CogFormer: Learn All Your Models Once

Jerry M. Huang[1,†] **Lukas Schumacher**[2] **Niek Stevenson**[3] **Stefan T. Radev**[1]

[1] Center for Modeling, Simulation, and Imaging in Medicine (CeMSIM), Rensselaer Polytechnic Institute, Troy, New York, United States
[2] Center for Economic Psychology, University of Basel, Basel, Switzerland
[3] Amsterdam Mathematical Psychology Lab, University of Amsterdam , Amsterdam, Netherlands
[†]Correspondence to huangm13@rpi.edu

Abstract

Simulation-based inference (SBI) with neural networks has accelerated and transformed cognitive modeling workflows. SBI enables modelers to fit complex models that were previously difficult or impossible to estimate, while also allowing rapid estimation across large numbers of datasets. However, the utility of SBI for iterating over varying modeling assumptions remains limited: changes to parameterizations, generative functions, priors, and design variables all necessitate model retraining, thereby diminishing the benefits of amortization. To address these issues, we pilot the CogFormer, a meta-amortized framework for cognitive modeling. Our framework trains a transformer-based architecture that remains valid across a combinatorial number of structurally similar models, allowing for changing data types, parameters, design matrices, and sample sizes. We present promising quantitative results across families of decision-making models for binary, multi-alternative, and continuous responses. Our evaluation suggests that CogFormer can accurately estimate parameters across model families with minimal amortization offset, making it a potentially powerful engine that catalyzes cognitive modeling workflows.

1. Introduction

Neural simulation-based inference (SBI; Cranmer et al., 2020; Deistler et al., 2025) employs neural networks trained on simulated data to avoid iterative estimation. In particular, *amortized methods* (Bürkner et al., 2023) learn from simulated parameter–observation pairs $(\boldsymbol{\theta}, \mathbf{y})$, enabling rapid sampling from the posterior $p(\boldsymbol{\theta} \mid \mathbf{y}^*)$ for any new observation $\mathbf{y}^*$ without retraining. This advantage has led to a plethora of SBI applications in cognitive modeling (e.g., Radev et al., 2020; Wieschen et al., 2020; Fengler et al., 2021; Boelts et al., 2022; von Krause et al., 2022; Ghaderi-Kangavari et al., 2023; Sokratous et al., 2023; Schumacher et al., 2023, 2024; Rmus et al., 2024; Hato et al., 2025; Schaefer et al., 2025; Scholten et al., 2026; Kvam et al., 2024, 2025; Belov et al., 2026; Wu et al., 2026).

Cognitive modeling is concerned with building probabilistic models that connect latent constructs $\boldsymbol{\theta}$ (e.g., processing speed) to observable responses $\mathbf{y}$ (e.g., reaction times). These models are typically *generative*; that is, they define a recipe for generating synthetic data from parameters in the form of a conditional distribution $p(\mathbf{y} \mid \boldsymbol{\theta})$. However, the generative mapping does not typically preserve all information about $\boldsymbol{\theta}$, so inference needs to represent uncertainty via the posterior $p(\boldsymbol{\theta} \mid \mathbf{y})$. Since posteriors of cognitive models are rarely tractable, inference often relies on time-consuming iterative Monte Carlo methods (e.g., Strickland et al., 2018; Miletić et al., 2017), further underscoring the advantage of amortized SBI in its efficiency and flexibility.

In many ways, cognitive models are ideal generators of synthetic data: they are easy to implement, fast to simulate (often in the millisecond range), and low-dimensional. Yet they present unique challenges for SBI. In practice, inference rarely involves fitting only a single model. Instead, we fit multiple candidate models, run many iterative cycles of free vs. fixed parameter configurations, regress parameters on covariates, and generally reconfigure the model as theory and data negotiate (Voss et al., 2019; Evans, 2019; Stevenson et al., 2026). This dynamic process can be packaged into principled *Bayesian workflows* (Gelman et al., 2020; Schad et al., 2021; Li et al., 2024).

For example, when modeling response times with Sequential Sampling Models (SSM; Smith and Ratcliff, 2025, see also Appendix B), a researcher might compare variants with a fixed decision threshold to ones with a collapsing threshold that decreases over time, repeatedly refit the model while constraining or freeing parameters, or test whether an experimental manipulation affects a single parameter (e.g., drift rate) or multiple components of the decision process (von Krause et al., 2022). This creates a dilemma for SBI workflows, since model changes typically demand re-training the network, steadily eroding the very amortization benefits that made SBI attractive to begin with.

The current work builds on a stream in SBI that extends amortization (Chang et al., 2025; Gloeckler et al., 2024; Elsemüller et al., 2023; Schröder and Macke, 2023): training a single inference network that remains valid across many, potentially infinitely many, structurally similar models. We dub our architecture the **CogFormer**, directly inspired by the Simformer (Gloeckler et al., 2024). CogFormer can adapt to any design configurations within a model's combinatorial parameter space and infer latent parameters accordingly, without substantially compromising inference quality compared to single-model posterior approximation. In this work, we taxonomize the scope of amortization in cognitive modeling and demonstrate compelling results from the first meta-amortized inference engine designed for amortizing cognitive models.

2. Methods

2.1. Amortization scopes: from models to classes

To formalize the scope of amortization in cognitive modeling, we introduce a hierarchical taxonomy that organizes models by structural similarity. This taxonomy specifies the level at which inference can be amortized and clarifies the scope of generalization for a given neural estimator. We distinguish three nested levels: **model instance**, **model family**, and **model class** (see also Figure 1), where each level contains multiple instances of the level below.

A **model instance** $\mathcal{M}$ is at the most specific level, fully specified with fixed parameters and likelihood. All structural aspects, including the number of parameters, their interpretation, and the form of the generative process, are fixed. This has been the standard setting for SBI in cognitive modeling, for which a neural estimator is trained to perform amortized inference for a single $\mathcal{M}$.

A **model family** $\mathcal{F}$ comprises multiple specifications sharing the same computational structure but differing in parameterization or auxiliary assumptions. For example, variants of the classic diffusion decision model (DDM; Ratcliff, 1978; Ratcliff et al., 2004) may differ in collapsing vs. fixed boundaries (Bogacz et al., 2006; Ditterich, 2006; Rasanan et al., 2025), starting-point variability, non-decision time structure, or regressors on parameters. Importantly, these specifications are typically nested: simpler variants can be recovered from more general ones by fixing certain parameters to zero or other constant values. In theory, a single network can thus perform inference across all such variants.

A **model class** $\mathcal{C}$ groups distinct model families that implement a shared computational principle. For instance, in decision-making, families such as the DDM, racing diffusion model (RDM; Tillman et al., 2020; Zandbelt et al., 2014), and circular diffusion model (CDM; Smith, 2016; Rasanan et al., 2024) form a broader class of sequential sampling models (SSMs). These models are generally not nested and cannot be obtained from one another through simple parameter constraints. Therefore, amortization at this level requires generalization across structurally distinct families while still exploiting shared features, such as common target observables.

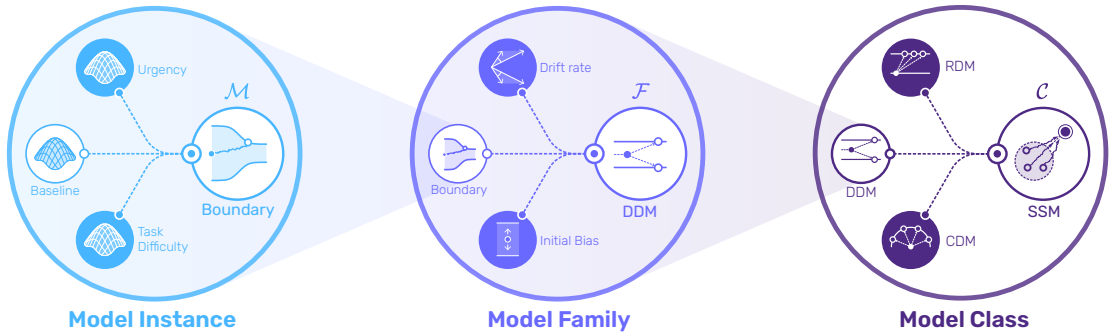

Figure 1. Taxonomy of amortization scopes. In cognitive modeling, one fits concrete model instances $\mathcal{M}$ nested within model families $\mathcal{F}$ as part of a larger model class $\mathcal{C}$. Here, we train neural estimators that generalizes across all model families within a given model class (in this paper, Sequential Sampling Models).

This paper demonstrates family- and class-level amortization, with future work aimed at even broader amortization scopes. Accordingly, we fix a model class $\mathcal{C}$ as the root and consider the following generative meta-model, sampled top-down:

$$p(\mathcal{F}, \mathcal{M}, \mathcal{D}, \theta_{\mathcal{F}}, \mathbf{Y} \mid \mathcal{C}) = p(\mathcal{F} \mid \mathcal{C})\, p(\mathcal{M}, \mathcal{D} \mid \mathcal{F})\, p(\theta_{\mathcal{F}} \mid \mathcal{M}, \mathcal{D})\, p(\mathbf{Y} \mid \theta_{\mathcal{F}}, \mathcal{M}, \mathcal{D}). \tag{1}$$

where $\mathcal{D}$ denotes a design configuration for the set of intrinsic parameters $\theta_{\mathcal{F}}$ and $\mathbf{Y}$ is model observables. The next sections describe how the abstract model ($\mathcal{M}$) and design ($\mathcal{D}$) variables are represented numerically through (embeddings of) positional encodings and design matrices and processed by our CogFormer architecture to arrive at an amortized posterior $p(\theta_{\mathcal{F}} \mid \mathbf{Y}, \mathcal{M}, \mathcal{D})$.

2.2. Meta-simulator and generative hierarchy

To realize the generative hierarchy in Eq. (1), we implement a meta-simulator that subsumes a broad class of cognitive models under a flexible generalized linear model (GLM) parameterization (see Figure 2; see also Algorithm 1 and Algorithm 2 for implementation details). Each model family $\mathcal{F}$ is characterized by $D_{\mathcal{F}}$ intrinsic parameters $\theta_{\mathcal{F}}$. For example, the DDM has six intrinsic parameters: drift rate ϑ, threshold a, bias z, non-decision time τ, and variabilities s_ϑ, s_τ. We express each intrinsic parameter as a linear function of covariates x_c, for example, $\vartheta = \beta_{0,\vartheta} + \beta_{1,\vartheta}\, x_1 + \beta_{2,\vartheta}\, x_2$. All regression weights are assembled into a coefficient matrix $\mathbf{B} \in \mathbb{R}^{I \times D_{\mathcal{F}}}$, where each column corresponds to one intrinsic parameter and each row to one column of the design matrix (intercept first, followed by regression weights). The intercepts are drawn from the intrinsic prior $p(\theta_{\mathcal{F}})$ while the regression weights are drawn from $\mathcal{N}(0, 1)$. A set of link functions $\mathcal{G} = \{g_j\}$ (e.g., sigmoid) maps the resulting linear predictors to the valid domain of each intrinsic parameter.

The generative process for a single dataset $\mathbf{Y} \in \mathbb{R}^{N \times O_{\mathcal{F}}}$ of N trials and $O_{\mathcal{F}}$ observables is thus:

$$\mathbf{Y} \sim \text{Model}(\Theta; \text{RNG}), \quad \Theta = g(\mathbf{X}\,(\mathbf{M} \odot \mathbf{B})) \tag{2}$$

where $\Theta \in \mathbb{R}^{N \times D_{\mathcal{F}}}$ is the realized parameter matrix, distinct from the intrinsic parameter vector $\theta_{\mathcal{F}}$, and RNG denotes the source of stochastic variation in the model outputs. The design matrix $\mathbf{X} \in \mathbb{R}^{N \times I}$ encodes the experimental structure across N observations and I design columns (i.e., regressors). Each $\mathbf{X}$ is sampled from a prior over design configurations. Its first column is an intercept (all ones), followed by one block of $C_{\max} - 1$ columns per regressor: continuous regressors occupy only the first column of their block; categorical regressors use all columns as dummy codes. Interaction terms are generated via column-wise multiplication. The parameter mask $\mathbf{M} \in \{0, 1\}^{I \times D_{\mathcal{F}}}$, sharing the shape of $\mathbf{B}$, distinguishes active (1) from fixed (0) coefficients. During training, each entry

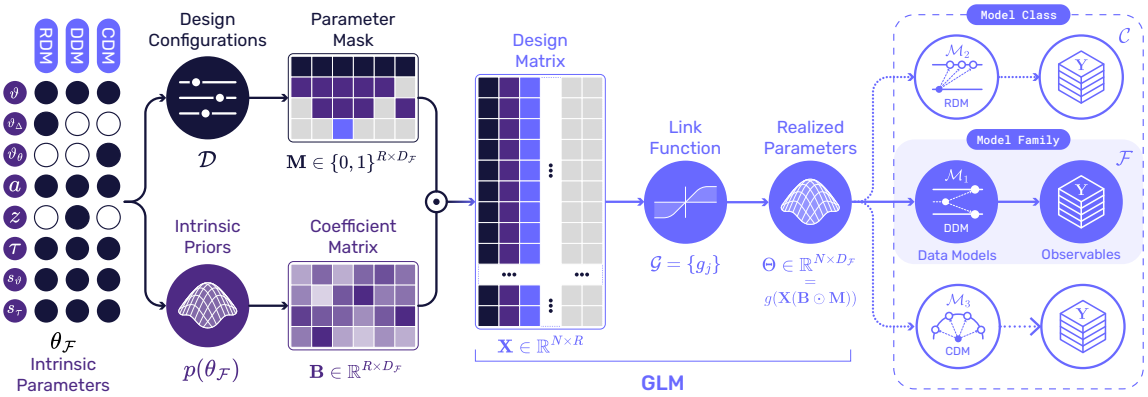

Figure 2. Meta-simulator architecture. For a single model family (*middle*), a coefficient matrix $\mathbf{B}$ is constructed under a flexible generalized linear model (GLM) configuration, and configured through a random binary parameter mask $\mathbf{M}$. Together with a sampled design matrix $\mathbf{X}$ and element-wise link functions g, this construction induces the trial-level parameter matrix Θ, which is passed to the observation model $\mathcal{M}$ to generate observables $\mathbf{Y}$. The same generative process applies to every model in the class $\mathcal{C}$.

$M_{i,j}$ is sampled from a Bernoulli distribution with a random probability parameter $p \sim \text{Beta}(2, 2)$, enabling the model to generalize to a power set of nested design configurations; during validation, $\mathbf{M}$ is fixed to pre-defined benchmark designs (see Appendix F).

Since N and I vary across simulations, $\mathbf{X}$ is zero-padded to a maximum capacity $I_{\max}$, reflecting the pragmatic limit on regressors in a realistic analysis. To avoid static padding, the number of trials N is held constant within a batch but varies across batches. Crucially, the parameter mask $\mathbf{M}$, design matrix $\mathbf{X}$, and coefficients $\mathbf{B}$ are all resampled for each individual simulation within a batch. Finally, to achieve amortization over model families, a model family index $f \sim p(f \mid \mathcal{C})$ is sampled for each simulation.

2.3. Generative encoder-decoder architecture

Flow matching for expressive posterior estimation Let $\mathcal{A} = (\mathbf{Y}, \mathbf{X}, \mathbf{M}, f)$ be a placeholder for all conditioning variables defining the amortization scope; equivalently, $\mathcal{A}$ collects the data $\mathbf{Y}$, the design $\mathcal{D} = (\mathbf{X}, \mathbf{M})$, and the model family index f. The goal of our framework is to estimate the full joint posterior $p(\mathbf{B} \mid \mathcal{A})$ without making restrictive parametric or autoregressive assumptions. To achieve this flexibility, we leverage *conditional flow matching* (Lipman et al., 2023; Liu et al., 2022). This choice is motivated by recent benchmarks in SBI suggesting that flow matching is a highly competitive free-form neural sampler (Arruda et al., 2025). Notably, our architecture is fully compatible with other free-form inference methods, such as score-based diffusion.

Intuitively, flow matching learns to transport a simple distribution, typically a Gaussian latent $\mathbf{Z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, into the complex target posterior $\mathbf{Z}_0 \equiv \mathbf{B} \sim p(\mathbf{B} \mid \mathcal{A})$ by learning a time-dependent conditional velocity field $\hat{u}(\mathbf{Z}_t; \mathcal{A}, t)$. The velocity field defines an ordinary differential equation (ODE) with boundary conditions so that $\mathbf{Z}_1$ is noise and $\mathbf{Z}_0$ is the target coefficient matrix. The model then learns an approximate velocity field $\hat{u}$ by minimizing the flow matching objective:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{p(t)\, p(\mathbf{B},\mathcal{A})\, p(\mathbf{Z}_1 \mid \mathbf{Z}_0)} \left[\omega_t \left\| \mathbf{M} \odot \left(\hat{u}(\mathbf{Z}_t; \mathcal{A}, t) - (\mathbf{Z}_1 - \mathbf{B}) \right) \right\|_2^2 \right] \tag{3}$$

where ω_t is an optional weight and $\mathbf{Z}_t = (1 - t)\mathbf{B} + t\mathbf{Z}_1$ defines a linear probability path (Liu et al., 2022). The expectation is taken over a uniform time distribution, the meta-simulator, and the Gaussian latent distribution of $\mathbf{Z}_1$; the mask $\mathbf{M}$ in the norm operator $\| \cdot \|$ ensures that only active coefficients contribute to the loss. At inference time, we generate samples from the approximate posterior $q(\mathbf{B} \mid \mathcal{A}, \hat{u})$ by integrating the learned velocity field from $t = 1$ to $t = 0$ using any off-the-shelf ODE solver. In our experiments, we use a simple Euler scheme with $\Delta t = 10^{-3}$, but various error-correcting schemes are also viable (Arruda et al., 2025).

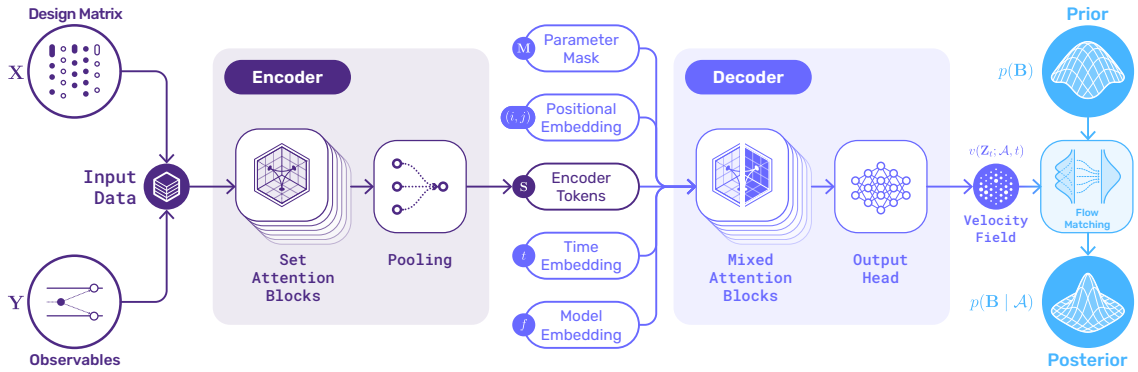

Figure 3. CogFormer architecture. CogFormer is an encoder-decoder architecture in which the encoder is a Set Transformer that converts a design matrix $\mathbf{X}$ and model observables $\mathbf{Y}$ into summary representations $\mathbf{S}$, and the decoder is a stack of mixed-attention layers that produce a learned velocity field $\hat{u}$. The latter parameterizes the learned velocity field $\hat{u}$. Training minimizes a conditional flow-matching objective, yielding a procedure for sampling from the resulting ensemble posterior.

Transformer backbone To realize the conditional velocity field $\hat{u}$, we develop a transformer-based encoder–decoder architecture (see Figure 3; further details about the individual attention blocks can be found in Figure 8). The architecture is dictated by an asymmetry in our problem: the observations are exchangeable, while the parameters are not. CogFormer treats each side accordingly.

