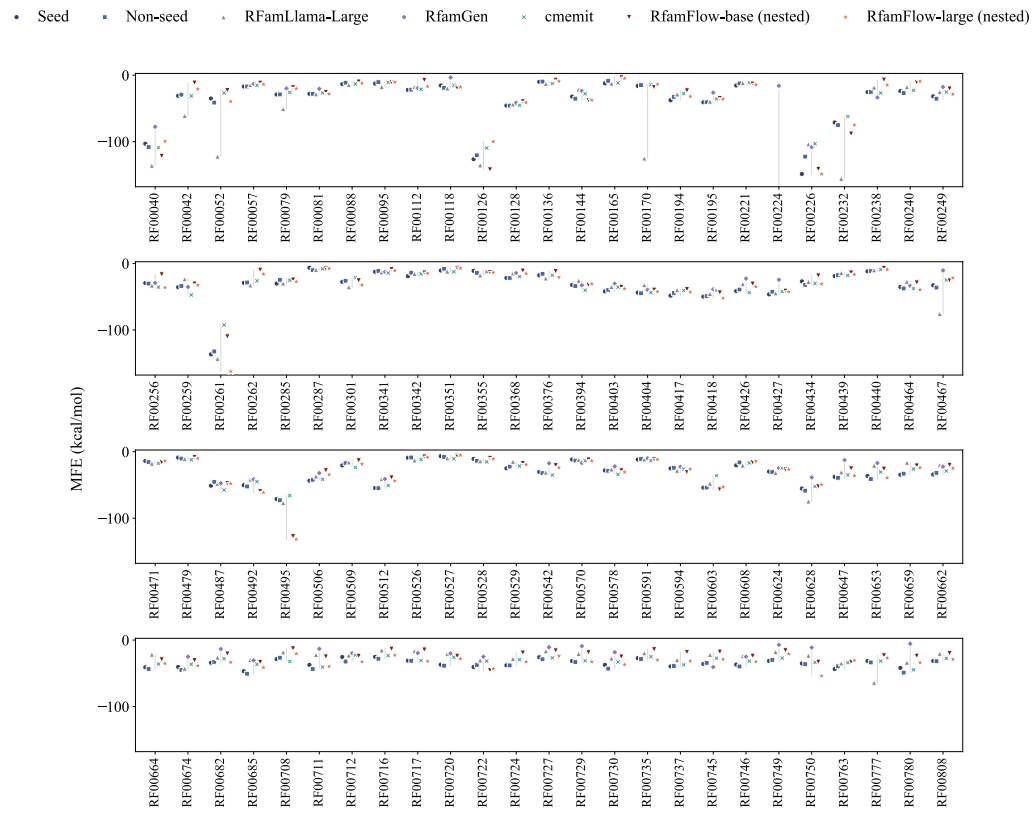


Figure 11: Per-family MFE of generated versus natural sequences on the small cohort.

Table 12: Pearson correlation between each generator's per-family median GC content or MFE and the natural non-seed median, using `cmsearch` hits only. RfamGen is evaluated only on families with at least one hit (91, 73, and 55 of 100).

Model	Large	Medium	Small
GC content			
RfamFlow-base (*flat*)	0.949	0.971	0.955
RfamFlow-base (*nested*)	0.949	0.975	0.970
RfamFlow-large (*nested*)	**0.972**	**0.984**	**0.983**
RFamLlama-Large	0.971	0.956	0.925
RfamGen	0.909	0.872	0.336
cmemit	0.873	0.896	0.911
MFE			
RfamFlow-base (*flat*)	0.939	0.933	0.903
RfamFlow-base (*nested*)	0.935	0.912	0.953
RfamFlow-large (*nested*)	0.894	**0.982**	0.968
RFamLlama-Large	0.940	0.902	0.856
RfamGen	0.927	0.922	0.390
cmemit	**0.941**	0.946	**0.979**

C.3.2 Baseline Reproduction

RfamLlama. We use the largest released model, `jinyuan22/RFamLlama-large`, with its HuggingFace checkpoint. RfamLlama is autoregressive and conditioned on the Rfam accession, so we prompt it with each family tag and sample 100 sequences per family, then uppercase them and map `T` to `U` to match our alphabet before scoring with `cmsearch`. For fitness prediction we prompt the same model with the assay's family tag and score each variant by its autoregressive log-likelihood, summed over the nucleotide tokens.