The **encoder** is a modernized version of the Set Transformer (Lee et al., 2019) that encodes the design matrix $\mathbf{X}$ and the model observables $\mathbf{Y}$ into a sequence of L learned summary representations, $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_L)$ with $\mathbf{s}_\ell \in \mathbb{R}^S$. Since the trial-level pairs $(\mathbf{x}_n, \mathbf{y}_n)$ are exchangeable, the posterior is invariant to joint permutations of the rows of $(\mathbf{X}, \mathbf{Y})$; a set-based encoder bakes this invariance in by construction rather than forcing the network to learn it. As a result, the permutation-invariant encoder can handle varying trial counts and experimental designs.

The **decoder** maps the encoder summaries $\mathbf{S}$ to the velocity field $\hat{u}$ of the parameters $\mathbf{B}$. Unlike the observations, the parameters are not exchangeable: each entry of $\mathbf{B} \in \mathbb{R}^{I \times D_{\mathcal{F}}}$ plays a distinct role, and their posterior correlations are themselves objects of inference. This calls for structured sequence attention rather than a set operation. A unique feature of our architecture is that it unrolls the coefficient matrix $\mathbf{B}$ into a 1D sequence of $I \times D_{\mathcal{F}}$ tokens, which is what allows us to work with both varying numbers of columns (intrinsic parameters) and rows (regressors).

To restore the spatial layout of the GLM coefficient matrix from the unrolled $\mathbf{B}$, each token is augmented with sinusoidal embeddings of the positional encoding triplet (i, j, f): the regressor index i, the intrinsic parameter index j, and the model family index f. The first two coordinates re-impose the grid structure a flat sequence would destroy, while f tells each parameter which model family it belongs to, so that semantically identical positions are not conflated across families. The binary parameter mask $\mathbf{M}$ is also provided as a query mask, allowing the network to distinguish active from fixed parameters.

Information in the decoder flows through mixed-attention blocks, each of which alternates between a cross-attention layer over the encoder summaries $\mathbf{S}$ and a self-attention layer over the tokens formed by concatenating $\mathbf{Z}_t$ with the positional embeddings. The use of both types of attention layers addresses the two things a posterior must do at once: cross-attention conditions each parameter on the data summary, while self-attention propagates information across the parameter grid to capture their joint dependencies. Within each decoder block, we use a residual network as the output head to fuse these inputs. Although FiLM (Perez et al., 2018) was found to improve conditional flow matching in previous SBI work (Wildberger et al., 2023; Arruda et al., 2025), it did not result in significant improvements in our meta-amortized setting (see results in Appendix G).

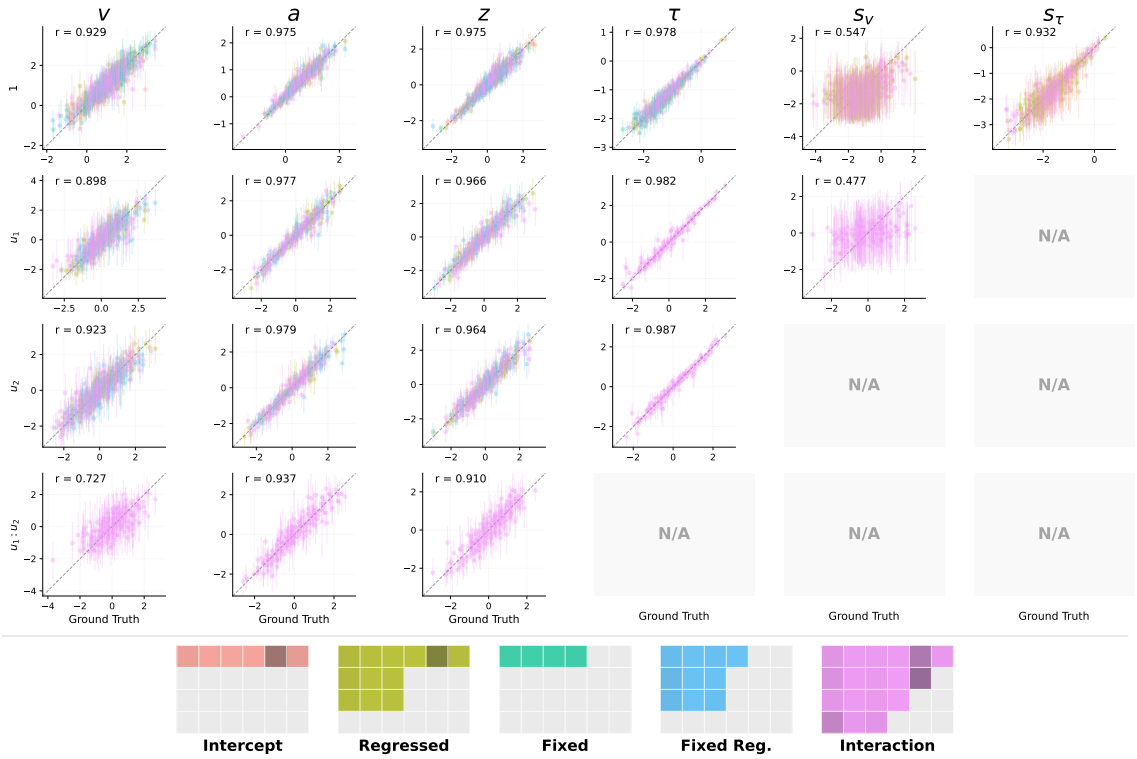

Figure 4. Ensemble parameter recoveries for CogFormer-$\mathcal{F}$ applied to the five design configurations for the DDM benchmark cases overlaid in their corresponding mapping to the shared coefficient matrix. The pixel thumbnails indicate the active estimation targets for each design configuration, with lighter shades of color depicting higher Pearson correlations r. The per-cell correlations are averaged across configurations.

3. Related work

Meta-amortization has earlier roots in variational settings. For example, MetaVAE (Wu et al., 2020) proposed doubly amortized inference across a family of related probabilistic models via a MetaELBO objective, but it was not designed for flexible posteriors in SBI. Within SBI, earlier work considered simultaneous inference both over discrete model indices (Radev et al., 2020) and over discrete models and continuous parameters for compositional simulators (Schröder and Macke, 2023).

A separate line of work moves toward multi-query SBI, in which a single amortized artifact can answer many conditional inference queries at test time. For instance, SimFormer (Gloeckler et al., 2024) trains a transformer-based diffusion model on the joint distribution of parameters and observations and uses masking and attention structure to sample arbitrary conditionals, including posteriors and likelihoods. OneFlowSBI (Nautiyal et al., 2026) pursues a similar goal with flow matching: it learns a vector field over the joint state space and realizes posterior, likelihood, and mixed conditionals. Reuter et al. (2025) amortizes over different statistical model families (GLMs, latent factor models) within a single in-context learning framework. These primarily expand query flexibility within a single learned joint model and serve as inspiration for our masking approach. In contrast, we shift the emphasis from many conditionals of a single joint distribution to a flexible posterior estimator that spans a power set of structural model configurations.

A complementary approach pursues amortization across different tasks, such as predictive and posterior estimation. For instance, ACE (Chang et al., 2025) uses transformers to flexibly reinterpret latent and observed variables at training and test time, enabling both runtime prior injection and continuous and discrete targets via parametric heads. Closest in spirit is SA-ABI (Elsemüller et al.,

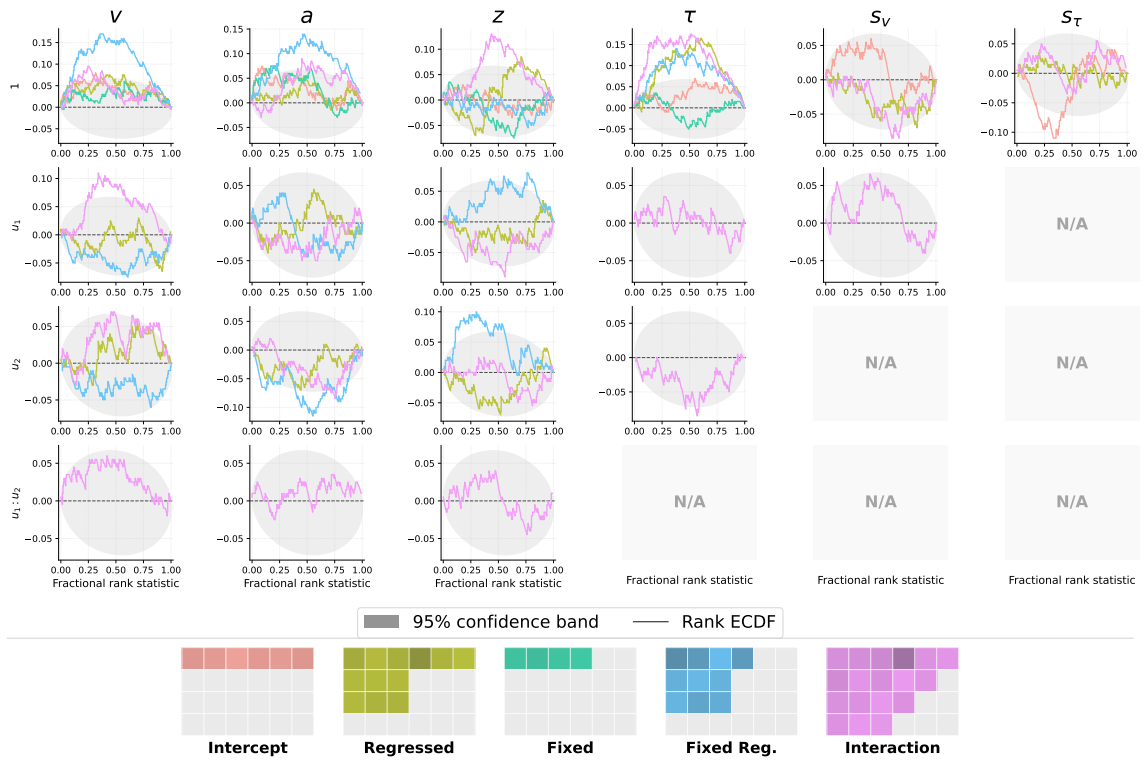

Figure 5. Ensemble calibration ECDFs for CogFormer-$\mathcal{F}$ applied to five design configurations for the DDM benchmark cases overlaid in their corresponding mapping to the coefficient matrix. The pixel thumbnails (bottom row) indicate the active estimation targets for each design configuration in the shared coefficient matrix, with lighter shades depicting lower calibration errors (ECEs).

2023), which formalizes sources of sensitivity, i.e., priors and data model assumptions, as context variables and uses weight sharing and deep ensembles to reduce refitting in sensitivity analysis. Similarly, the Distribution Transformer (Whittle et al., 2025) maps prior distributions directly to (time-varying) posteriors via learnable Gaussian mixtures. In our IID cognitive-modeling settings, we take meta-amortization one step further by substantially expanding generalization to combinatorial design spaces and learning an expressive neural estimator without parametric assumptions. Furthermore, we conduct a systematic and multidimensional analysis of amortization gaps.

4. Experiments

Meta-model Our experiments focus on SSMs, a widely used class of decision-making models that capture the dynamics of information processing underlying both choices and response times via stochastic differential equations (Smith and Ratcliff, 2025). Within this model class, we evaluate our framework on three representative model families: the DDM, the RDM, and the CDM. Detailed formulations, parameterizations, and priors for each model are provided in Appendix B. We train CogFormer with meta-simulators both for each model family (CogFormer-$\mathcal{F}$) and for the model class as a whole (CogFormer-$\mathcal{C}$).

These models were selected for several reasons. First, they have been extensively validated in cognitive neuroscience and psychology (Ratcliff and McKoon, 2008; Voss et al., 2004; Forstmann et al., 2016; Smith et al., 2020; Tillman et al., 2020), providing a well-established benchmark for inference methods. Second, despite differences in their generative structure, they share core parameters such as drift rates, boundary separation, and non-decision times.

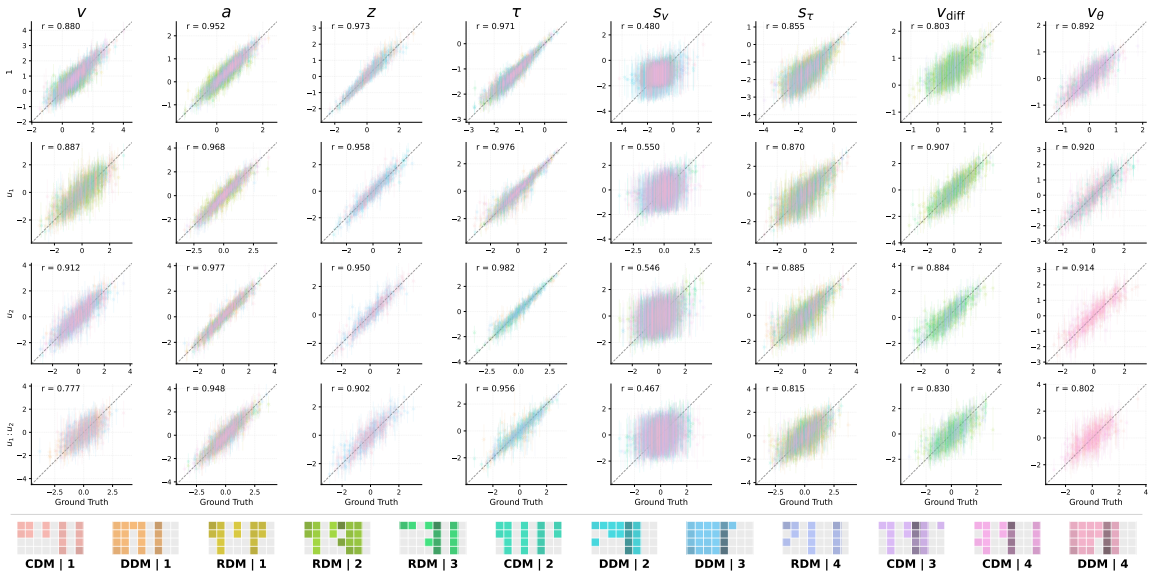

Figure 6. Ensemble parameter recoveries for CogFormer-$\mathcal{C}$ for 2,400 simulated datasets with 500 trials each over 12 randomly generated design configurations. The pixel thumbnails at the bottom indicate the active estimation targets within each configuration, with lighter shades of color indicate higher Pearson correlation r. Results are overlaid on top of each other, and the per-cell values are averaged across configurations.

At the same time, the three models differ in key aspects, including response format, dimensionality, and internal dynamics, making them sufficiently diverse to test generalization. Further, they span the spectrum of response types commonly studied in decision-making: the DDM models binary choices, the RDM multi-alternative responses, and the CDM continuous responses.

Evaluation metrics We evaluate the quality of the trained amortized estimators using multiple complementary metrics (see also Appendix E for more information). In particular, we assess posterior quality and the resulting amortization gap across progressively broader levels of data model amortization: instance ($\mathcal{M}$; baseline), family ($\mathcal{F}$), and class ($\mathcal{C}$). Posterior quality is evaluated using comprehensive diagnostics from the Bayesian workflow (Schad et al., 2021). Specifically, we report (i) normalized root mean square error (NRMSE) between true data-generating parameters and posterior samples, (ii) expected calibration error (ECE), and (iii) posterior contraction (PC). (i) and (iii) serve as a proxy for posterior sharpness (Gneiting et al., 2007), while (ii) reflects computational faithfulness. Finally, to quantify distributional differences between posterior estimates obtained under different amortization levels, we report results from classifier two-sample tests between samples from approximate joint posteriors (C2ST; Lopez-Paz and Oquab, 2016).

Conservative evaluation To ensure comparative fairness, we first defined five practically relevant design configurations as testing points. Details of these design configurations can be found in Appendix F. For our baseline, we used the `BayesFlow` software (Kühmichel et al., 2026) to train separate flow matching architectures for each of the five design configurations over 1000 epochs, with 100 steps per epoch and 64 simulations per training step (i.e., batch size). These architectures also use a Set Transformer (Lee et al., 2019) as a summary network. Additionally, we train CogFormer for each of the three model families ($\mathcal{F}$) and the model class ($\mathcal{C}$) over 5000 epochs, with 100 steps per epoch and 64 simulations per training step. Thus, *the total training budgets for all levels of amortization are the same*: 32,000,000 simulations. We evaluate sharpness metrics on 200 held-out simulations per configuration and compute the C2ST on pairs of posteriors obtained from a subset of 10 datasets. All network hyperparameters are listed in Appendix D.

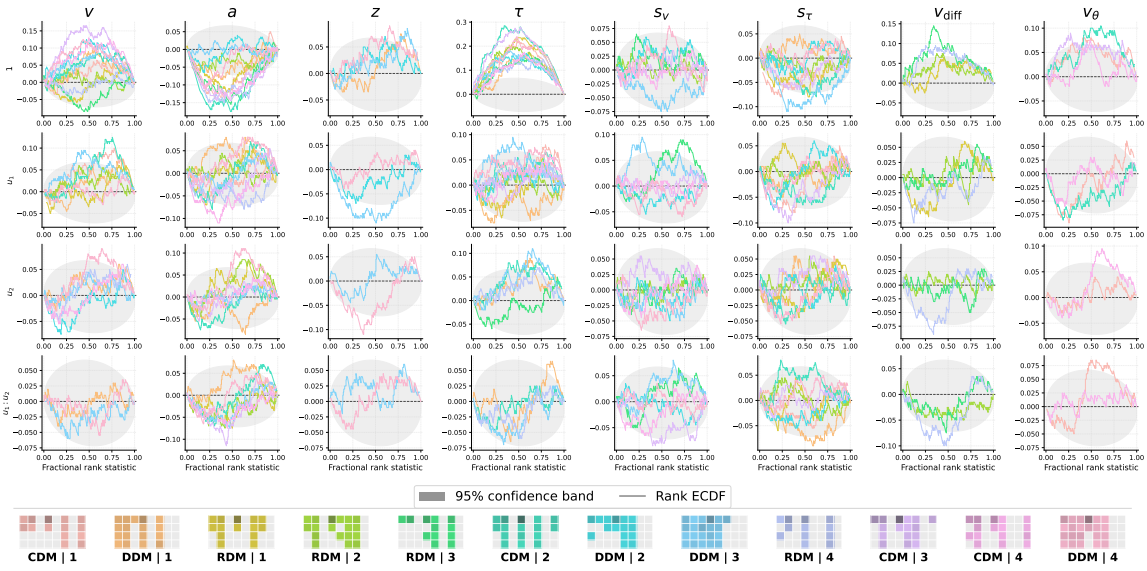

Figure 7. Ensemble calibration ECDFs for CogFormer-$\mathcal{C}$ for 2,400 simulated datasets with 500 trials each over 12 model configurations. The pixel thumbnails at the bottom indicate the active estimation targets within each configuration, with lighter shades indicating lower calibration errors (ECEs). Results are plotted corresponding to these thumbnails at their corresponding locations within the shared coefficient matrix.

Ablation studies To comprehensively examine CogFormer's architecture, we also conducted a series of ablation studies (see results in Appendix G).

4.1. Generalization within model families

To demonstrate generalization across model families, we use CogFormer-$\mathcal{F}$ to evaluate parameter estimation across the pre-specified benchmark design configuration ensemble. Our recovery results for DDM are shown in Figure 4, with further results for RDM and CDM, along with their calibration ECDF available in Appendix L. We observe that, across all configurations, all coefficients $\mathbf{B}$ show comparable recovery to the baseline. This suggests that CogFormer can perform as well as single-model amortized frameworks, while covering more design configurations under the same simulation budget. Further, we evaluate the calibration ECDF for these benchmark configurations (see Figure 5). The results suggest CogFormer-$\mathcal{F}$ is generally well-calibrated across all benchmark configurations, with calibration curves largely within the 95% confidence bands for each parameter.

4.2. Generalization within a model class

To demonstrate generalizability across model classes and beyond the benchmarks above, we use CogFormer-$\mathcal{C}$ to evaluate parameter recovery over a random ensemble of model configurations. This ensemble is generated by 1) randomly selecting a model family, 2) selecting the corresponding subset of parameters within the model class, and 3) specifying a randomized parameter mask given a defined free and fixed parameter configurations. To ensure minimal bias towards any specific model family, we evenly distribute model family selection so that the ensemble includes 4 configurations from each of the DDM, RDM, and CDM families (a total of 12 configurations).

The random ensemble parameter recovery results are shown in Figure 6, with more results presented in Appendix K, including excellent results for each of the five test configurations shown in the previous section. As can be observed, all estimation targets are well recovered and generally well-calibrated across all model families within the model class.

Table 1. Absolute metric values (median ± SEM) aggregated across the five benchmark design configurations and all parameters in each configuration.

Model	Method	NRMSE (↓)	Cal. Error (↓)	Post. Contr. (↑)
DDM	**Baseline**	0.078 ± 0.007	0.022 ± 0.002	0.902 ± 0.023
	CogFormer-$\mathcal{F}$	0.087 ± 0.007	0.021 ± 0.002	0.881 ± 0.025
	CogFormer-$\mathcal{C}$	0.091 ± 0.007	0.029 ± 0.003	0.862 ± 0.026
RDM	**Baseline**	0.112 ± 0.008	0.061 ± 0.010	0.803 ± 0.032
	CogFormer-$\mathcal{F}$	0.112 ± 0.009	0.026 ± 0.004	0.786 ± 0.033
	CogFormer-$\mathcal{C}$	0.117 ± 0.009	0.029 ± 0.003	0.750 ± 0.036
CDM	**Baseline**	0.075 ± 0.008	0.020 ± 0.001	0.896 ± 0.030
	CogFormer-$\mathcal{F}$	0.083 ± 0.008	0.024 ± 0.002	0.874 ± 0.032
	CogFormer-$\mathcal{C}$	0.087 ± 0.008	0.028 ± 0.002	0.859 ± 0.032

Table 2. Joint C2ST (median ±SEM) aggregated across 10 posteriors for each configuration.

Comparison	DDM	RDM	CDM
CogFormer-$\mathcal{F}$ vs. **Baseline**	0.693 ± 0.067	0.702 ± 0.076	0.668 ± 0.068
CogFormer-$\mathcal{C}$ vs. **CogFormer-$\mathcal{F}$**	0.640 ± 0.049	0.600 ± 0.032	0.635 ± 0.041

4.3. Amortization gaps

To summarize performance at each amortization level, we aggregate results across all model configurations and parameters, and report absolute values in Table 1. Across all three model families, CogFormer-$\mathcal{F}$ and CogFormer-$\mathcal{C}$ achieve NRMSE within ∼15% of the baseline, with posterior contraction declining by only 2–4% in absolute terms—a consistent and modest cost of vastly expanded amortization. Calibration error in absolute values is consistently minimal across all three model families. Finally, posterior contraction tends to decrease with broader amortization scope.

The C2ST results in Table 2 show that while the marginal posteriors of CogFormer-$\mathcal{F}$, CogFormer-$\mathcal{C}$, and the single-posterior baseline are highly similar, there are detectable differences in their joint posteriors. This suggests that, when simulation budgets are matched, meta-amortization induces an amortization gap in capturing the joint structure. The gap can be reduced by increasing the simulation budget for meta-amortization, as deemed typical in practice (see results in Appendix J).

5. Conclusions and future work

In this work, we presented CogFormer, a meta-amortized Bayesian framework for cognitive modeling. Leveraging a transformer-based architecture that flexibly adapts to varying configurations of structurally similar cognitive models, we can use CogFormer to learn all your models once. We demonstrated that, at a minimal amortization cost, we can reliably estimate model parameters for any given design configuration within a combinatorial design space, offering a powerful approach to further accelerate cognitive modeling workflows.

CogFormer reveals endless possibilities for further developments and improvements. For example, one of its current limitations is that it only deals with exchangeable models, and is not yet capable of amortizing more complex probabilistic symmetries, such as hierarchical models (Elsemüller et al., 2023) and dynamic models (Schumacher et al., 2023). These generative landscapes are what we intend to tackle in our future work. Further, adding entirely new models to a class would currently necessitate retraining or fine-tuning; the latter is what we intend to explore in future work, along with unsupervised continual learning (Mishra et al., 2026). We hope that all results inspire cognitive modelers about the potential for a general, shareable, and extensive end-to-end inference framework.

Acknowledgments

This work was funded by the National Science Foundation under Grant No. 2448380. We would like to thank Niels L. Bracher for his insightful feedback on the transformer backbone and the manuscript. We would also like to thank Mischa von Krause on his helpful suggestions on sequential sampling model implementation.

Code Availability

The code and interactive demo associated with this paper is available as a GitHub repository: https://github.com/bayesflow-org/CogFormer.

References

Jonas Arruda, Niels Bracher, Ullrich Köthe, Jan Hasenauer, and Stefan T Radev. Diffusion models in simulation-based inference: A tutorial review. *arXiv preprint arXiv:2512.20685*, 2025.

Dmitry I Belov, Oliver Lüdtke, and Esther Ulitzsch. A supervised learning approach to estimating irt models in small samples. *British Journal of Mathematical and Statistical Psychology*, 79(1): 66–94, 2026.

Jan Boelts, Jan-Matthis Lueckmann, Richard Gao, and Jakob H Macke. Flexible and efficient simulation-based inference for models of decision-making. *Elife*, 11:e77220, 2022.

Rafal Bogacz, Eric Brown, Jeff Moehlis, Philip Holmes, and Jonathan D. Cohen. The physics of optimal decision making: A formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review*, 113(4):700–765, 2006. ISSN 1939-1471. doi: 10.1037/0033-295X.113.4.700.

Paul-Christian Bürkner, Maximilian Scholz, and Stefan T Radev. Some models are useful, but how do we know which ones? towards a unified bayesian model taxonomy. *Statistic Surveys*, 17: 216–310, 2023.

Paul E Chang, Nasrulloh Loka, Daolang Huang, Ulpu Remes, Samuel Kaski, and Luigi Acerbi. Amortized probabilistic conditioning for optimization, simulation and inference. In *International Conference on Artificial Intelligence and Statistics*, pages 703–711. JMLR, 2025.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Michael Deistler, Jan Boelts, Peter Steinbach, Guy Moss, Thomas Moreau, Manuel Gloeckler, Pedro LC Rodrigues, Julia Linhart, Janne K Lappalainen, Benjamin Kurt Miller, et al. Simulation-based inference: A practical guide. *arXiv preprint arXiv:2508.12939*, 2025.

Jochen Ditterich. Evidence for time-variant decision making. *European Journal of Neuroscience*, 24 (12):3628–3641, 2006. ISSN 1460-9568. doi: 10.1111/j.1460-9568.2006.05221.x.

Lasse Elsemüller, Hans Olischläger, Marvin Schmitt, Paul-Christian Bürkner, Ullrich Köthe, and Stefan T Radev. Sensitivity-aware amortized bayesian inference. *arXiv preprint arXiv:2310.11122*, 2023.

Nathan J Evans. Assessing the practical differences between model selection methods in inferences about choice response time tasks. *Psychonomic bulletin & review*, 26(4):1070–1098, 2019.

Alexander Fengler, Lakshmi N Govindarajan, Tony Chen, and Michael J Frank. Likelihood approximation networks (lans) for fast inference of simulation models in cognitive neuroscience. *Elife*, 10:e65074, 2021.

Birte U Forstmann, Roger Ratcliff, and E-J Wagenmakers. Sequential sampling models in cognitive neuroscience: Advantages, applications, and extensions. *Annual review of psychology*, 67(1):641–666, 2016.

Andrew Gelman, Aki Vehtari, Daniel Simpson, Charles C Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. Bayesian workflow. *arXiv preprint arXiv:2011.01808*, 2020.

Amin Ghaderi-Kangavari, Jamal Amani Rad, and Michael D Nunez. A general integrative neurocognitive modeling framework to jointly describe eeg and decision-making on single trials. *Computational Brain & Behavior*, 6(3):317–376, 2023.

Manuel Gloeckler, Michael Deistler, Christian Weilbach, Frank Wood, and Jakob H Macke. All-in-one simulation-based inference. *arXiv preprint arXiv:2404.09636*, 2024.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(2): 243–268, 2007.

Tuba Hato, Lukas Schumacher, Stefan T. Radev, and Andreas Voss. Lévy Versus Wiener: Assessing the Effects of Model Misspecification on Diffusion Model Parameters. *Computational Brain & Behavior*, May 2025. doi: 10.1007/s42113-025-00248-6.

Lars Kühmichel, Jerry M Huang, Valentin Pratz, Jonas Arruda, Hans Olischläger, Daniel Habermann, Simon Kucharsky, Lasse Elsemüller, Aayush Mishra, Niels Bracher, et al. Bayesflow 2.0: Multi-backend amortized bayesian inference in python. *arXiv preprint arXiv:2602.07098*, 2026.

Peter D Kvam, Konstantina Sokratous, Anderson Fitch, and Arend Hintze. Using artificial intelligence to fit, compare, evaluate, and discover computational models of decision behavior. *Decision*, 2024.

Peter D Kvam, Konstantina Sokratous, Anderson K Fitch, and Jasmin Vassileva. Comparing likelihood-based and likelihood-free approaches to fitting and comparing models of intertemporal choice. *Behavior Research Methods*, 57(9):252, 2025.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

Chengkun Li, Aki Vehtari, Paul-Christian Bürkner, Stefan T Radev, Luigi Acerbi, and Marvin Schmitt. Amortized bayesian workflow. *arXiv preprint arXiv:2409.04332*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XVjTT1nw5z.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *International conference on artificial intelligence and statistics*, pages 343–351. PMLR, 2021.

Steven Miletić, Brandon M. Turner, Birte U. Forstmann, and Leendert van Maanen. Parameter recovery for the Leaky Competing Accumulator model. *Journal of Mathematical Psychology*, 76: 25–50, February 2017. ISSN 0022-2496. doi: 10.1016/j.jmp.2016.12.001.

Aayush Mishra, Šimon Kucharskỳ, and Paul-Christian Bürkner. Unsupervised continual learning for amortized bayesian inference. *arXiv preprint arXiv:2602.22884*, 2026.

Mayank Nautiyal, Li Ju, Melker Ernfors, Klara Hagland, Ville Holma, Maximilian Werkö Söderholm, Andreas Hellander, and Prashant Singh. Oneflowsbi: One model, many queries for simulation-based inference. *arXiv preprint arXiv:2601.22951*, 2026.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Stefan T Radev, Andreas Voss, Eva Marie Wieschen, and Paul-Christian Bürkner. Amortized bayesian inference for models of cognition. *arXiv preprint arXiv:2005.03899*, 2020.

Amir Hosein Hadian Rasanan, Nathan J. Evans, Laura Fontanesi, Catherine Manning, Cynthia Huang-Pollock, Dora Matzke, Andrew Heathcote, Jörg Rieskamp, Maarten Speekenbrink, Michael J. Frank, Stefano Palminteri, Christopher G. Lucas, Jerome R. Busemeyer, Roger Ratcliff, and Jamal Amani Rad. Beyond discrete-choice options. *Trends in Cognitive Sciences*, 28(9): 857–870, September 2024. ISSN 1364-6613, 1879-307X. doi: 10.1016/j.tics.2024.07.004.

Amir Hosein Hadian Rasanan, Lukas Schumacher, Michael D. Nunez, Gabriel Weindel, and Jörg Rieskamp. Non-decision time-informed collapsing threshold diffusion model: A joint modeling framework with identifiable time-dependent parameters. *eLife*, 14, December 2025. doi: 10.7554/eLife.109730.1.

Roger Ratcliff. A theory of memory retrieval. *Psychological Review*, 85(2):59–108, 1978. doi: 10.1037/0033-295X.85.2.59.

Roger Ratcliff and Gail McKoon. The Diffusion Decision Model: Theory and Data for Two-Choice Decision Tasks. *Neural computation*, 20(4):873–922, April 2008. ISSN 0899-7667. doi: 10.1162/neco.2008.12-06-420.

Roger Ratcliff, Pablo Gomez, and Gail McKoon. A Diffusion Model Account of the Lexical Decision Task. *Psychological review*, 111(1):159–182, January 2004. ISSN 0033-295X. doi: 10.1037/0033-295X.111.1.159.

Arik Reuter, Tim GJ Rudner, Vincent Fortuin, and David Rügamer. Can transformers learn full bayesian inference in context? In *International Conference on Machine Learning*, pages 51531–51582. PMLR, 2025.

Milena Rmus, Ti-Fen Pan, Liyu Xia, and Anne GE Collins. Artificial neural networks for model identification and parameter estimation in computational cognitive models. *PLOS Computational Biology*, 20(5):e1012119, 2024.

Daniel J Schad, Michael Betancourt, and Shravan Vasishth. Toward a principled bayesian workflow in cognitive science. *Psychological methods*, 26(1):103, 2021.

Simon B Schaefer, Stefan T Radev, Jan Göttmann, and Anna-Lena Schubert. Amortized Bayesian Workflow for Modeling Congruency Effects Using the Diffusion Model for Conflict Tasks, August 2025.

Florian Scholten, Lukas Schumacher, and Paul Kelber. Brunswik's fundamental principle explained: A diffusion lens model of vicarious functioning. *Psychonomic Bulletin & Review*, 33(3):61, February 2026. doi: 10.3758/s13423-025-02764-9.

Cornelius Schröder and Jakob H Macke. Simultaneous identification of models and parameters of scientific simulators. *arXiv preprint arXiv:2305.15174*, 2023.

Lukas Schumacher, Paul-Christian Bürkner, Andreas Voss, Ullrich Köthe, and Stefan T Radev. Neural superstatistics for bayesian estimation of dynamic cognitive models. *Scientific Reports*, 13 (1):13778, 2023.

Lukas Schumacher, Martin Schnuerch, Andreas Voss, and Stefan T. Radev. Validation and Comparison of Non-stationary Cognitive Models: A Diffusion Model Application. *Computational Brain & Behavior*, October 2024. doi: 10.1007/s42113-024-00218-4.

Philip L. Smith. Diffusion theory of decision making in continuous report. *Psychological Review*, 123(4):425–451, 2016. doi: 10.1037/rev0000023.

Philip L. Smith and Roger Ratcliff. *Diffusion Process Models of Decision Making: Volume 1: Fundamental Processes*. Cambridge University Press, November 2025. ISBN 978-1-009-65269-8.

Philip L. Smith, Saam Saber, Elaine A. Corbett, and Simon D. Lilburn. Modeling continuous outcome color decisions with the circular diffusion model: Metric and categorical properties. *Psychological Review*, 127(4):562–590, July 2020. doi: 10.1037/rev0000185.

Konstantina Sokratous, Anderson K Fitch, and Peter D Kvam. How to ask twenty questions and win: Machine learning tools for assessing preferences from small samples of willingness-to-pay prices. *Journal of choice modelling*, 48:100418, 2023.

Niek Stevenson, Michelle C. Donzallaz, Reilly J. Innes, Birte U. Forstmann, Dora Matzke, and Andrew Heathcote. Bayesian hierarchical cognitive modeling with the EMC2 package. *Behavior Research Methods*, 58(1):35, January 2026. ISSN 1554-3528. doi: 10.3758/s13428-025-02869-y.

Luke Strickland, Shayne Loft, Roger W. Remington, and Andrew Heathcote. Racing to remember: A theory of decision control in event-based prospective memory. *Psychological Review*, 125(6): 851–887, November 2018. ISSN 1939-1471, 0033-295X. doi: 10.1037/rev0000113.

Sean Talts, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman. Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*, 2018.

Gabriel Tillman, Trish Van Zandt, and Gordon D. Logan. Sequential sampling models without random between-trial variability: The racing diffusion model of speeded decision making. *Psychonomic Bulletin & Review*, 27(5):911–936, October 2020. doi: 10.3758/s13423-020-01719-6.

Mischa von Krause, Stefan T Radev, and Andreas Voss. Mental speed is high until age 60 as revealed by analysis of over a million participants. *Nature human behaviour*, 6(5):700–708, 2022.

Andreas Voss, Klaus Rothermund, and Jochen Voss. Interpreting the parameters of the diffusion model: An empirical validation. *Memory & Cognition*, 32(7):1206–1220, October 2004. doi: 10.3758/BF03196893.

Andreas Voss, Veronika Lerche, Ulf Mertens, and Jochen Voss. Sequential sampling models with variable boundaries and non-normal noise: A comparison of six models. *Psychonomic bulletin & review*, 26(3):813–832, 2019.

George Whittle, Juliusz Ziomek, Jacob Rawling, and Maike A Osborne. Distribution transformers: Fast approximate bayesian inference with on-the-fly prior adaptation. *arXiv preprint arXiv:2502.02463*, 2025.

Eva Marie Wieschen, Andreas Voss, and Stefan Radev. Jumping to conclusion? a lévy flight model of decision making. *The Quantitative Methods for Psychology*, 16(2):120–132, 2020.

Jonas Wildberger, Maximilian Dax, Simon Buchholz, Stephen Green, Jakob H Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. *Advances in Neural Information Processing Systems*, 36:16837–16864, 2023.

Mike Wu, Kristy Choi, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6404–6412, 2020.

Yufei Wu, Stefan T Radev, and Francis Tuerlinckx. Testing and improving the robustness of amortized bayesian inference for cognitive models. *Psychological Methods*, 2026.

Bram Zandbelt, Braden A. Purcell, Thomas J. Palmeri, Gordon D. Logan, and Jeffrey D. Schall. Response times from ensembles of accumulators. *Proceedings of the National Academy of Sciences*, 111(7):2848–2853, February 2014. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1310577111.

A. Summary of notations

A summary of all notations presented in the main manuscript is shown in Table 3.

Table 3. Table of notations

Category	Symbol	Meaning
Amortization scopes	$\mathcal{C}$	Model class (root; e.g. sequential sampling models)
	$\mathcal{F}$	Model family (shared computational structure)
	$\mathcal{M}$	Model instance / observation model (fully specified)
	$\mathcal{D}$	Design configuration over the intrinsic parameters
Generative objects	$\theta_{\mathcal{F}}$	Intrinsic parameters of family $\mathcal{F}$
	$\mathbf{B}$	Coefficient matrix, $\mathbf{B} \in \mathbb{R}^{R \times D_{\mathcal{F}}}$ (inference target)
	$\mathbf{M}$	Binary parameter mask, $\mathbf{M} \in \{0,1\}^{R \times D_{\mathcal{F}}}$; entries $M_{i,j}$
	$\mathbf{X}$	Design matrix, $\mathbf{X} \in \mathbb{R}^{N \times R}$
	Θ	Realized parameter matrix, $\Theta = g(\mathbf{X}(\mathbf{M} \odot \mathbf{B})) \in \mathbb{R}^{N \times D_{\mathcal{F}}}$
	$\mathbf{Y}$	Model observables, $\mathbf{Y} \in \mathbb{R}^{N \times O_{\mathcal{F}}}$
	RNG	Source of stochastic variation in the observation model
Dimensions and indices	N	Number of trials (observations) per dataset
	$D_{\mathcal{F}}$	Number of intrinsic parameters in family $\mathcal{F}$
	$O_{\mathcal{F}}$	Number of observable dimensions in family $\mathcal{F}$
	I	Number of regressors
	R	Number of design-matrix columns, $R = I\,(C_{\max} - 1) + 1$
	$R_{\max}$	Maximum column capacity (zero-padding target)
	$C_{\max}$	Maximum number of categories per regressor
	i, j	Row (regressor/design column) and column (parameter) indices of $\mathbf{B}, \mathbf{M}$
Functions and parameters	$\mathcal{G} = \{g_j\}$	Per-parameter link functions; g denotes column-wise application
	$\beta_{i,j}$	Regression weight (entry of $\mathbf{B}$); $\beta_{0,j}$ is the intercept
	x_c	Covariate c
Flow matching and inference	$\mathcal{A}$	Conditioning set, $\mathcal{A} = (\mathbf{Y}, \mathbf{X}, \mathbf{M}, \mathcal{F})$
	$\hat{u}$	Learned conditional velocity field, $\hat{u}(\mathbf{Z}_t; \mathcal{A}, t)$
	$\mathbf{Z}_t$	Flow state at time t, $\mathbf{Z}_t = (1-t)\mathbf{B} + t\mathbf{Z}_1$
	$\mathbf{Z}_0$	Target endpoint, $\mathbf{Z}_0 \equiv \mathbf{B}$
	$\mathbf{Z}_1$	Gaussian latent, $\mathbf{Z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
	t	Flow time, $t \in [0,1]$
	ω_t	Optional time-dependent loss weight
	$\mathcal{L}_{\mathrm{FM}}$	Conditional flow matching loss
	$q(\mathbf{B} \mid \mathcal{A}, \hat{u})$	Approximate posterior induced by $\hat{u}$
Architecture	$\mathbf{S}$	Encoder summary representations, $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_L)$
	$\mathbf{s}_\ell$	Summary vector, $\mathbf{s}_\ell \in \mathbb{R}^S$
	L	Number of summary tokens (encoder seeds)
	S	Summary-representation dimension
	$B_{i,j}$	Decoder token for coefficient cell (i,j)
	f	Integer index of family $\mathcal{F}$ (model-family embedding)

B. Sequential sampling models and priors

B.1. Cognitive model formulations

Diffusion decision model (DDM) The DDM describes binary choices as a process of noisy evidence accumulation toward one of two decision boundaries. Let $x(t)$ denote the accumulated evidence at time t. The evolution of the decision variable follows the stochastic differential equation

$$dx(t) = \vartheta\, dt + s\, dW_t, \tag{4}$$

where ϑ denotes the drift rate, reflecting the average rate of evidence accumulation, s is the diffusion coefficient, and W_t denotes a Wiener process. The process starts at an initial state $x(0) = z\, a$, where the relative start point $z \in (-1, 1)$ controls the initial bias between the two absorbing boundaries at $-a$ and $+a$. A value of $z = 0$ corresponds to an unbiased start, whereas positive (negative) values bias the process toward the upper (lower) boundary.