RfamGen. We follow the authors' released pipeline, fitting a separate CM-guided VAE per family. For each family we (i) clean its Rfam sequences and convert `T` to `U`, (ii) align them to the family covariance model and convert the alignment traceback to a one-hot encoding, (iii) split the sequences $70/15/15$ into train, validation, and test (random seed 42) and reweight them at a 0.1 identity threshold, (iv) train the VAE, and (v) draw new sequences by sampling from the standard-normal latent and decoding through the CM. The VAE has latent dimension 16 and hidden width 128; we train it with learning rate 10^{-3}, batch size 8, and gradient clipping at 20, for up to 10 epochs with early stopping (patience 3) and a KL weight $\beta = 10^{-3}$ under cyclic annealing, using the bug-fixed ELBO branch. Because each family needs its own model, families whose covariance-model representation is shorter than the convolutional decoder's kernel, or that have too few sequences to align and train, produce no output; we score these as zero so all methods are compared on the same family set. For fitness prediction we reuse these per-family models and score each variant by its ELBO, averaged over 25 latent draws and negated so that a higher score means higher predicted fitness. Scoring runs the same preprocessing as training, so every variant is aligned to the family covariance model before it is encoded.

cmemit. We sample with Infernal 1.1.5 cmemit (Nawrocki & Eddy, 2013) from each family's covariance model, fetched from the Rfam covariance-model library with `cmfetch`; this is the same model that `cmsearch` later scores against. For each family we draw 100 unaligned sequences with the default settings (global emission, no `--exp` scaling) and a fixed random seed of 42, and score them with `cmsearch` exactly as for the other methods.

RNACG. We use the authors' released code and its per-family checkpoints for RF00001, RF00002, and RF00005, without retraining; no checkpoint is released for any other family. Sampling calls the released sampler unchanged: a uniform Dirichlet prior, 10 integration steps (the released default), and a final argmax, without the optional secondary-structure correction (the "w/o ss" variant, which scores higher in the original paper). RNACG takes the sequence length as an input rather than generating it. Its own evaluation takes lengths from natural held-out sequences, so

for each of the 100 samples per family we draw a length, with replacement, from that family's sequences in our training set; the mean sampled lengths (114.3, 145.2, and 72.5 nt) match the training means (115.4, 143.6, and 72.8 nt). The generated sequences are scored with `cmsearch` exactly as for the other methods.

C.4 Masked Sequence Reconstruction

We evaluate reconstruction on the held-out seed sequences of the large cohort, the 100 families sampled from the large group (Section A). For each sequence we mask a fraction $\rho \in \{15\%, 30\%, 50\%\}$ of the nucleotide positions uniformly at random. The masked positions are drawn once and stored, so every model, guided or unguided, is scored on the identical (sequence, mask) pairs. We report the macro average of the macro average, first the mean per-sequence recovery rate within a family and then the mean over the 100 families, so that families count equally regardless of how many sequences they contribute.

A masked sequence at rate ρ is exactly a draw from our forward corruption at time $t = 1 - \rho$, since the corruption independently masks each position with probability $1 - t$. RfamFlow therefore reconstructs with a single denoising forward pass at $t = 1 - \rho$ and no iterative sampling. At each masked position we keep the base with the largest logit among {A,C,G,U} and count it correct when it matches the original nucleotide. Because this is a one-step point prediction rather than a sampling trajectory, guidance shifts almost none of the argmax decisions, so we report the unguided pass in the main text.