A decision is made when the diffusion trajectory first reaches one of the boundaries. Let T_{dec} denote the corresponding first-passage time. The observed decision D is determined by the boundary reached:

$$D = \begin{cases} 1 & \text{if } x(T_{\text{dec}}) = +a, \\ 0 & \text{if } x(T_{\text{dec}}) = -a. \end{cases} \tag{5}$$

The observed response time is modeled as the sum of the decision time and a non-decision component,

$$RT = T_{\text{dec}} + \tau, \tag{6}$$

where τ represents non-decision time capturing processes such as stimulus encoding and motor execution.

Racing diffusion model (RDM) The RDM generalizes evidence accumulation to multiple alternatives by assuming independent accumulators that race toward their respective decision thresholds. For K alternatives, each option k is associated with a diffusion process

$$dx_k(t) = \vartheta_k\, dt + s\, dW_{k,t}, \tag{7}$$

where ϑ_k denotes the drift rate of accumulator k and $W_{k,t}$ are independent Wiener processes. Each process starts at $x_k(0) = z_k$ and evolves toward a threshold a_k.

A decision occurs when the first accumulator reaches its threshold. Let T_k denote the first-passage time of accumulator k. The chosen alternative corresponds to the winning accumulator

$$k^* = \arg\min_k T_k. \tag{8}$$

Response time is given by the winning finishing time plus the non-decision component τ.

In this paper, we employ a simplified version of the RDM. First, the model is restricted to two alternatives ($K = 2$). Second, both accumulators start at zero,

$$x_0(0) = x_1(0) = 0, \tag{9}$$

and share a common decision threshold,

$$a_0 = a_1 = a. \tag{10}$$

Drift rates are parameterized using response coding rather than stimulus coding. That is, drift rates are specified directly for each accumulator without reference to stimulus correctness. Let ϑ

denote the base drift rate for accumulator 0, and let ϑ_Δ denote the drift difference between the two accumulators. The drift rates are defined as

$$\vartheta_0 = \vartheta, \qquad \vartheta_1 = \vartheta + \vartheta_\Delta. \tag{11}$$

Under this parameterization, positive values of ϑ_Δ favor response 1, whereas negative values favor response 0.

Circular diffusion model (CDM) The CDM extends the diffusion framework to continuous response spaces, such as directional decisions. Evidence accumulation occurs in two dimensions:

$$d\mathbf{x}(t) = \begin{pmatrix} dx_1(t) \\ dx_2(t) \end{pmatrix} = \boldsymbol{\mu}\, dt + s\, d\mathbf{W}_t, \tag{12}$$

where $\boldsymbol{\mu}$ is a two-dimensional drift vector and $\mathbf{W}_t$ is a two-dimensional Wiener process.

Rather than parameterizing the drift directly in Cartesian coordinates, we express it in polar coordinates using a drift magnitude ϑ and a drift angle ϑ_θ. The Cartesian drift components are then obtained via

$$\vartheta_x = \vartheta \cos(\vartheta_\theta), \qquad \vartheta_y = \vartheta \sin(\vartheta_\theta). \tag{13}$$

The decision space is bounded by a circular absorbing boundary with radius a. A response is generated when the diffusion trajectory first reaches this boundary. The angular position of the boundary crossing determines the response direction, while the hitting time determines the decision time.

In this paper, the model is formulated using error coding. Specifically, the observed response variable corresponds to the angular deviation between the true stimulus direction and the participant's response. Consequently, the response variable is centered at zero, corresponding to unbiased responses. Under this parameterization, the drift angle ϑ_θ reflects a systematic error bias: values different from zero indicate that responses tend to deviate from the correct direction. The drift magnitude ϑ controls the speed of evidence accumulation and is therefore directly comparable to the drift rate parameter in the other diffusion models. Again, response time is determined by the first-passage time plus the non-decision component τ.

Shared model assumptions The following assumptions were applied to all three models. First, diffusion noise was fixed to unit variance ($s = 1$), which is a standard identifiability constraint in diffusion models.

Second, we allowed for across-trial variability in both the drift rate and the non-decision time. Specifically, the drift rate on trial i was assumed to vary according to

$$\vartheta_i \sim \mathcal{N}(\vartheta, s_\vartheta), \tag{14}$$

where ϑ denotes the mean drift rate and s_ϑ controls across-trial variability in drift. Similarly, non-decision time was allowed to vary across trials according to

$$\tau_i = \tau + U(0, s_\tau), \tag{15}$$

where τ denotes the minimum non-decision time and s_τ determines the range of the uniform variability component.

The following priors were used for simulating the respective models. All priors are Gaussian distributions defined in an unconstrained latent space. Before simulation, latent parameter draws are transformed using link functions to enforce the appropriate parameter constraints.

Diffusion decision model (DDM) priors

$$\begin{aligned} \vartheta &\sim \mathcal{N}(1.0, 0.8), & a &\sim \mathcal{N}(0.5, 0.5), \\ z &\sim \mathcal{N}(0.0, 0.8), & \tau &\sim \mathcal{N}(-1.2, 0.5), \\ s_\vartheta &\sim \mathcal{N}(-1.2, 1.0), & s_\tau &\sim \mathcal{N}(-1.5, 0.7) \end{aligned} \tag{16}$$

Racing diffusion model (RDM) priors

$$\begin{aligned} \vartheta &\sim \mathcal{N}(0.6, 0.5), & \vartheta_\Delta &\sim \mathcal{N}(0.6, 0.5), \\ a &\sim \mathcal{N}(0.25, 0.5), & \tau &\sim \mathcal{N}(-1.2, 0.5), \\ s_\vartheta &\sim \mathcal{N}(-1.0, 0.6), & s_\tau &\sim \mathcal{N}(-1.5, 0.7) \end{aligned} \tag{17}$$

Circular diffusion model (CDM) priors :

$$\begin{aligned} \vartheta &\sim \mathcal{N}(1.0, 0.8), & \vartheta_\theta &\sim \mathcal{N}(0.0, 0.5), \\ a &\sim \mathcal{N}(0.5, 0.5), & \tau &\sim \mathcal{N}(-1.2, 0.5), \\ s_\vartheta &\sim \mathcal{N}(-1.0, 0.6), & s_\tau &\sim \mathcal{N}(-1.5, 0.7) \end{aligned} \tag{18}$$

To ensure that parameters respect their natural constraints, all models employed differentiable link functions that transform unconstrained latent values into their respective domains prior to simulation. Across all three models, drift rates, decision thresholds, non-decision times, and across-trial drift variability were mapped using a *softplus* function,

$$f(x) = \log(1 + \exp(x)), \tag{19}$$

which guarantees positive outputs. In the RDM, the drift-difference parameter ϑ_Δ was transformed using the same softplus link. Parameters with bounded support, namely the starting point z in the DDM and the drift angle ϑ_θ in the CDM, were transformed using a *scaled sigmoid* function,

$$f(x) = \text{lower} + (\text{upper} - \text{lower})\frac{1}{1 + \exp(-x)}. \tag{20}$$

These link functions allow estimation on an unconstrained space while ensuring that simulated parameters remain within their admissible ranges.

C. Meta-simulator algorithms

Below, we present the detailed algorithm of our meta-simulator for a model family $\mathcal{F}$ (Algorithm 1) and a model class $\mathcal{C}$ (Algorithm 2).

Algorithm 1 Model Family Simulator

Require: Model family $\mathcal{F}$ with observation model $\mathcal{M}$, coefficient prior $p(\mathbf{B})$, intrinsic parameters $\theta_{\mathcal{F}}$, link g

Require: Number of observations N, number of regressors I

1: **Sample design configuration** $\mathcal{D}$: assign each regressor (intercept + main effects + interactions) to a random subset of intrinsic parameters

2: **Build parameter mask** $\mathbf{M} \in \{0,1\}^{R \times D_{\mathcal{F}}}$ from $\mathcal{D}$, where R = number of design columns, $D_{\mathcal{F}}$ = number of intrinsic parameters

3: **Build design matrix** $\mathbf{X} \in \mathbb{R}^{N \times R}$: sample continuous or discrete regressor values per trial

4: **Sample coefficient matrix** $\mathbf{B} \sim p(\mathbf{B})$, with $\mathbf{B} \in \mathbb{R}^{R \times D_{\mathcal{F}}}$

5: **Compute per-trial parameters**:

$$\Theta = g(\mathbf{X}\,(\mathbf{M} \odot \mathbf{B})) \in \mathbb{R}^{N \times D_{\mathcal{F}}}$$

6: **Simulate trial data** $\mathbf{Y} \sim \mathcal{M}(\Theta)$

7: **return** $\{\mathbf{X}, \mathbf{B}, \mathbf{M}, \Theta, \mathbf{Y}\}$

Algorithm 2 Model Class Meta-Simulator

Require: Model families $\{\mathcal{F}_f\}_{f=1}^{K}$ (constituting model class $\mathcal{C}$), sampling weights $\mathbf{w}$

Require: Global parameter space $\Omega = \bigcup_f \theta_{\mathcal{F}_f}$

Require: Batch size N_{batch}, observation range $[N_{\min}, N_{\max}]$, regressor range $[I_{\min}, I_{\max}]$

1: Sample $N \sim \text{Uniform}(N_{\min}, N_{\max})$

2: **for** $i = 1, \ldots, N_{\text{batch}}$ **do**

3: Sample model family index $f_i \sim \text{Categorical}(\mathbf{w})$

4: Sample number of regressors $I_i \sim \text{Uniform}(I_{\min}, I_{\max})$

5: Run **Algorithm 1** with $\mathcal{F}_{f_i}$, N, I_i to obtain $\{\mathbf{X}_i, \mathbf{B}_i, \mathbf{M}_i, \Theta_i, \mathbf{Y}_i\}$

6: **Lift** $\mathbf{B}_i$, $\mathbf{M}_i$ to global space Ω: insert local parameter columns at their global indices, pad with zeros for parameters absent in $\mathcal{F}_{f_i}$

7: **end for**

8: **Collate** batch: zero-pad all arrays to maximum dimensions

9: **return** $\left\{\mathbf{X}_i,\, \mathbf{B}_i^{\Omega},\, \mathbf{M}_i^{\Omega},\, \mathbf{Y}_i,\, f_i\right\}_{i=1}^{N_{\text{batch}}}$

D. Further details on CogFormer architecture

D.1. Attention blocks

A detailed breakdown of the attention blocks in CogFormer architecture can be found in Figure 8. There are three attention blocks at play: 1) self-attention block for the encoder, 2) multi-head cross-attention block as the pooling layer for the encoder, and 3) mixed attention block that consists of an undulated pattern between self-attention and cross-attention.

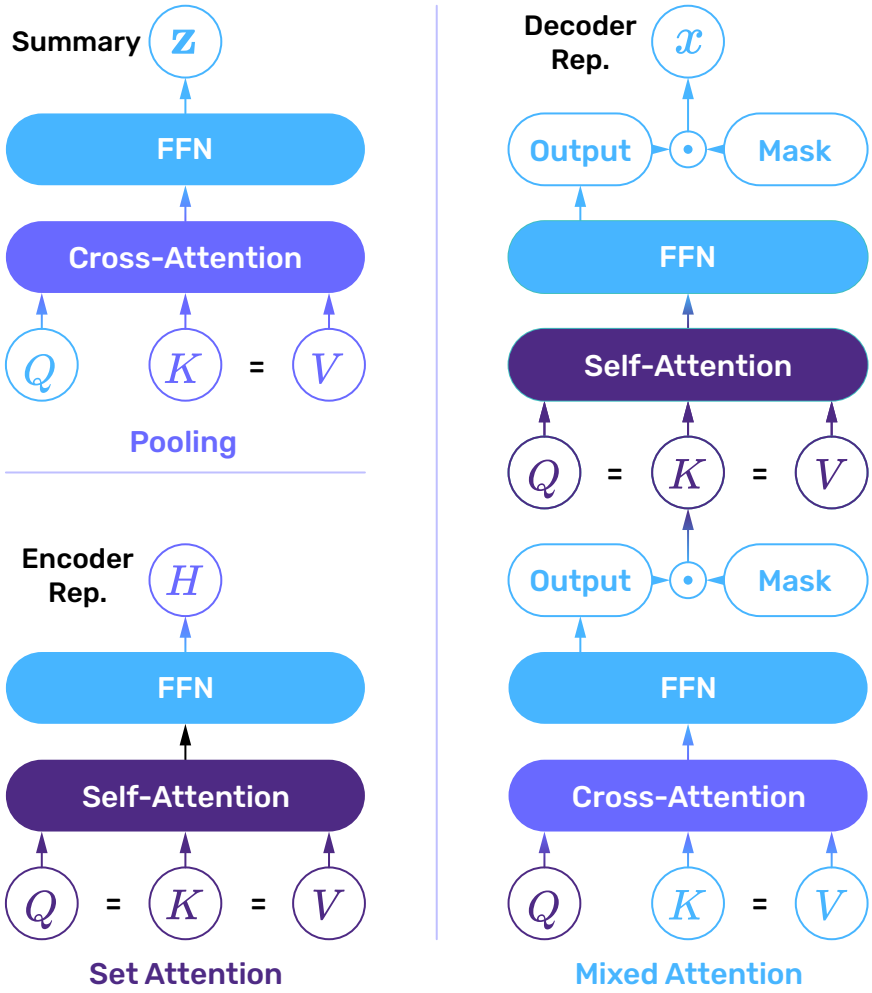

Figure 8. Breakdown of the attention blocks in the CogFormer architecture. For set attention, the input QKV matrices are identical linear layer outputs of the input data. Its output H is the encoder representation, which serves as the input KV matrices for pooling along with a learnable seed vector as the query Q. The mixed attention later takes the encoder summary $\mathbf{z}$ as query, along with the constructed decoder embeddings listed in Figure 3 as the KV matrices. The mixed-attention architecture places the cross-attention layer first, followed by the self-attention layer, and further alternates between the two. The layer-wise output is element-wise multiplied by the parameter mask M before passing to the next layer. The resulting output x serves as the decoder representation that is used to generate a velocity field v.

D.2. Hyperparameters

Table 4. CogFormer hyperparameters for the family-level models (DDM, RDM, CDM) and the model-class model. QKV dim per head = projection dim ÷ heads.

Hyperparameter	**CogFormer-$\mathcal{F}$**	**CogFormer-$\mathcal{C}$**
Layers		
Encoder layers	8	8
Decoder layers	8	8
Attention		
Heads (encoder & decoder)	8	8
Projection dim d_{model}	256	256
QKV dim per head	32	32
Embeddings		
Time embedding dim	32	32
Positional embedding dim	32	32
Seed tokens	32	32
Seed dim	64	128
Model embedding dim	–	8

E. Formal definition of evaluation metrics

Parameter recovery Given model observables $\mathbf{Y}$ and a discrepancy (loss) metric δ (e.g., RMSE), parameter recovery examines whether the trained ABI framework can recover the true data-generating parameter θ from its posterior estimates $\tilde{\theta}$:

$$\text{Recovery}(\delta) := \mathbb{E}_{(\theta,\,\mathbf{Y})\,\sim\,p(\theta,\mathbf{Y})}\left[\int \delta(\theta,\tilde{\theta})\, q(\tilde{\theta} \mid \mathbf{Y})\, \mathrm{d}\tilde{\theta}\right]. \tag{21}$$

Parameter recovery is a significant step in model verification because it informs the accuracy of parameter estimation of trained ABI frameworks (Kühmichel et al., 2026).

Simulation-based calibration (SBC) Talts et al. (2018) introduced SBC for checking the consistency of the data-averaged posterior with respect to the prior via their respective rank statistics. For a scalar parameter θ, given S simulations each with M posterior draws, we compute the rank statistic as

$$\text{rank}^{(n)} = \sum_{m=1}^{M} \mathbf{1}\left[\tilde{\theta}_m^{(n)} < \theta^{(n)}\right], \quad \tilde{\theta}_m^{(n)} \sim q(\theta \mid \mathbf{Y}^{(n)}), \quad \theta^{(n)} \sim p(\theta) \tag{22}$$

so $\text{rank}^{(n)} \in \{0, 1, \ldots, M\}$. Under a well-calibrated posterior, $\text{rank}^{(n)} \sim \text{Uniform}\{0, \ldots, M\}$. The ECDF is defined over the rank statistic:

$$\hat{F}(\text{rank}) = \frac{1}{S} \sum_{n=1}^{S} \mathbf{1}\left[\text{rank}^{(n)} \leq \text{rank}\right]. \tag{23}$$

Single-value metrics Three scalar metrics are used for our evaluation: 1) normalized root mean square error (NRMSE), 2) empirical calibration error (ECE), and 3) posterior contraction. These metrics are well-defined in Appendix A of Arruda et al. (2025).

F. Experiment benchmark designs

For our experiments, we use the design configurations listed in Table 5 as our benchmarks to evaluate all relevant metrics. Taken as a whole, these configurations cover a comprehensive set of scenarios in which intrinsic parameters are free, fixed, and regressed, and allow us to isolate individual conditions for model validation.

Table 5. Test design configurations. Each cell lists the parameters assigned to the corresponding effect type. A dash (−) indicates no parameters are assigned. ϑ_Δ abbreviates ϑ_Δ (RDM).