The six baselines are bidirectional encoders pretrained with masked language modeling, and we query each with its own objective. We replace the masked nucleotides with the model's mask token, run one forward pass, and read the argmax at each masked position, restricting it to the four base logits because these vocabularies also contain IUPAC degenerate codes that a free argmax would otherwise sometimes prefer. ERNIE-RNA (Yin et al., 2024) is loaded from its released fairseq checkpoint and RNA-FM (Chen et al., 2022) from its released `rna_fm_t12` checkpoint, while RiNALMo (Penić et al., 2024), RNA-MSM (Zhang et al., 2024), RNAErnie (Wang et al., 2024a), and SpliceBERT (Chen et al., 2024) are loaded from the `multimolecule` release, all with single-nucleotide tokenization. Table 13 lists the sizes. Five of the six baselines are larger than RfamFlow-large, only SpliceBERT is smaller, and RiNALMo is more than ten times its size.

Table 13: Parameter counts and checkpoints of the masked-reconstruction baselines, with our two models for reference. Sizes are counted from the loaded checkpoints.

Model	Params	Checkpoint / version
ERNIE-RNA	86M	`ERNIE-RNA_pretrain`
RiNALMo	650M	`rinalmo-giga`
RNA-FM	99.5M	`rna_fm_t12`
RNAErnie	87M	`multimolecule/rnaernie`
RNA-MSM	96M	`multimolecule/rnamsm`
SpliceBERT	20M	`multimolecule/splicebert`
RfamFlow-large	58.5M	this work
RfamFlow-base	10.2M	this work

C.5 Zero-Shot Fitness Prediction

C.5.1 DMS Assays

We map each assay in RNAGym (Arora et al., 2025) to the Rfam family of the RNA it scans and evaluate on three whose family falls in our large cohort, so that RfamFlow is conditioned on a family it has seen in quantity: the tRNA fitness landscape (RF00005) (Li et al., 2016), the glmS ribozyme (RF00234) (Andreasson et al., 2020), and the CPEB3 ribozyme (RF00622) (Zhang et al., 2020). Each is a substitution scan of a single reference sequence, measuring growth-based relative fitness for tRNA and self-cleavage activity for the two ribozymes. Keeping every variant that carries

a fitness value gives 65,536 tRNA variants of length 72, 7,343 glmS variants of length 74, and 111,417 CPEB3 variants of length 81, with a median of 3, 2, and 4 substitutions from the reference.

None of the three libraries contains an explicitly labeled wild-type row, so we reconstruct the reference as the per-position consensus, the most frequent base at each position across the library. Because every position is mutated in only a minority of variants, the consensus recovers the published wild type. The reconstructed wild-type sequences are listed in Table 14, and all mutant and marginal scores are computed against them.

Table 14: The three DMS assays, with the number of scored variants, the shared variant length in nucleotides, and the reconstructed wild-type sequence used as the scoring reference.

Assay	Rfam	Variants	Len.	Wild-type sequence ($5' \to 3'$)
tRNA	RF00005	65,536	72	GUUCCGUUGGCGUAAUGGUAA CGCGUCUCCCUCCUAAGGAGA AGACUGCGGGUUCGAGUCCCG UACGGAACG
glmS ribozyme	RF00234	7,343	74	AUAAAAGCGCCAGAACUACAG AAAUGUAGUUGACGAGGAGGA GGUUUAUCGAAUUUUCGGCGG AUGCCUCCAAA
CPEB3 ribozyme	RF00622	111,417	81	UAACAGGGGGCCACAGCAGAA GCGUUCACGUCGCAGCCCCUG UCAGAUUCUGGUGAAUCUGCG AAUUCUGCUGUAUAUCUC

C.5.2 Scoring Schemes

Let x^{wt} be the wild type of an assay and let a variant x^m differ from it at the position set $S = \{i : x_i^m \neq x_i^{\mathrm{wt}}\}$. Conditioned on the family label c at diffusion time t, the model gives a per-position distribution over the four bases, obtained by a softmax restricted to the A, C, G, and U logits, which we write $\log p_t(i, b \mid x, c)$ for base b at position i in context x. We score a variant by two marginal estimators adapted from protein language models (Meier et al., 2021), and in both a higher score means higher predicted fitness.