Configuration	**Model**	**Intercept** (**1**)	**Slope** (u_1)	**Slope** (u_2)	**Interaction** ($u_1 \times u_2$)
Intercept Only	DDM	$\vartheta, a, z, \tau, s_\vartheta, s_\tau$	–	–	–
	RDM	$\vartheta, \vartheta_\Delta, a, \tau, s_\vartheta, s_\tau$	–	–	–
	CDM	$\vartheta, \vartheta_\theta, a, \tau, s_\vartheta, s_\tau$	–	–	–
Fixed Variability	DDM	ϑ, a, z, τ	–	–	–
	RDM	$\vartheta, \vartheta_\Delta, a, \tau$	–	–	–
	CDM	$\vartheta, \vartheta_\theta, a, \tau$	–	–	–
Regressed	DDM	$\vartheta, a, z, \tau, s_\vartheta, s_\tau$	ϑ, a, z	ϑ, a, z	–
	RDM	$\vartheta, \vartheta_\Delta, a, \tau, s_\vartheta, s_\tau$	ϑ_Δ, a	ϑ_Δ, a	–
	CDM	$\vartheta, \vartheta_\theta, a, \tau, s_\vartheta, s_\tau$	ϑ, a	ϑ, a	–
Fixed + Regressed	DDM	ϑ, a, z, τ	ϑ, a, z	ϑ, a, z	–
	RDM	$\vartheta, \vartheta_\Delta, a, \tau$	ϑ_Δ, a	ϑ_Δ, a	–
	CDM	$\vartheta, \vartheta_\theta, a, \tau$	ϑ, a	ϑ, a	–
With Interaction	DDM	$\vartheta, a, z, \tau, s_\vartheta, s_\tau$	$\vartheta, a, z, \tau, s_\vartheta$	ϑ, a, z, τ	ϑ, a, z
	RDM	$\vartheta, \vartheta_\Delta, a, \tau, s_\vartheta, s_\tau$	$\vartheta, \vartheta_\Delta, a, \tau, s_\vartheta$	$\vartheta, \vartheta_\Delta, a, \tau$	$\vartheta, \vartheta_\Delta, a$
	CDM	$\vartheta, \vartheta_\theta, a, \tau, s_\vartheta, s_\tau$	$\vartheta, \vartheta_\theta, a, \tau, s_\vartheta$	$\vartheta, \vartheta_\theta, a, \tau$	$\vartheta, \vartheta_\theta, a$

G. Ablation studies

G.1. Architectural changes

We investigated CogFormer-$\mathcal{F}$'s performance under several architectural changes outlined in Table 6, with their corresponding single-value metrics listed in Table 7.

Table 6. Summary of component ablation variants and the architectural role of each ablated component.

Variant	**Component Role**
No SAB	Self-attention across parameter tokens; allows parameters to coordinate with one another during the denoising pass.
No MAB	Cross-attention from parameter tokens to the encoder summary; the primary pathway through which the decoder reads the observed data.
Baseline	Feature-wise linear modulation (FiLM) of the output projection, conditioned on the current diffusion timestep; scales and shifts the final representation before predicting velocity.
No FE	Learnable Fourier embedding (FE) of the diffusion timestep; projects the scalar t into a high-dimensional frequency space before concatenation.
With FiLM	Feature-wise linear modulation (FiLM) of the output projection, conditioned on the current diffusion timestep; scales and shifts the final representation before predicting velocity.

Table 7. Component ablation metrics across all design conditions, ordered by number of active parameters. NRMSE and Calibration Error: lower is better (↓). Posterior Contraction: higher is better (↑). Bold indicates best per metric within each condition group. †No MAB collapses Posterior Contraction across all conditions; its Cal. Error optimum is not meaningful, since the prior itself is perfectly calibrated.

Condition	Variant	NRMSE ↓	Cal. Error ↓	Contraction ↑
Fixed ($n = 4$)	Baseline	0.055	0.021	**0.966**
	No SAB	0.058	0.039	0.944
	No MAB	0.255	0.017	0.037
	No FE	**0.053**	0.023	0.965
	With FiLM	0.058	**0.017**	0.966
Intercept Only ($n = 6$)	Baseline	**0.087**	0.022	**0.845**
	No SAB	0.103	0.022	0.825
	No MAB	0.249	0.026	0.077
	No FE	0.090	**0.019**	0.820
	With FiLM	0.096	0.025	0.833
Fixed + Regressed ($n = 10$)	Baseline	0.071	0.023	0.946
	No SAB	0.077	0.018	0.925
	No MAB†	0.263	0.016	0.030
	No FE	**0.067**	0.027	**0.947**
	With FiLM	0.072	0.020	0.938
Regressed ($n = 12$)	Baseline	**0.073**	0.021	0.903
	No SAB	0.082	0.021	0.891
	No MAB	0.262	0.027	0.086
	No FE	0.076	**0.020**	**0.904**
	With FiLM	0.081	0.023	0.894
Interaction ($n = 18$)	Baseline	0.099	0.020	0.838
	No SAB	0.103	0.028	0.834
	No MAB	0.268	0.018	0.029
	No FE	**0.094**	0.025	**0.849**
	With FiLM	0.103	**0.017**	0.822

G.2. Model embedding for CogFormer-$\mathcal{C}$

To demonstrate the utility of model embedding for CogFormer-$\mathcal{C}$, we performed a comparative study with a similar architecture without model embedding. The metric results are shown in Table 8. From these results, we can observe that in the majority of designs, CogFormer-$\mathcal{C}$ can estimate better with embedding, at a small expense of calibration errors.

H. Size comparison study

We ran four versions of CogFormer-$\mathcal{F}$ with different hyperparameter sets – small (S), medium (M), large (L), extra-large (XL) – over 1000 epochs with varying number of attention layers, with the following aggregated results showing how CogFormer performs with increasing level of expressiveness (see Table 9). This result for hyperparameter sweep suggests that deeper architecture is not necessarily better, with our baseline setting (8 layers and 8 heads) performing the best among all candidate hyperparameter sets.

Table 8. Model embedding ablation for the DDM across all five design conditions. Mean metrics across all active parameters. Δ (in parentheses) is the change relative to the with-embedding baseline. NRMSE and Calibration Error: lower is better (↓). Posterior Contraction: higher is better (↑). Best result per condition per metric is **bolded**.

Design	Condition	NRMSE ↓	Cal. Error ↓	Contraction ↑
Intercept Only	Without Embedding	0.104 (+0.005)	**0.033** (-0.009)	0.794 (-0.033)
	With Embedding	**0.099**	0.042	**0.827**
Fixed	Without Embedding	0.068 (+0.008)	**0.044** (-0.019)	0.947 (-0.006)
	With Embedding	**0.060**	0.063	**0.954**
Regressed	Without Embedding	**0.082** (-0.002)	0.031 (+0.000)	0.884 (-0.002)
	With Embedding	0.084	**0.031**	**0.886**
Fixed Regressed	Without Embedding	0.079 (+0.001)	**0.025** (-0.010)	0.919 (-0.003)
	With Embedding	**0.078**	0.035	**0.922**
Interaction	Without Embedding	**0.105** (-0.002)	**0.024** (-0.002)	**0.811** (+0.006)
	With Embedding	0.107	0.026	0.806

Table 9. Mean posterior approximation metrics across all parameters and conditions for each CogFormer-$\mathcal{F}$ size variant over the number of attention layers (L) and the number of attention heads (H). NRMSE and Calibration Error: lower is better (↓). Posterior Contraction: higher is better (↑). Best result per column is **bolded**.

	S (2L, 2H)	M (4L, 4H)	**L (8L, 8H; baseline)**	XL (16L, 16H)
NRMSE ↓	0.179	0.170	**0.124**	0.127
Cal. Error ↓	0.024	0.032	**0.022**	0.045
Post. Contr. ↑	0.472	0.566	**0.746**	0.701

I. Baseline comparison study

Our baseline uses Set Transformer (Lee et al., 2019) and flow matching implemented in `BayesFlow` (Kühmichel et al., 2026), which is closely aligned with the encoder-decoder architecture of CogFormer. The choice of flow matching as inference network is justified by the most comprehensive benchmark of free-form models to date (Arruda et al., 2025), which demonstrated that flow matching with said settings achieves the best results on established SBI benchmarks (Lueckmann et al., 2021) and high-dimensional problems of genuine scientific interest.

Table 10. Mean posterior approximation metrics across all DDM parameters for each inference network. NRMSE and Calibration Error: lower is better (↓). Posterior Contraction: higher is better (↑). Best result per column is **bolded**.

	Coupling Flow	Diffusion Model	**Flow Matching**	Consistency Model
NRMSE ↓	0.128	0.131	**0.117**	0.129
Cal. Error ↓	0.020	0.025	**0.017**	0.067
Post. Contr. ↑	0.793	0.746	**0.820**	0.706

To confirm our justification, we compare the baseline against 3 other neural estimators with different inference algorithms: coupling flows, diffusion model, and consistency model. Training them over one of the benchmark cases (Interaction) for the DDM model family yields aggregated results shown in Table 10. We can observe that, across all evaluated metrics, flow matching outperforms all other inference networks.

J. Simulation budget study

To investigate how the amortization gap closes as simulation budget increases, we trained CogFormer-$\mathcal{F}$ and CogFormer-$\mathcal{C}$ under the standard training protocol (5000 epochs, 100 steps per epoch, and a batch size of 64), and evaluated the joint C2ST for each benchmark design every 1000 epochs. The result is shown in Figure 9. In general, the result suggests that as the simulation budget increases, the amortization gap closes correspondingly. The gaps are generally larger with more parameters to estimate, with regressed cases having larger gaps than non-regressed cases.

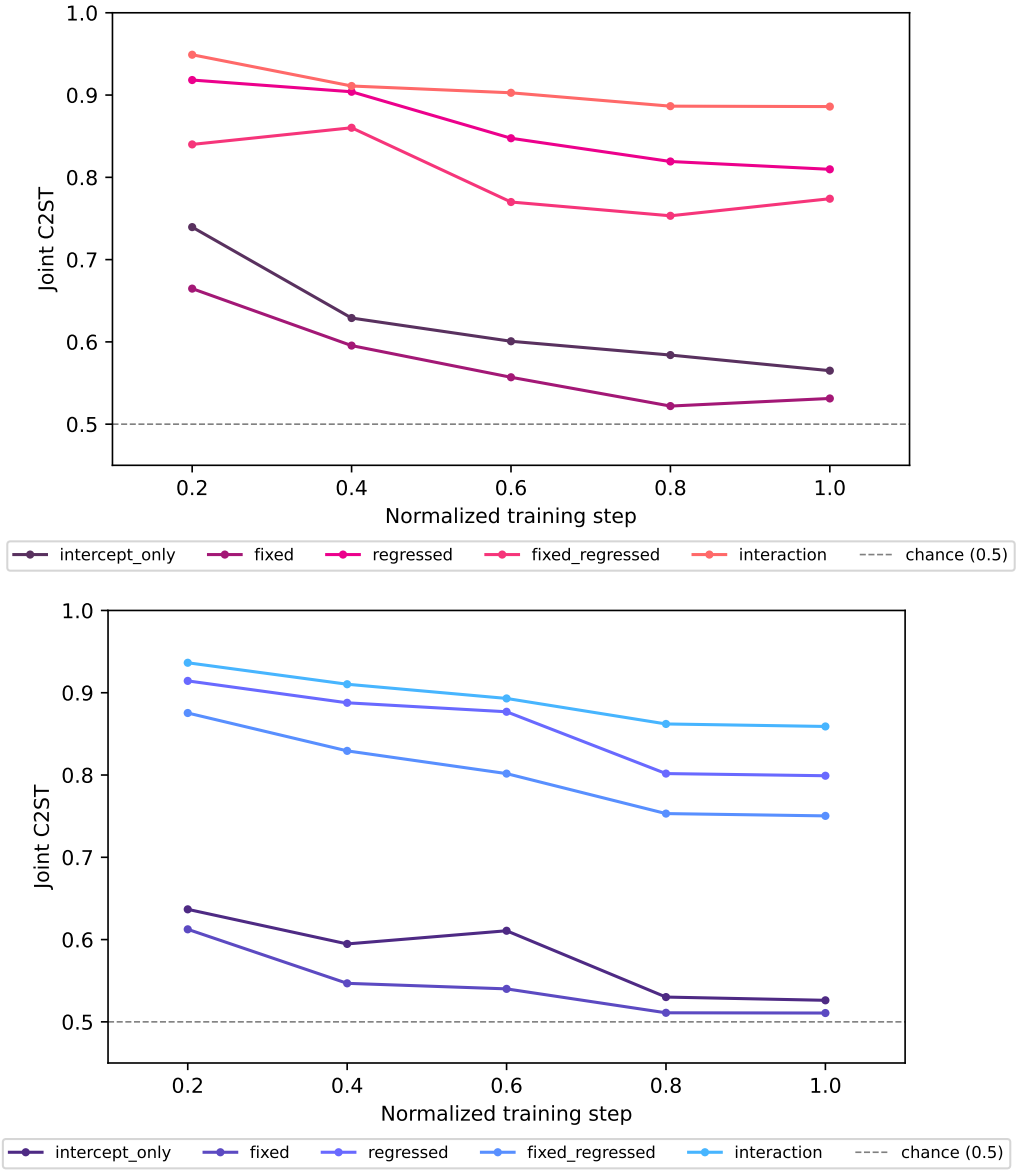

Figure 9. Joint C2ST aggregated across 10 posteriors for each benchmark design configuration per increasing simulation budget for CogFormer-$\mathcal{F}$ (*top*) and CogFormer-$\mathcal{C}$ (*bottom*). The training epochs are normalized to between 0 and 1, with 0.2 indicating Epoch 1000 and 1 indicating Epoch 5000.

K. Additional results for ensemble evaluations

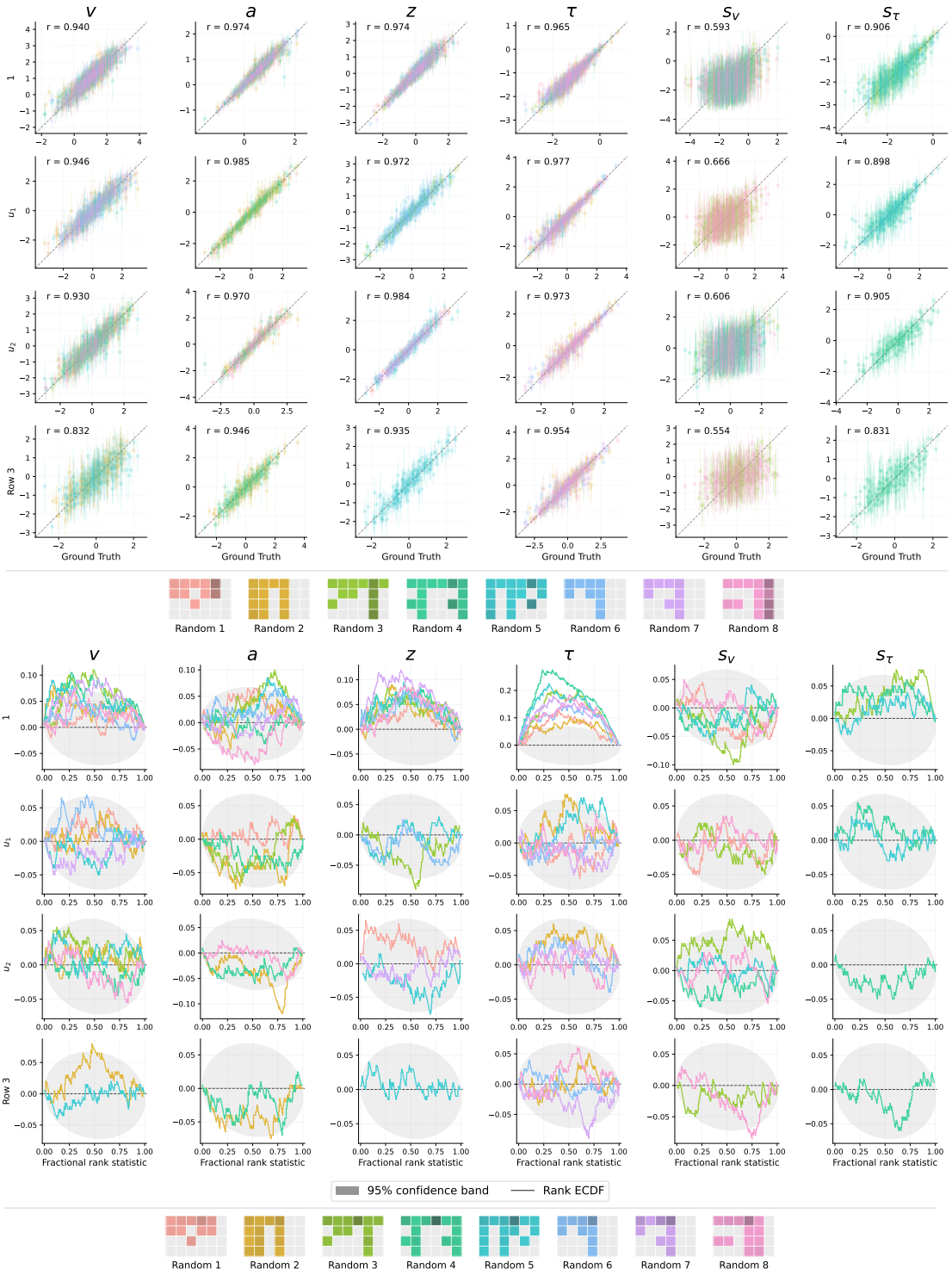

Figure 10. Random ensemble parameter recovery (*top*) and calibration ECDF (*bottom*) for meta-amortized model family for DDM. The pixel legends at the bottom indicates the parameter mask for each benchmark design configurations.

Figure 11. Ensemble parameter recoveries for CogFormer-$\mathcal{F}$ applied to the five design configurations for the RDM benchmark cases overlayed in their corresponding mapping to the shared coefficient matrix. The pixel thumbnails indicates the active estimation targets for each design configuration, with lighter shades of color depicting higher correlations r. The per-cell r-s are averaged across cases.

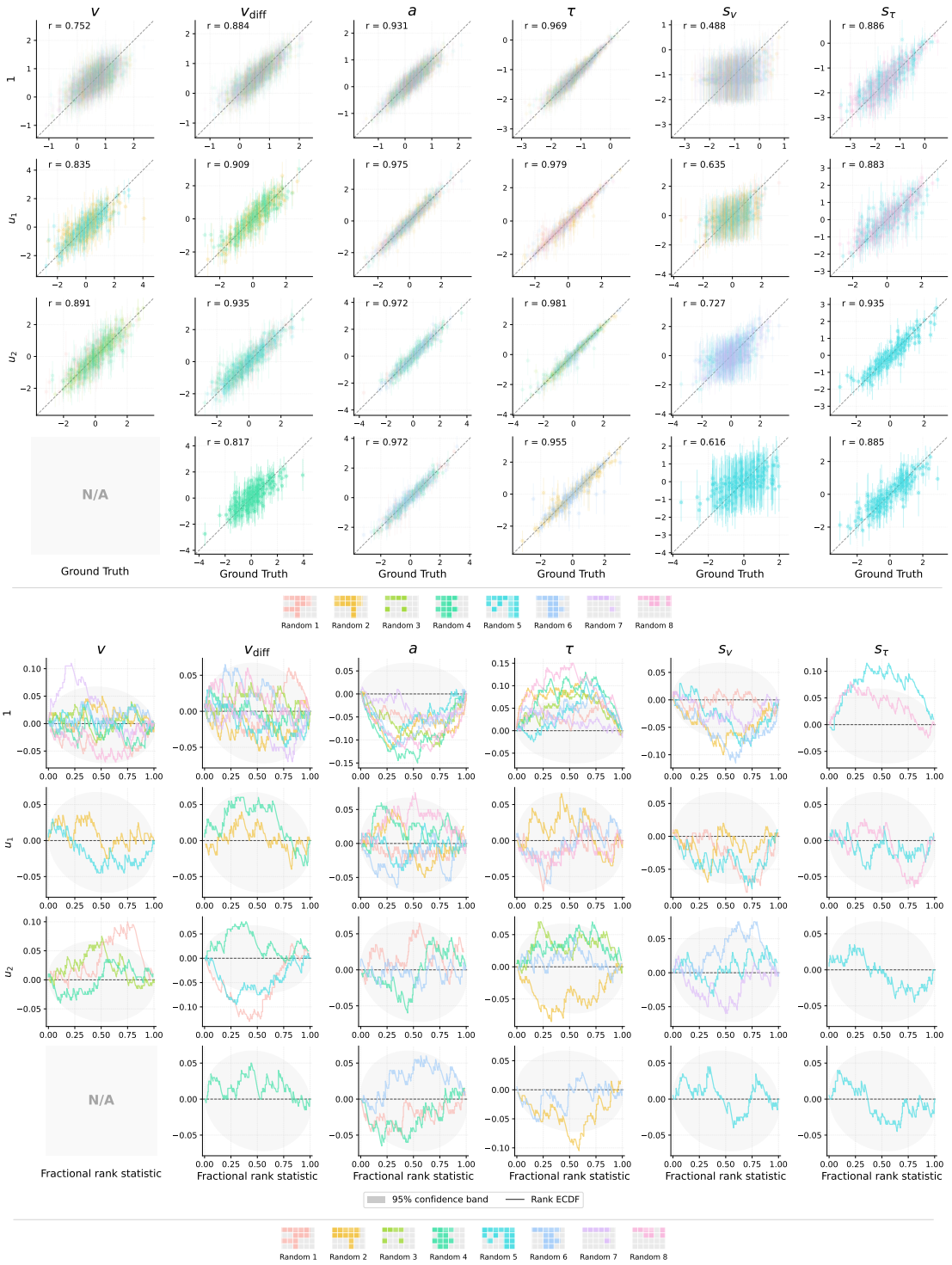

Figure 12. Random ensemble parameter recovery (*top*) and calibration ECDF (*bottom*) for meta-amortized model family for RDM. The pixel legends at the bottom indicates the parameter mask for each benchmark design configurations.