The *WT-marginal* scheme runs a single forward pass on the fully revealed wild type at $t = t_{\max} = 0.999$ and reads the base log-probabilities $\log p^{\mathrm{wt}}(i, b)$ at every position. A variant is scored by the summed log-likelihood ratio at its mutated positions,

$$s_{\mathrm{wt}} = \sum_{i \in S} \big[\log p^{\mathrm{wt}}(i, x_i^m) - \log p^{\mathrm{wt}}(i, x_i^{\mathrm{wt}}) \big]. \tag{11}$$

It costs one forward pass for the whole assay, since every variant reads off the same wild-type pass.

The *mutant-marginal* scheme masks the mutated positions of a variant, keeps the rest revealed, and forwards at $t = \max(L - |S|, 1)/L$, the fraction of revealed positions. Reading the base log-probabilities at the masked positions gives

$$s_{\mathrm{mut}} = \sum_{i \in S} \big[\log p(i, x_i^m) - \log p(i, x_i^{\mathrm{wt}}) \big]. \tag{12}$$

Masking the mutated positions rather than revealing them stops the absorbing model from copying the observed token, so the score reflects a genuine prediction. This costs one forward pass per variant.

We correlate each score with the measured fitness by Spearman's ρ. As in reconstruction, scoring is a single forward pass rather than iterative sampling, so guidance barely changes the ranking, and we report the plain conditional scores, namely flat $\gamma = 1$ and nested weights $(1, 1)$. The three discriminative baselines are scored with the identical two schemes, replacing the flow-matching forward with one bidirectional encoder forward, while the two generative baselines are scored by their own likelihood proxies, which we run ourselves on the same variants: RFamLlama by its

family-tagged autoregressive log-likelihood and RfamGen by the variational ELBO of the per-family CM-VAE (Appendix C.3.2).

Table 15: The 100 large-cohort Rfam families with their taxonomy and RNA type.

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00001	Gene	rRNA	5S_rRNA	RF00002	Gene	rRNA	5_8S_rRNA
RF00003	Gene	snRNA; splicing	U1	RF00004	Gene	snRNA; splicing	U2
RF00005	Gene	tRNA	tRNA	RF00006	Gene	Gene	Vault
RF00009	Gene	ribozyme	RNaseP_nuc	RF00010	Gene	ribozyme	RNaseP_bact_a
RF00012	Gene	snRNA; snoRNA; CD-box	U3	RF00013	Gene	Gene	6S
RF00015	Gene	snRNA; splicing	U4	RF00017	Gene	Gene	Metazoa_SRP
RF00019	Gene	Gene	Y_RNA	RF00020	Gene	snRNA; splicing	U5
RF00023	Gene	Gene	tmRNA	RF00026	Gene	snRNA; splicing	U6
RF00028	Intron	Intron	Intron_gpI	RF00029	Intron	Intron	Intron_gpII
RF00030	Gene	ribozyme	RNase_MRP	RF00032	Cis-reg	Cis-reg	Histone3
RF00049	Gene	snRNA; snoRNA; CD-box	SNORD36	RF00050	Cis-reg	riboswitch	FMN
RF00059	Cis-reg	riboswitch	TPP	RF00066	Gene	snRNA	U7
RF00075	Gene	miRNA	MIR166	RF00080	Cis-reg	riboswitch	Manganese
RF00097	Gene	snRNA; snoRNA; CD-box	snoR71	RF00100	Gene	Gene	7SK
RF00156	Gene	snRNA; snoRNA; HACA-box	SNORA70	RF00162	Cis-reg	riboswitch	SAM
RF00163	Gene	ribozyme	Hammer-head_1	RF00167	Cis-reg	riboswitch	Purine
RF00168	Cis-reg	riboswitch	Lysine	RF00169	Gene	Gene	Bacteria_small_SRP
RF00174	Cis-reg	riboswitch	Cobalamin	RF00181	Gene	snRNA; snoRNA; CD-box	SNORD113
RF00230	Cis-reg	leader	T-box	RF00234	Cis-reg	riboswitch	glmS
RF00309	Gene	snRNA; snoRNA; CD-box	snosnR60_Z15	RF00321	Gene	snRNA; snoRNA; CD-box	snoZ247
RF00379	Cis-reg	riboswitch	c-di-AMP	RF00409	Gene	snRNA; snoRNA; HACA-box	SNORA7
RF00432	Gene	snRNA; snoRNA; HACA-box	SNORA51	RF00436	Cis-reg	Cis-reg	UnaL2
RF00442	Cis-reg	riboswitch	Guanidine-I	RF00485	Cis-reg	Cis-reg	K_chan_RES
RF00504	Cis-reg	riboswitch	Glycine	RF00515	Cis-reg	Cis-reg	PyrR
RF00548	Gene	snRNA; splicing	U11	RF00557	Cis-reg	leader	L10_leader
RF00560	Gene	snRNA; snoRNA; HACA-box	SNORA17	RF00561	Gene	snRNA; snoRNA; HACA-box	SNORA40

Table 15, continued

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00619	Gene	snRNA; splicing	U6atac	RF00622	Gene	ribozyme	CPEB3_ ribozyme
RF00634	Cis-reg	riboswitch	SAM-IV	RF00643	Gene	miRNA	MIR171_1
RF00645	Gene	miRNA	MIR169_2	RF00782	Gene	miRNA	MIR480
RF00865	Gene	miRNA	MIR169_5	RF00906	Gene	miRNA	MIR1120
RF00951	Gene	miRNA	mir-1302	RF01045	Gene	miRNA	mir-544
RF01051	Cis-reg	riboswitch	c-di-GMP-I	RF01055	Cis-reg	riboswitch	Molybdenum
RF01057	Cis-reg	riboswitch	SAH	RF01068	Cis-reg	riboswitch	Guanidine-II
RF01210	Gene	snRNA; snoRNA; CD-box	snoU13	RF01419	Gene	antisense	IsrR
RF01690	Gene	sRNA	Bacillaceae-1	RF01695	Gene	antisense	C4
RF01705	Gene	sRNA	Flavo-1	RF01734	Cis-reg	riboswitch	Fluoride
RF01750	Cis-reg	riboswitch	ZMP-ZTP	RF01766	Cis-reg	thermoregulator	cspA
RF01782	Gene	antisense	ASpks	RF01787	Gene	ribozyme	drz-agam-1
RF01847	Gene	snRNA; snoRNA; CD-box	Plant_U3	RF01852	Gene	tRNA	tRNA-Sec
RF01855	Gene	Gene	Plant_SRP	RF01912	Gene	miRNA	mir-2807
RF01924	Gene	miRNA	mir-2774	RF02001	Intron	Intron	group-II-D1D4-3
RF02271	Cis-reg	Cis-reg	uc_338	RF02514	Gene	sRNA	5_ureB_ sRNA
RF03167	Cis-reg	riboswitch	c-di-GMP-I-GGC	RF03577	Gene	miRNA	mir-9193
RF03662	Gene	miRNA	mir-2060	RF03874	Gene	miRNA	mir-8834
RF03883	Gene	miRNA	mir-9256	RF03940	Gene	miRNA	MIR9563
RF03981	Gene	miRNA	mir-8799	RF04021	Gene	miRNA	mir-3492
RF04032	Gene	miRNA	MIR2912	RF04088	Gene	miRNA	MIR812
RF04134	Gene	miRNA	MIR7504	RF04204	Gene	miRNA	mir-3473
RF04224	Gene	miRNA	mir-9229	RF04225	Gene	miRNA	MIR7502
RF04248	Gene	miRNA	MIR7486	RF04274	Gene	miRNA	mir-9223

Table 16: The 100 medium-cohort Rfam families with their taxonomy and RNA type.

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00007	Gene	snRNA; splicing	U12	RF00031	Cis-reg	Cis-reg	SECIS_1
RF00096	Gene	snRNA; snoRNA; CD-box	U8	RF00099	Gene	snRNA; snoRNA; CD-box	SNORD22
RF00108	Gene	snRNA; snoRNA; CD-box	SNORD116	RF00127	Gene	sRNA	t44
RF00135	Gene	snRNA; snoRNA; CD-box	snoZ223	RF00137	Gene	snRNA; snoRNA; CD-box	SNORD83
RF00142	Gene	snRNA; snoRNA; CD-box	snoZ118	RF00150	Gene	snRNA; snoRNA; CD-box	SNORD42