Figure 13. Ensemble parameter recoveries for CogFormer-$\mathcal{F}$ applied to the five design configurations for the CDM benchmark cases overlayed in their corresponding mapping to the shared coefficient matrix. The pixel thumbnails indicates the active estimation targets for each design configuration, with lighter shades of color depicting higher correlations r. The per-cell r-s are averaged across cases.

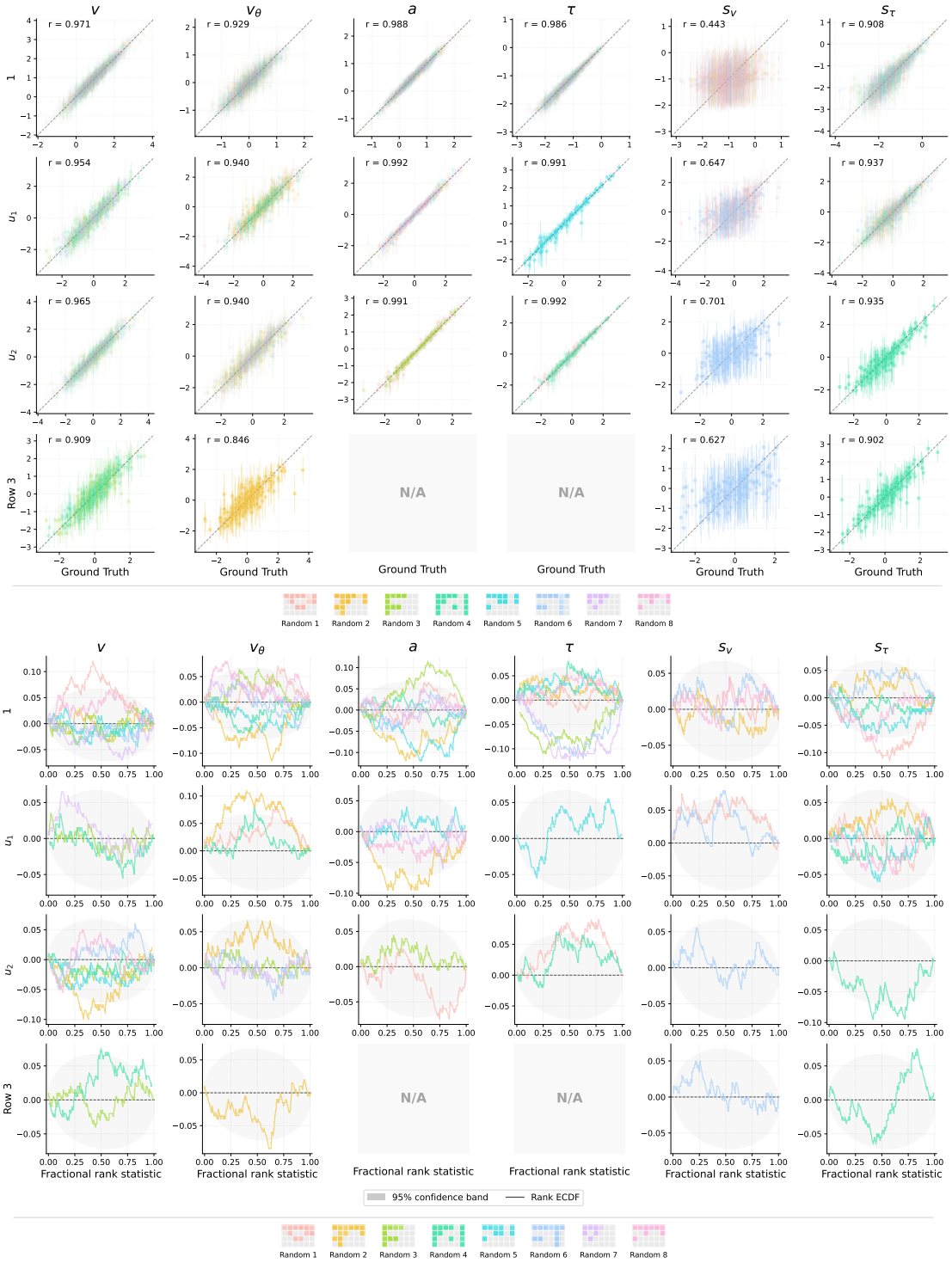

Figure 14. Random ensemble parameter recovery (*top*) and calibration ECDF (*bottom*) for meta-amortized model family for CDM. The pixel legends at the bottom indicates the parameter mask for each benchmark design configurations.

L. Additional results for benchmarking

Table 11. Corresponding frameworks, figures, and colors for the additional results for benchmarking.

Frameworks	**Figures**	**Color Scheme**
Baseline	15-19 (DDM) / 30-34 (RDM) / 45-49 (CDM)	Green
CogFormer-$\mathcal{F}$	20-24 (RDM) / 35-39 (CDM) / 50-54 (DDM)	Red
CogFormer-$\mathcal{C}$	25-29 (RDM) / 40-44 (CDM) / 55-59 (DDM)	Blue

Below, we present results for all benchmarking cases. To guide readers through these results, we represent Table 11 that outlines the corresponding figures and visual aids for the individual benchmark cases and their corresponding frameworks. All figures preserve the layout of their shared coefficient matrix $\mathbf{B}$ to demonstrate that they are structured and masked inputs to their respective frameworks.

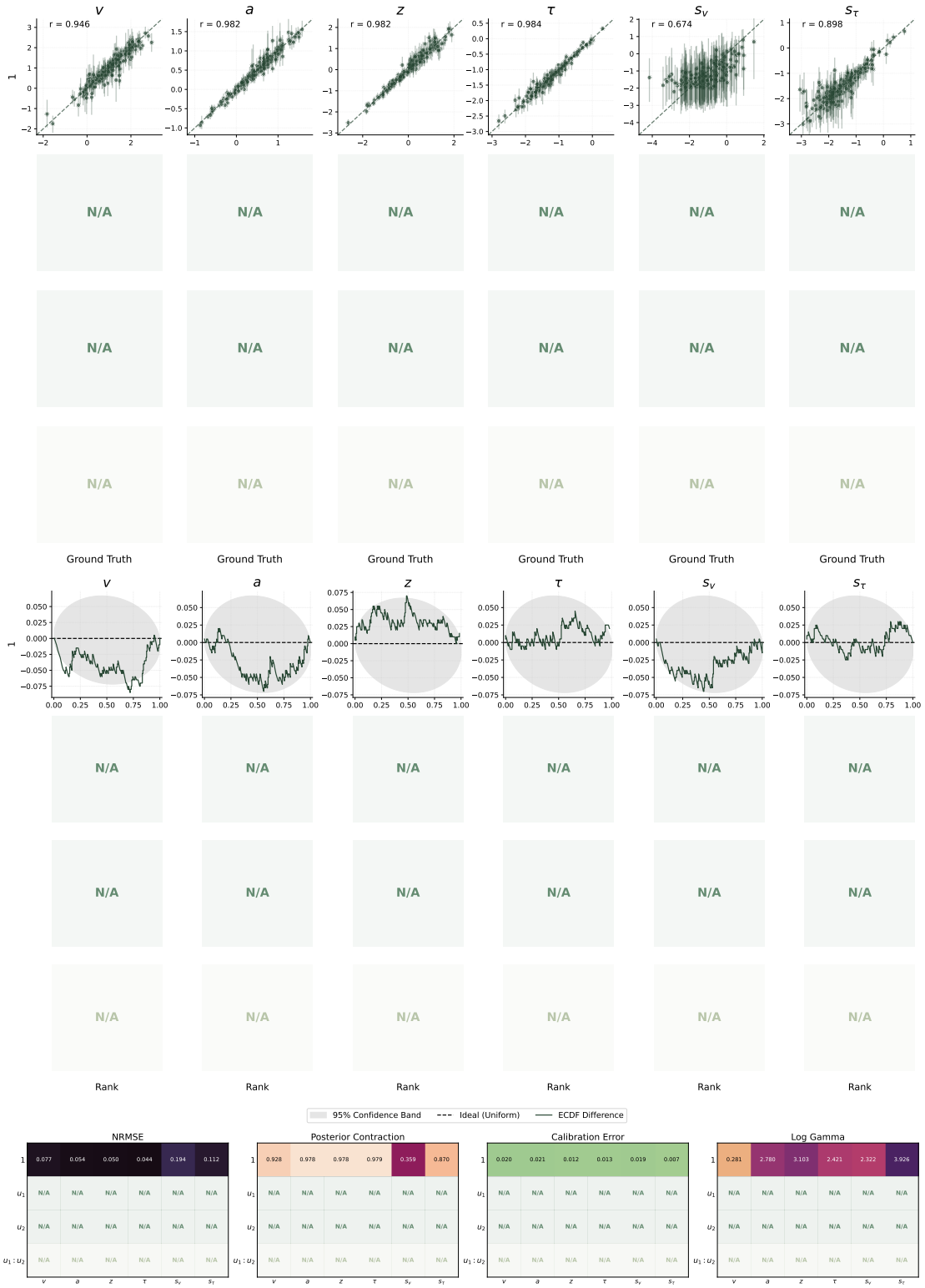

Figure 15. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM baseline (Case **intercept_only**).

Figure 16. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM baseline (Case **fixed**).

Figure 17. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM baseline (Case **regressed**).

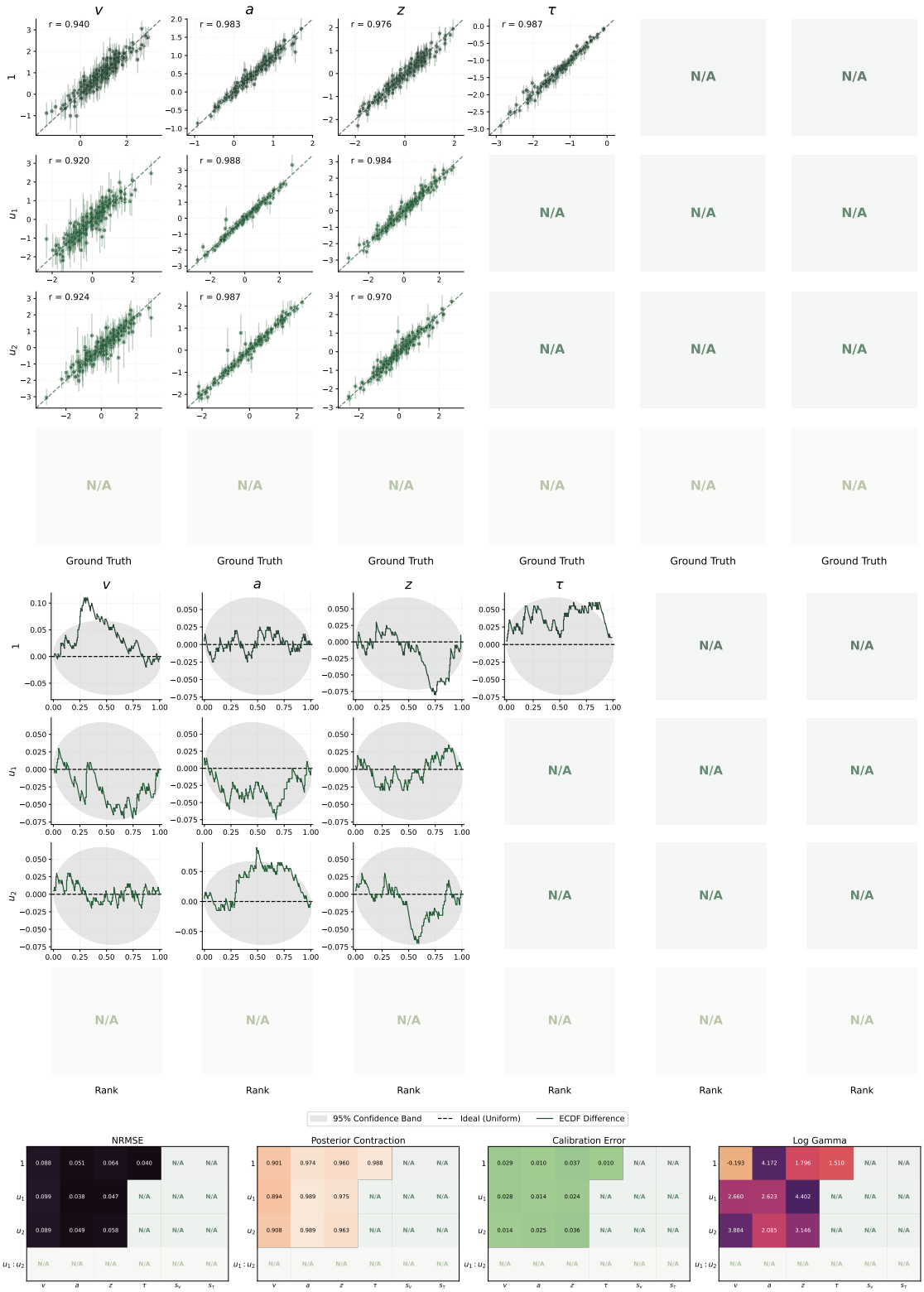

Figure 18. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM baseline (Case **fixed_regressed**).

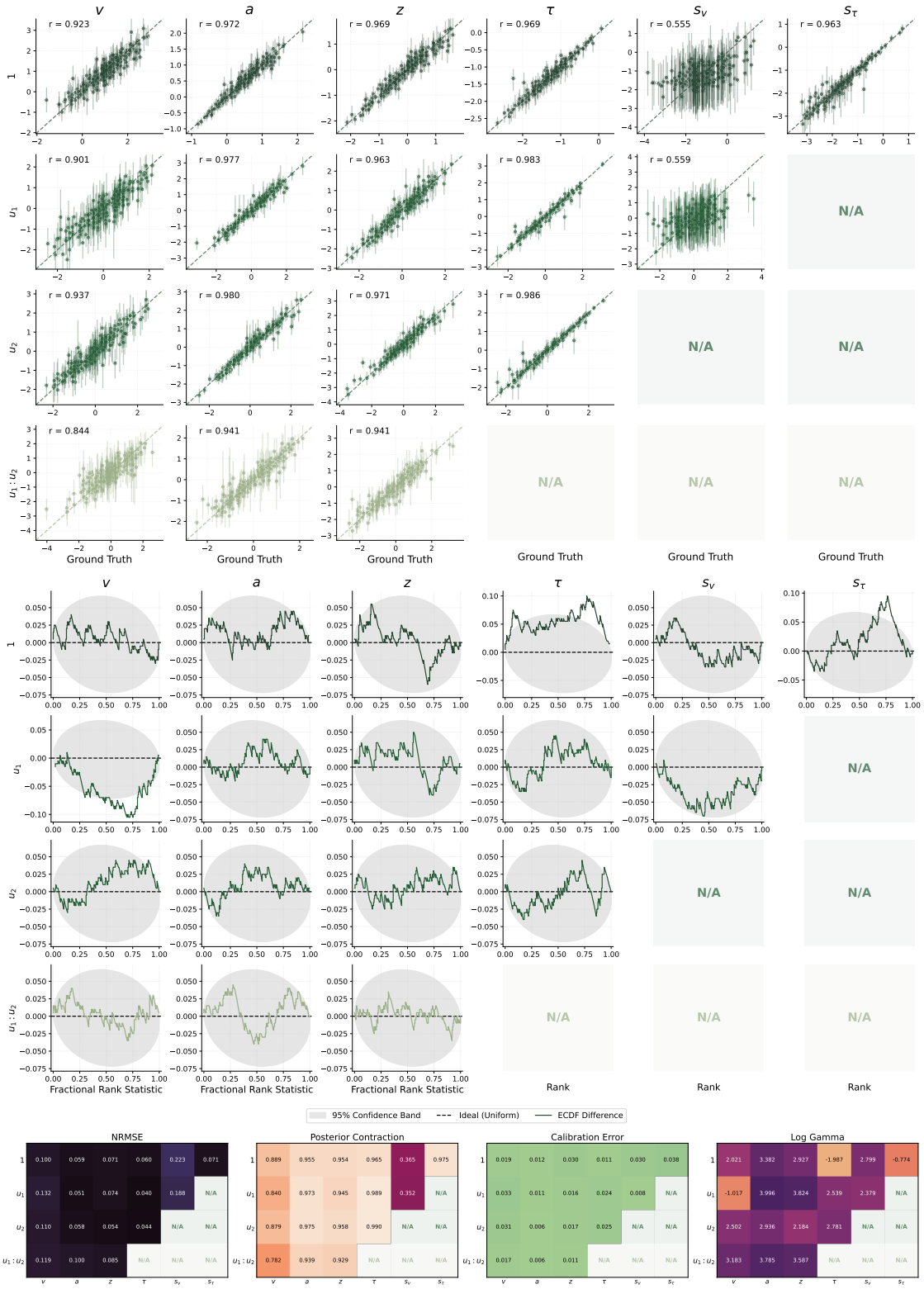

Figure 19. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM baseline (Case **interaction**).

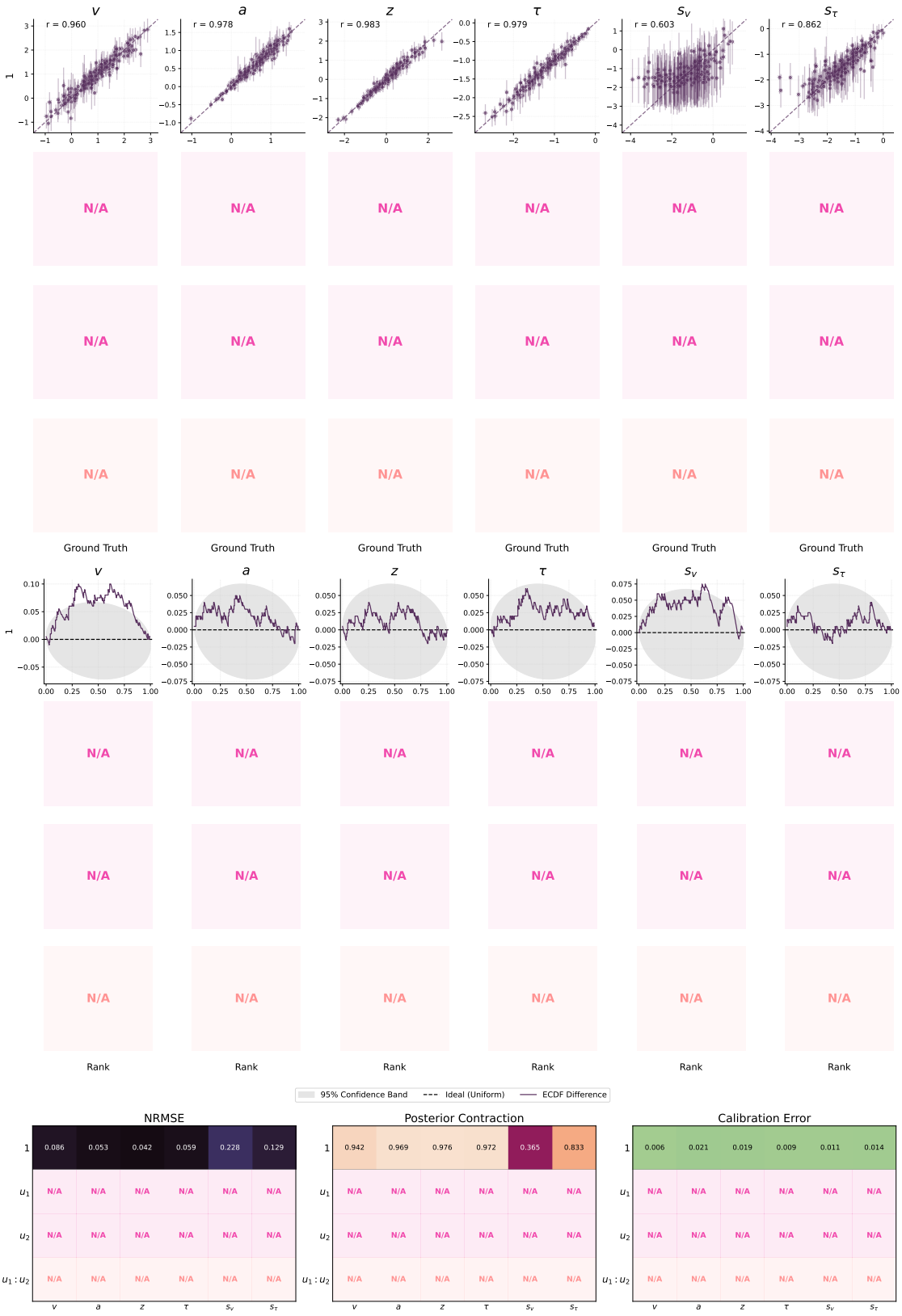

Figure 20. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM model family (Case **intercept_only**).

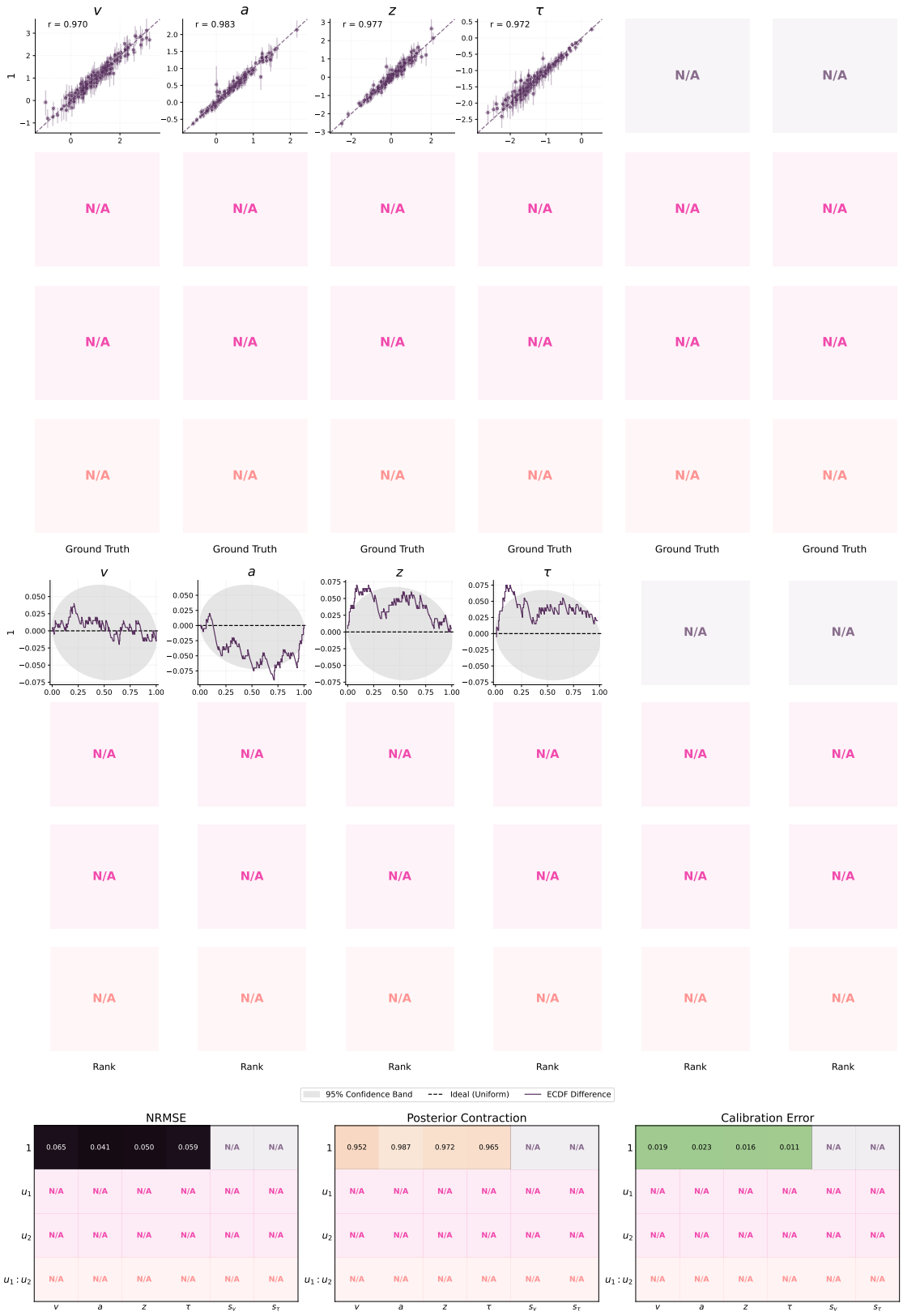

Figure 21. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM model family (Case **fixed**).

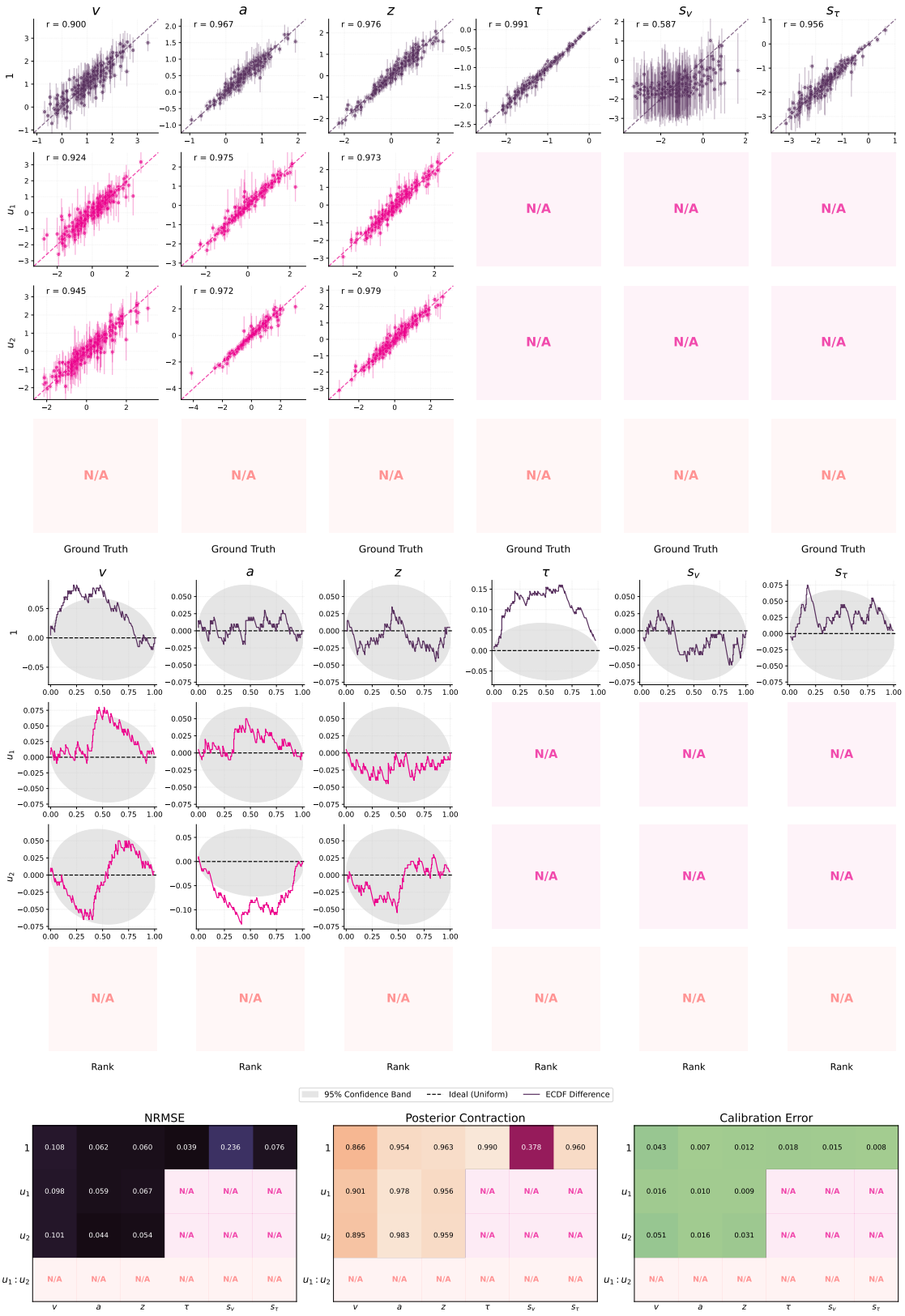

Figure 22. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM model family (Case **regressed**).

Figure 23. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM model family (Case **fixed_regressed**).

Figure 24. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for DDM model family (Case **interaction**).

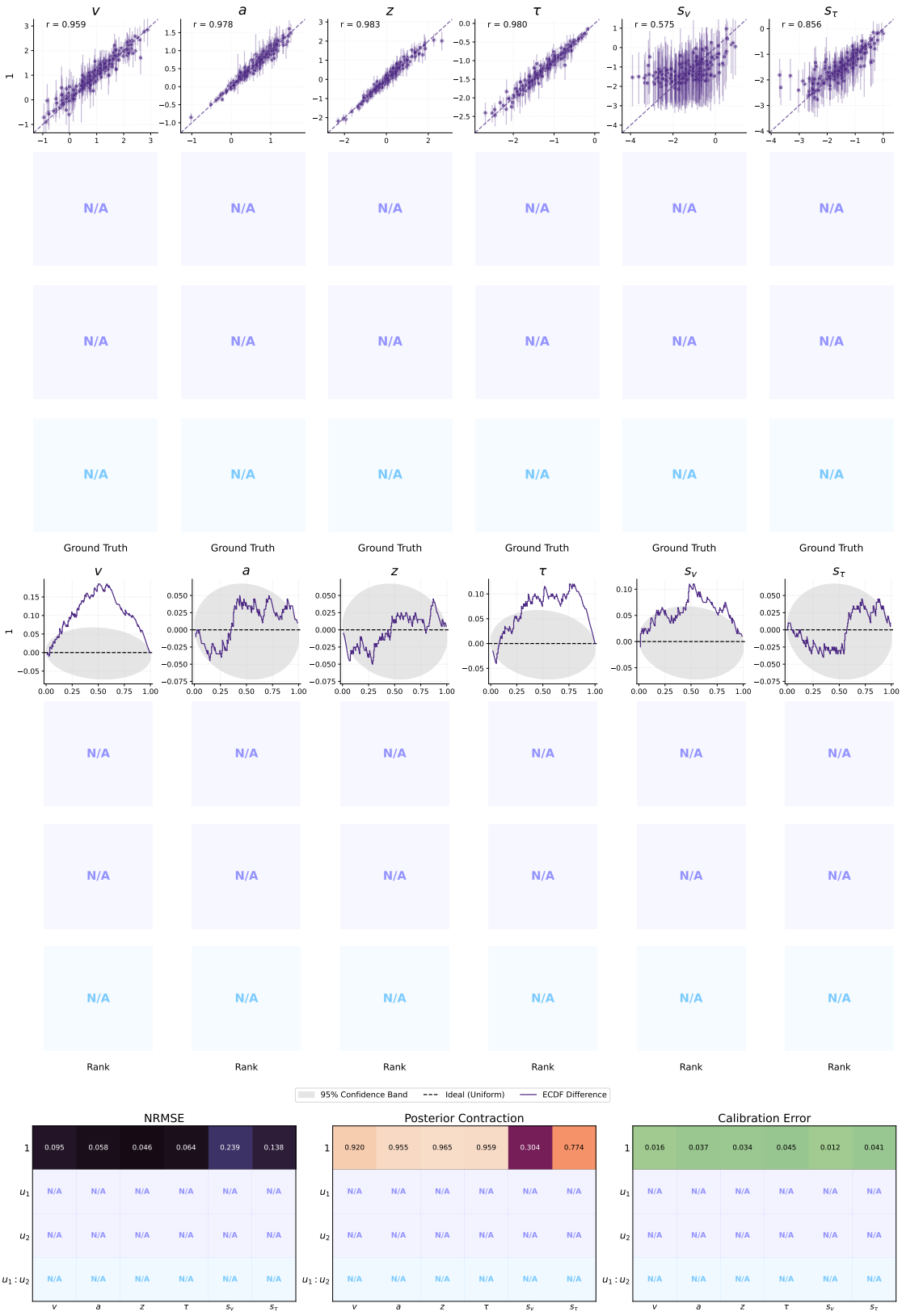

Figure 25. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for DDM model class (Case **intercept_only**).

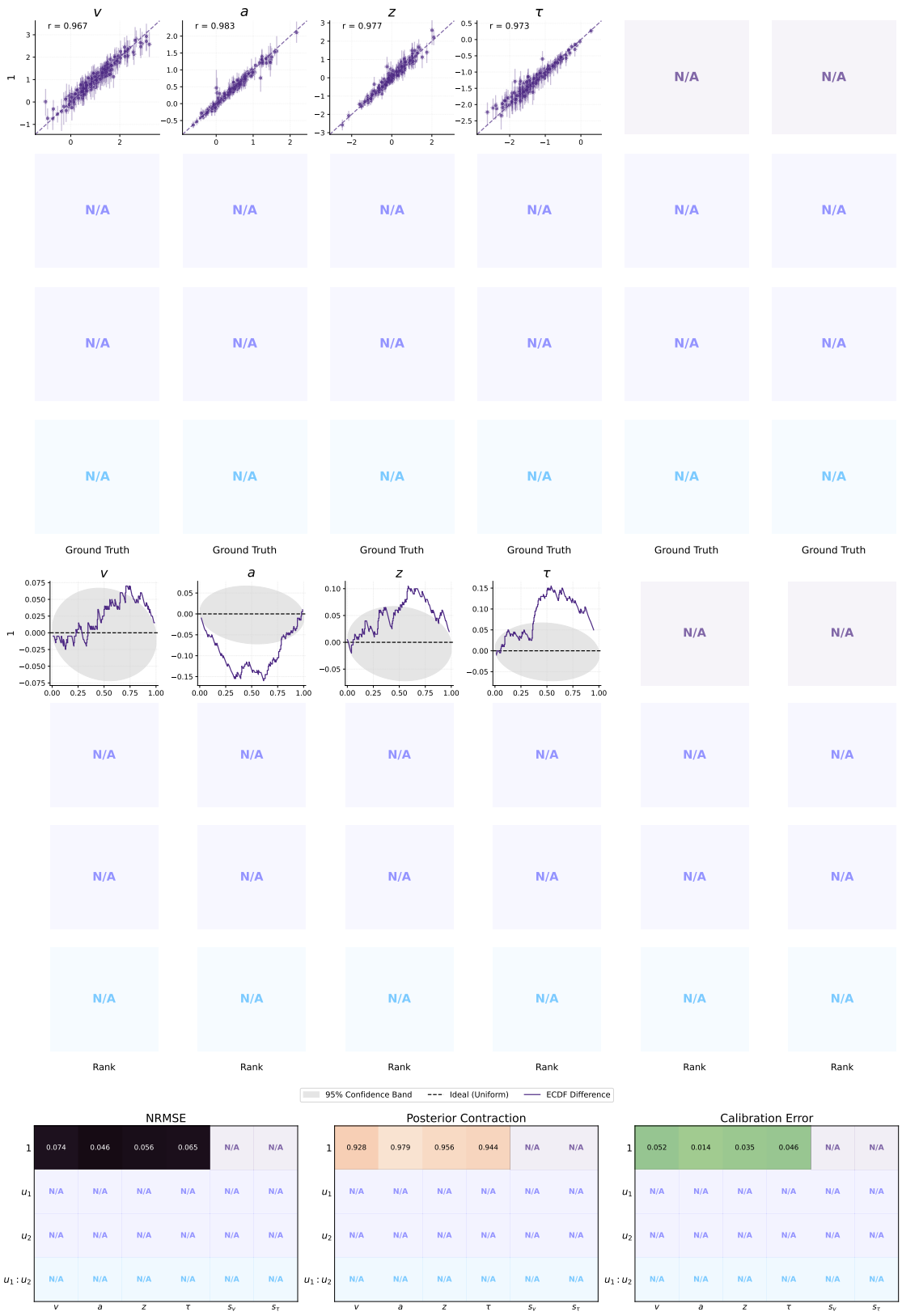

Figure 26. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for DDM model class (Case **fixed**).

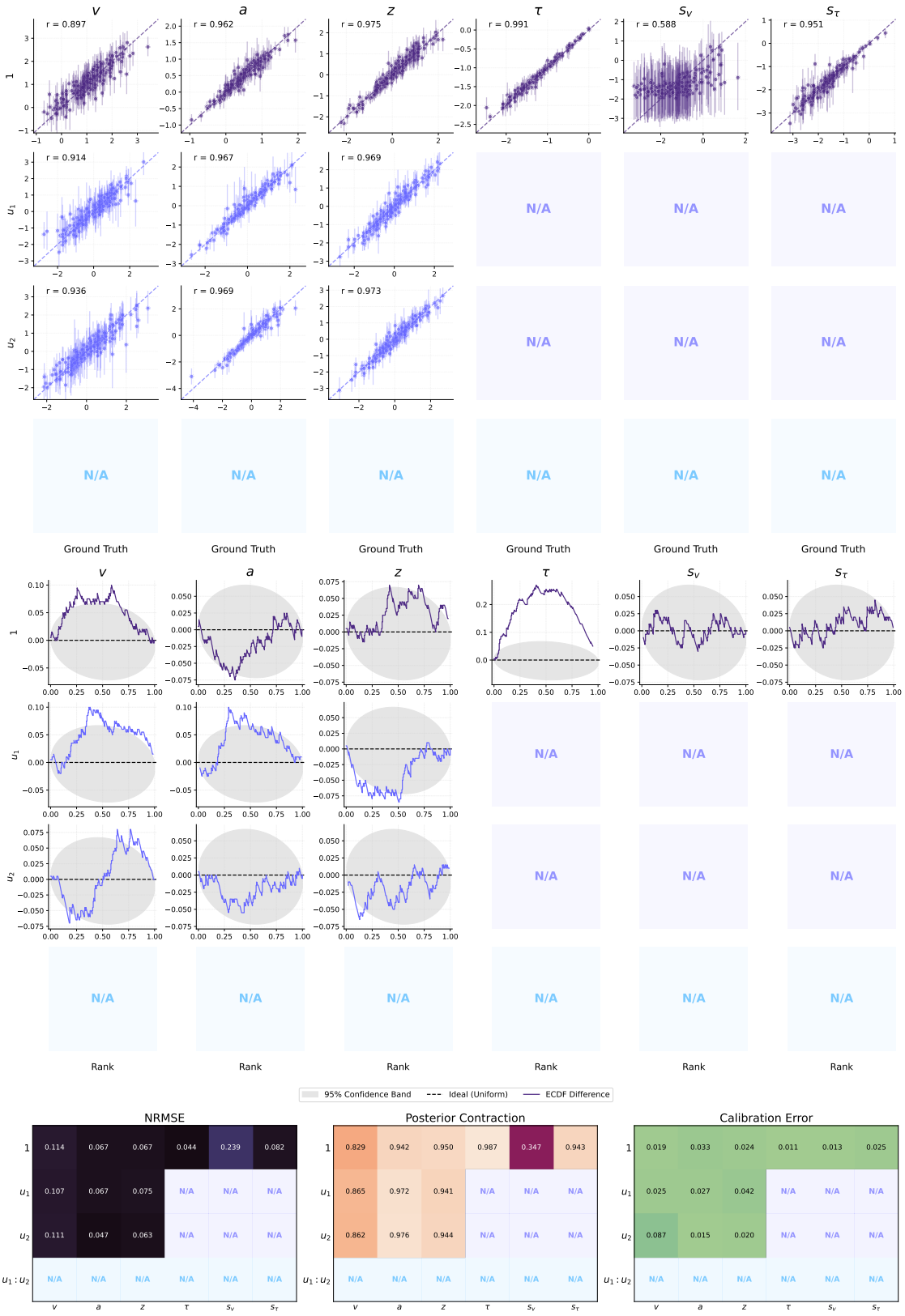

Figure 27. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for DDM model class (Case **regressed**).