Table 16, continued

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00160	Gene	snRNA; snoRNA; CD-box	snoZ159	RF00203	Gene	snRNA; snoRNA; CD-box	snoR160
RF00272	Gene	snRNA; snoRNA; HACA-box	SNORA67	RF00277	Gene	snRNA; snoRNA; CD-box	SNORD49
RF00281	Gene	snRNA; snoRNA; CD-box	SNORD47	RF00315	Gene	snRNA; snoRNA; CD-box	snoJ33
RF00333	Gene	snRNA; snoRNA; CD-box	snoZ157	RF00335	Gene	snRNA; snoRNA; CD-box	snoZ13_ snr52
RF00352	Gene	snRNA; snoRNA; CD-box	snoR21	RF00391	Cis-reg	Cis-reg	RtT
RF00392	Gene	snRNA; snoRNA; HACA-box	SNORA5	RF00400	Gene	snRNA; snoRNA; HACA-box	SNORA28
RF00401	Gene	snRNA; snoRNA; HACA-box	SNORA20	RF00407	Gene	snRNA; snoRNA; HACA-box	SNORA50
RF00416	Gene	snRNA; snoRNA; HACA-box	SNORA43	RF00421	Gene	snRNA; snoRNA; HACA-box	SNORA32
RF00425	Gene	snRNA; snoRNA; HACA-box	SNORA18	RF00430	Gene	snRNA; snoRNA; HACA-box	SNORA54
RF00431	Gene	snRNA; snoRNA; HACA-box	SNORA55	RF00478	Gene	snRNA; snoRNA; scaRNA	SCARNA6
RF00566	Gene	snRNA; snoRNA; HACA-box	SNORA35	RF00601	Gene	snRNA; snoRNA; scaRNA	SCARNA20
RF00604	Gene	snRNA; snoRNA; CD-box	SNORD88	RF00613	Gene	snRNA; snoRNA; CD-box	SNORD94
RF00630	Gene	Gene	P26	RF00688	Gene	miRNA	MIR394
RF00692	Gene	miRNA	MIR171_2	RF00736	Gene	miRNA	mir-320
RF00832	Gene	miRNA	mir-71	RF01061	Gene	miRNA	mir-548
RF01073	Cis-reg	Cis-reg	GP_knot1	RF01179	Gene	snRNA; snoRNA; CD-box	snoU83
RF01185	Gene	snRNA; snoRNA; CD-box	snR75	RF01214	Gene	snRNA; snoRNA; CD-box	snR51
RF01219	Gene	snRNA; snoRNA; HACA-box	snoR100	RF01224	Gene	snRNA; snoRNA; HACA-box	snoR80
RF01227	Gene	snRNA; snoRNA; HACA-box	snoR83	RF01231	Gene	snRNA; snoRNA; HACA-box	snoR74
RF01234	Gene	snRNA; snoRNA; HACA-box	SNORA47	RF01241	Gene	snRNA; snoRNA; HACA-box	SNORA81

Table 16, continued

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF01281	Gene	snRNA; snoRNA; CD-box	snoR35	RF01291	Gene	snRNA; snoRNA; CD-box	SNORD97
RF01295	Gene	snRNA; snoRNA; scaRNA	SCARNA7	RF01299	Gene	snRNA; snoRNA; CD-box	SNORD2
RF01302	Gene	snRNA; snoRNA; CD-box	snoU36a	RF01320	Gene	CRISPR	CRISPR-DR7
RF01411	Gene	sRNA	BsrF	RF01424	Gene	snRNA; snoRNA; CD-box	snoR118
RF01504	Gene	sRNA	Afu_254	RF01514	Gene	snRNA; snoRNA; CD-box	Afu_513
RF01692	Cis-reg	leader	Bacteroid-trp	RF01693	Gene	sRNA	Bacteroidales-1
RF01701	Gene	sRNA	Cyano-1	RF01726	Cis-reg	riboswitch	SAM-II
RF01731	Cis-reg	Cis-reg	TwoAYG-GAY	RF01749	Cis-reg	Cis-reg	pan
RF01854	Gene	Gene	Bacteria_large_SRP	RF01998	Intron	Intron	group-II-D1D4-1
RF02047	Gene	lncRNA	Sphinx_2	RF02194	Gene	antisense	HPnc0260
RF02221	Gene	sRNA	sRNA-Xcc1	RF02253	Cis-reg	Cis-reg	IRE_II
RF02278	Gene	sRNA	Betaproteobacteria_toxic_sRNA	RF02347	Gene	sRNA	ar45
RF02349	Gene	sRNA	psRNA2	RF02413	Gene	snRNA; snoRNA; HACA-box	snoR145
RF02471	Gene	sRNA	Ms_IGR-5	RF02516	Gene	miRNA	MIR393
RF02542	Gene	rRNA	SSU_rRNA_microsporidia	RF02553	Gene	sRNA	YrlA
RF02605	Gene	sRNA	scr5239	RF02647	Gene	sRNA	Cis52_sRNA
RF02790	Gene	sRNA	sodF_sRNA	RF02835	Gene	sRNA	Bcj11
RF02996	Gene	sRNA	int-alpA	RF03034	Gene	sRNA	nrdJ
RF03057	Cis-reg	riboswitch	nhaA-I	RF03158	Cis-reg	leader	L31-Actinobacteria
RF03163	Gene	antisense	c4-a1b1	RF03749	Gene	miRNA	MIR2617
RF03789	Gene	miRNA	MIR5269	RF03833	Gene	miRNA	mir-5928
RF03860	Gene	miRNA	MIR1520	RF03896	Gene	miRNA	MIR2275
RF04223	Gene	miRNA	MIR2619	RF04287	Gene	miRNA	mir-9195
RF04319	Gene	sRNA	BTncRNA_FopS	RF04345	Cis-reg	Cis-reg	Plastid-rps12
RF04355	Intron	Intron	Plastid-atpF	RF04357	Intron	Intron	Plastid-ndhA-ndhB-2

Table 17: The 100 small-cohort Rfam families with their taxonomy and RNA type.

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00040	Cis-reg	Cis-reg	rne5	RF00042	Gene	antisense	CopA
RF00052	Gene	miRNA	lin-4	RF00057	Gene	sRNA	RyhB

Table 17, continued

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00079	Gene	sRNA	OmrA-B	RF00081	Gene	sRNA	ArcZ
RF00088	Gene	snRNA; snoRNA; CD-box	SNORD30	RF00095	Gene	snRNA; snoRNA; CD-box	snoPyro_CD
RF00112	Gene	sRNA	CyaR_RyeE	RF00118	Gene	sRNA	rydB
RF00126	Gene	sRNA	ryfA	RF00128	Gene	sRNA	GlmY_tke1
RF00136	Gene	snRNA; snoRNA; CD-box	SNORD81	RF00144	Gene	miRNA	mir-199
RF00165	Cis-reg	Cis-reg	Corona_pk3	RF00170	Gene	Gene	msr
RF00194	Cis-reg	Cis-reg	Rubella_3	RF00195	Gene	sRNA	RsmY
RF00221	Gene	snRNA; snoRNA; CD-box	SNORD43	RF00224	Cis-reg	IRES	IRES_FGF2
RF00226	Cis-reg	IRES	IRES_n-myc	RF00232	Cis-reg	Cis-reg	Spi-1
RF00238	Gene	antisense	ctRNA_ pND324	RF00240	Gene	antisense	RNA-OUT
RF00249	Gene	miRNA	mir-46	RF00256	Gene	miRNA	mir-196
RF00259	Cis-reg	Cis-reg	IFN_gamma	RF00261	Cis-reg	IRES	IRES_L-myc
RF00262	Gene	antisense	sar	RF00285	Gene	snRNA; snoRNA; CD-box	snoZ6
RF00287	Gene	snRNA; snoRNA; CD-box	SNORD44	RF00301	Gene	snRNA; snoRNA; CD-box	snoZ256
RF00341	Gene	snRNA; snoRNA; CD-box	snoZ39	RF00342	Gene	snRNA; snoRNA; CD-box	snoZ40
RF00351	Gene	snRNA; snoRNA; CD-box	snoR20	RF00355	Gene	snRNA; snoRNA; CD-box	snoR28
RF00368	Gene	sRNA	sroB	RF00376	Cis-reg	Cis-reg	HIV_GSL3
RF00394	Gene	snRNA; snoRNA; HACA-box	SNORA4	RF00403	Gene	snRNA; snoRNA; HACA-box	SNORA41
RF00404	Gene	snRNA; snoRNA; HACA-box	SNORA46	RF00417	Gene	snRNA; snoRNA; HACA-box	SNORA56
RF00418	Gene	snRNA; snoRNA; HACA-box	SNORA58	RF00426	Gene	snRNA; snoRNA; scaRNA	SCARNA15
RF00427	Gene	snRNA; snoRNA; scaRNA	SCARNA23	RF00434	Cis-reg	Cis-reg	BTE
RF00439	Gene	snRNA; snoRNA; CD-box	SNORD87	RF00440	Gene	snRNA; snoRNA; CD-box	SNORD37
RF00464	Gene	miRNA	mir-92	RF00467	Cis-reg	Cis-reg	RSV_PBS
RF00471	Gene	snRNA; snoRNA; CD-box	snosnR48	RF00479	Gene	snRNA; snoRNA; CD-box	snosnR71
RF00487	Cis-reg	IRES	IRES_Cx43	RF00492	Gene	snRNA; snoRNA; scaRNA	SCARNA17
RF00495	Cis-reg	IRES	IRES_Hsp70	RF00506	Cis-reg	leader	Thr_leader

Table 17, continued

Accession	Top	Sub	RNA	Accession	Top	Sub	RNA
RF00509	Gene	snRNA; snoRNA; CD-box	snosnR64	RF00512	Cis-reg	leader	Leu_leader
RF00526	Gene	snRNA; snoRNA; CD-box	snoMe28S-U3344	RF00527	Gene	snRNA; snoRNA; CD-box	snoMe28S-G3255
RF00528	Gene	snRNA; snoRNA; CD-box	snoMe28S-Gm1083	RF00529	Gene	snRNA; snoRNA; CD-box	snoMe28S-Am2589
RF00542	Gene	snRNA; snoRNA; HACA-box	snopsi28S-1192	RF00570	Gene	snRNA; snoRNA; CD-box	SNORD64
RF00578	Gene	snRNA; snoRNA; CD-box	SNORD89	RF00591	Gene	snRNA; snoRNA; CD-box	SNORD77
RF00594	Gene	snRNA; snoRNA; CD-box	SNORD86	RF00603	Gene	snRNA; snoRNA; CD-box	SNORD23
RF00608	Gene	snRNA; snoRNA; CD-box	SNORD99	RF00624	Gene	Gene	P9
RF00628	Gene	sRNA	RgsA	RF00647	Gene	miRNA	MIR164
RF00653	Gene	miRNA	mir-22	RF00659	Gene	miRNA	mir-365
RF00662	Gene	miRNA	mir-132	RF00664	Gene	miRNA	mir-223
RF00674	Gene	miRNA	mir-187	RF00682	Gene	miRNA	mir-144
RF00685	Gene	miRNA	mir-36	RF00708	Gene	miRNA	mir-450
RF00711	Gene	miRNA	mir-449	RF00712	Gene	miRNA	mir-460
RF00716	Gene	miRNA	mir-3	RF00717	Gene	miRNA	mir-315
RF00720	Gene	miRNA	mir-317	RF00722	Gene	miRNA	mir-451
RF00724	Gene	miRNA	mir-282	RF00727	Gene	miRNA	bantam
RF00729	Gene	miRNA	mir-278	RF00730	Gene	miRNA	mir-277
RF00735	Gene	miRNA	mir-367	RF00737	Gene	miRNA	mir-322
RF00745	Gene	miRNA	mir-499	RF00746	Gene	miRNA	mir-454
RF00749	Gene	miRNA	mir-208	RF00750	Gene	miRNA	mir-458
RF00763	Gene	miRNA	mir-339	RF00777	Gene	miRNA	mir-541
RF00780	Gene	miRNA	MIR477	RF00808	Gene	miRNA	mir-86