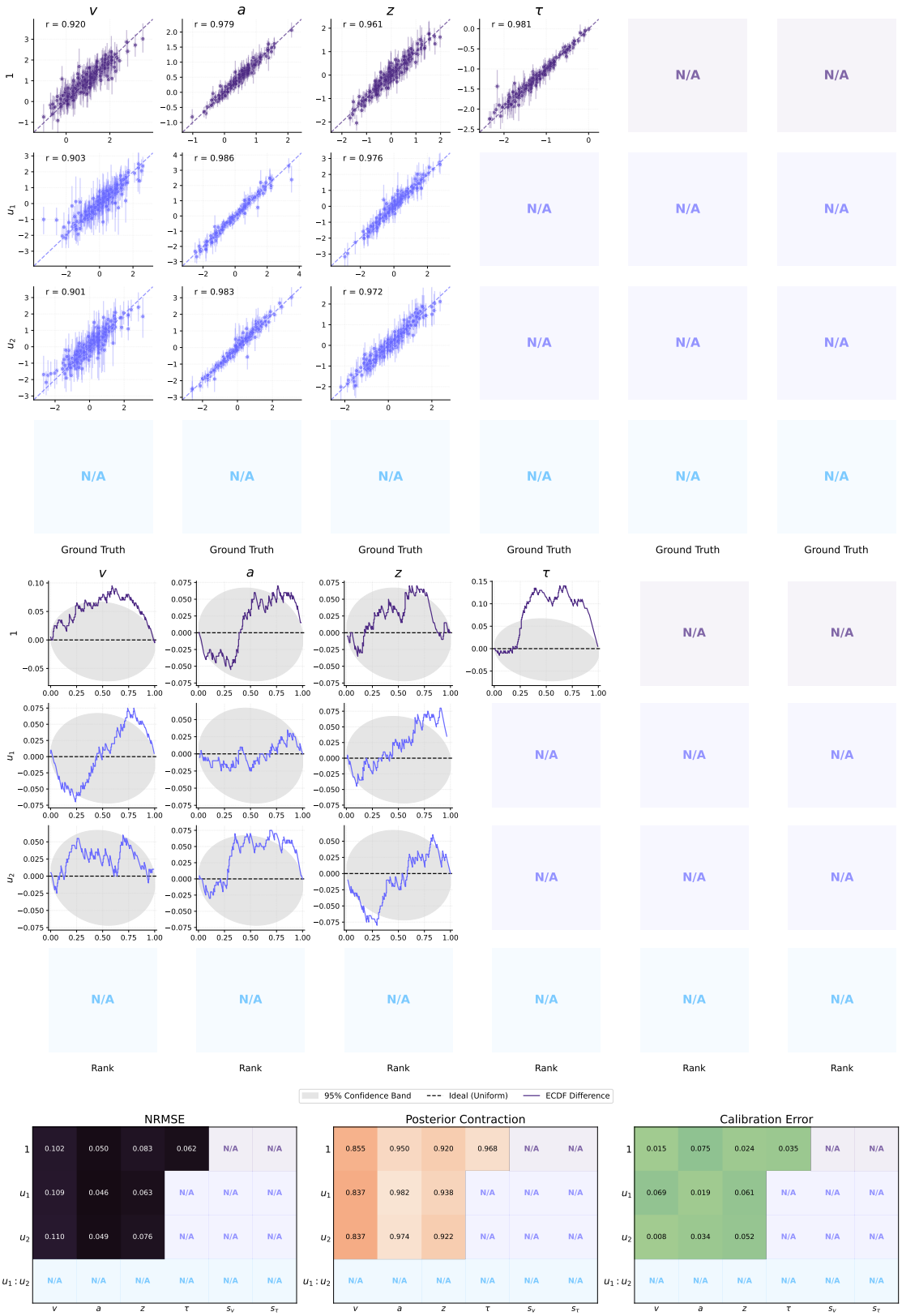

Figure 28. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for DDM model class (Case **fixed_regressed**).

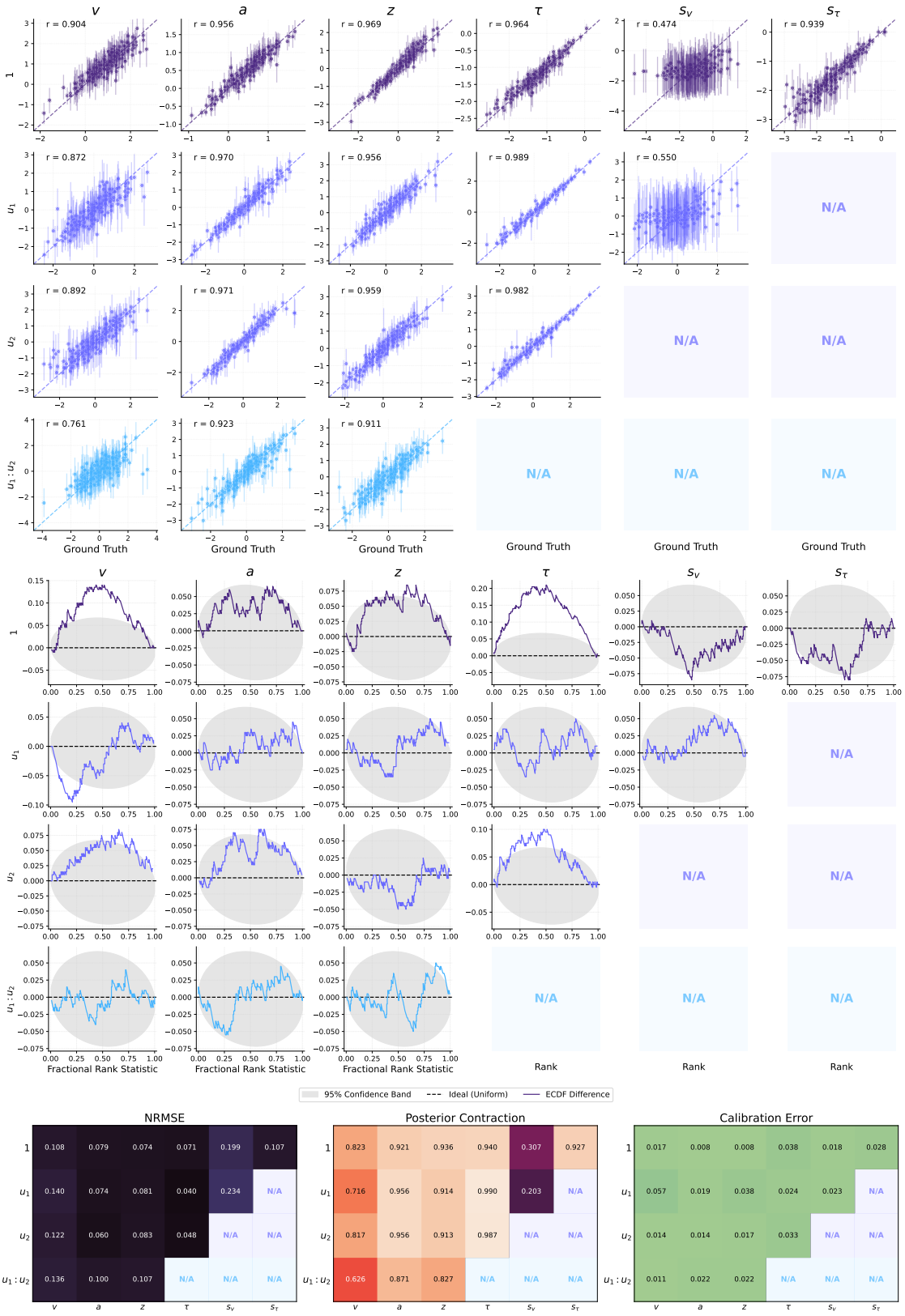

Figure 29. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for DDM model class (Case **interaction**).

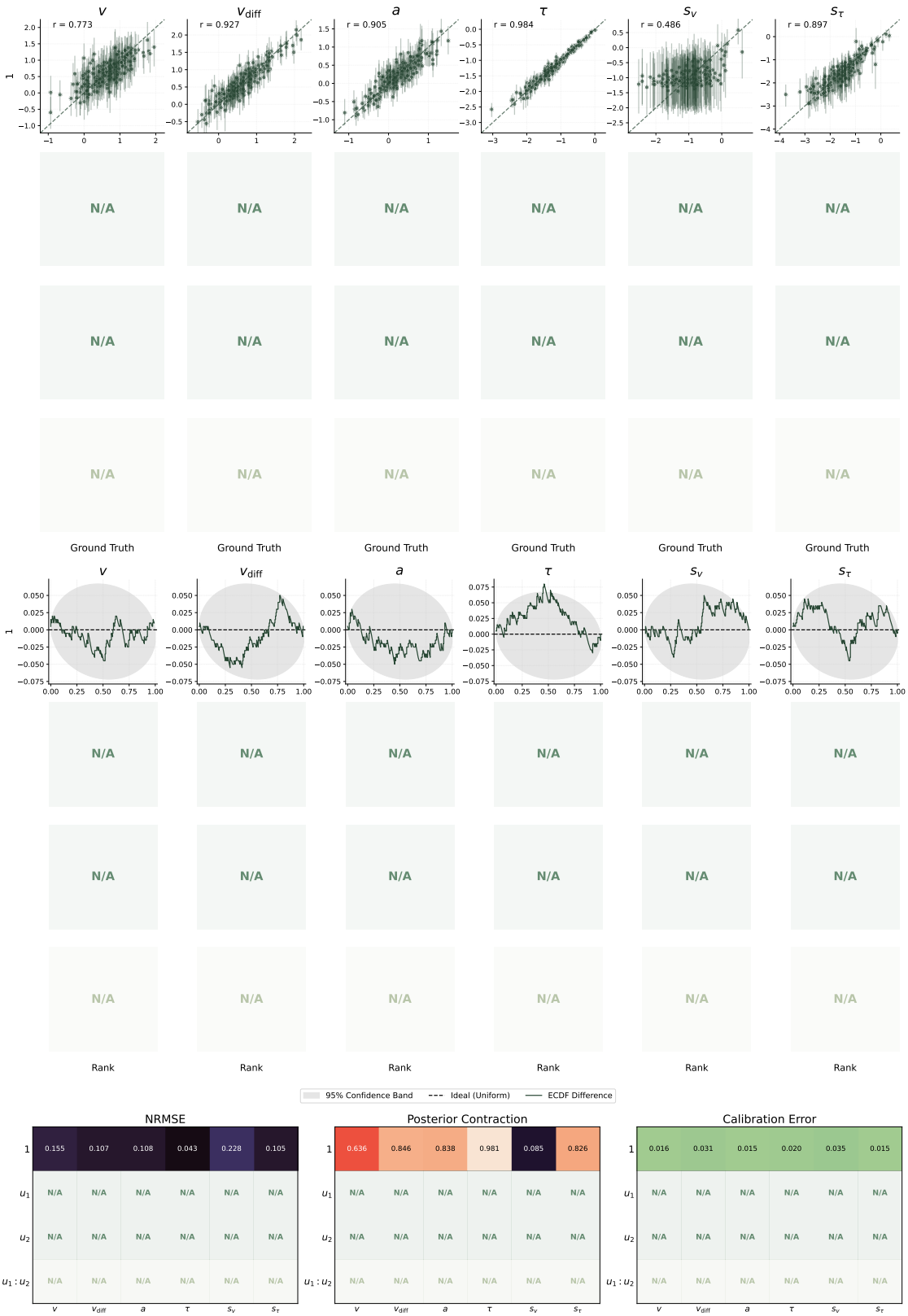

Figure 30. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM baseline (Case **intercept_only**).

Figure 31. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM baseline (Case **fixed**).

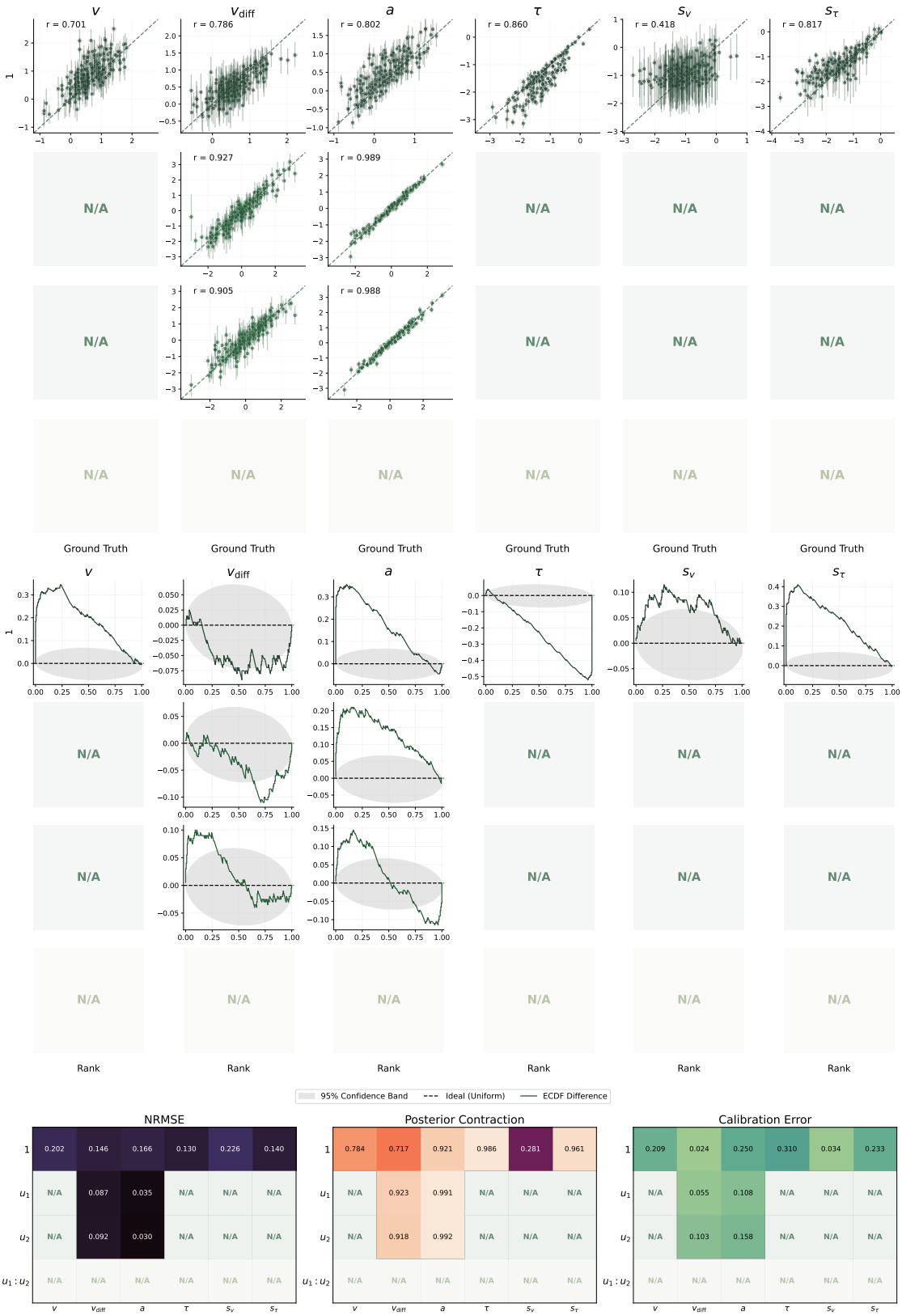

Figure 32. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM baseline (Case **regressed**).

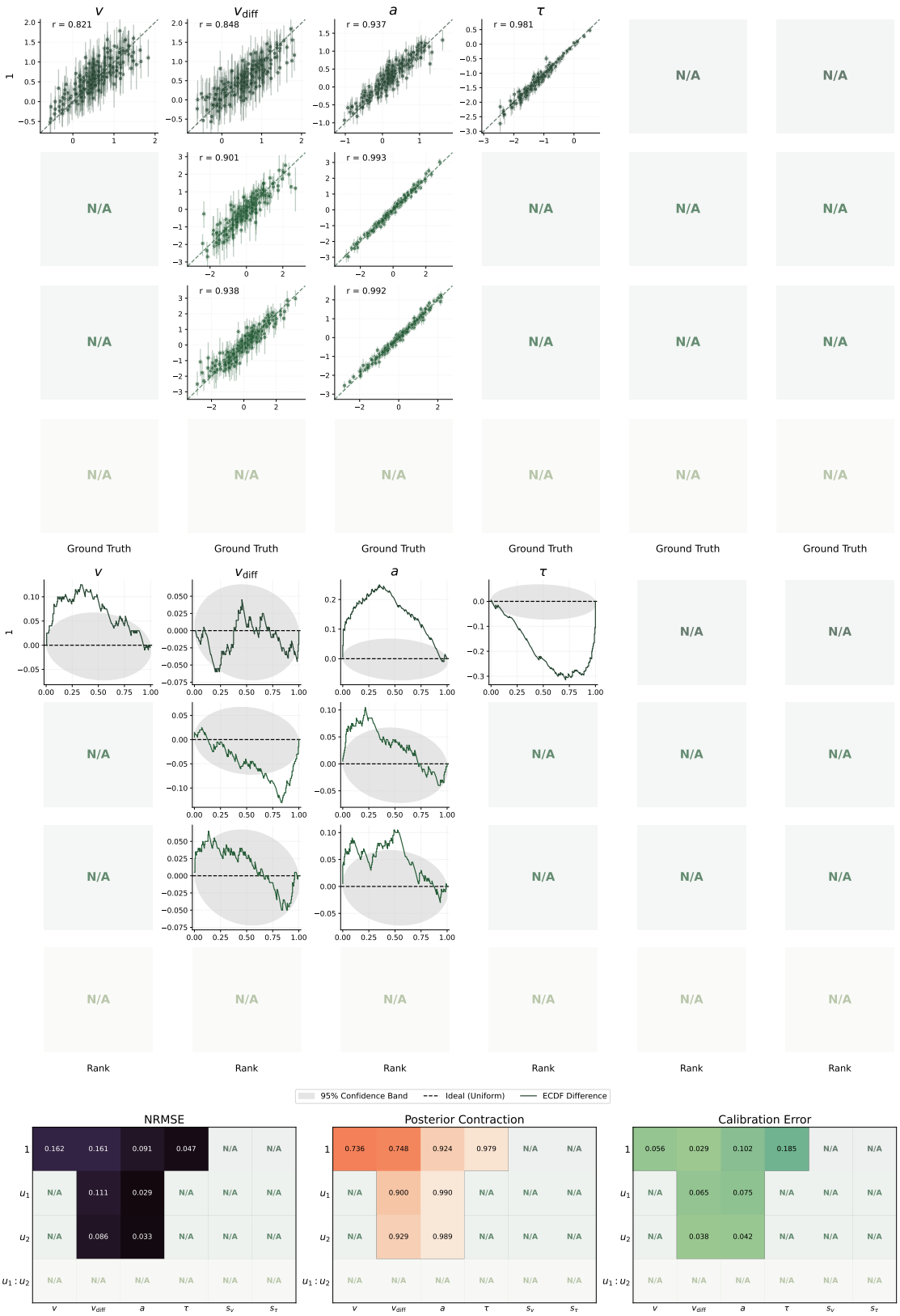

Figure 33. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM baseline (Case **fixed_regressed**).

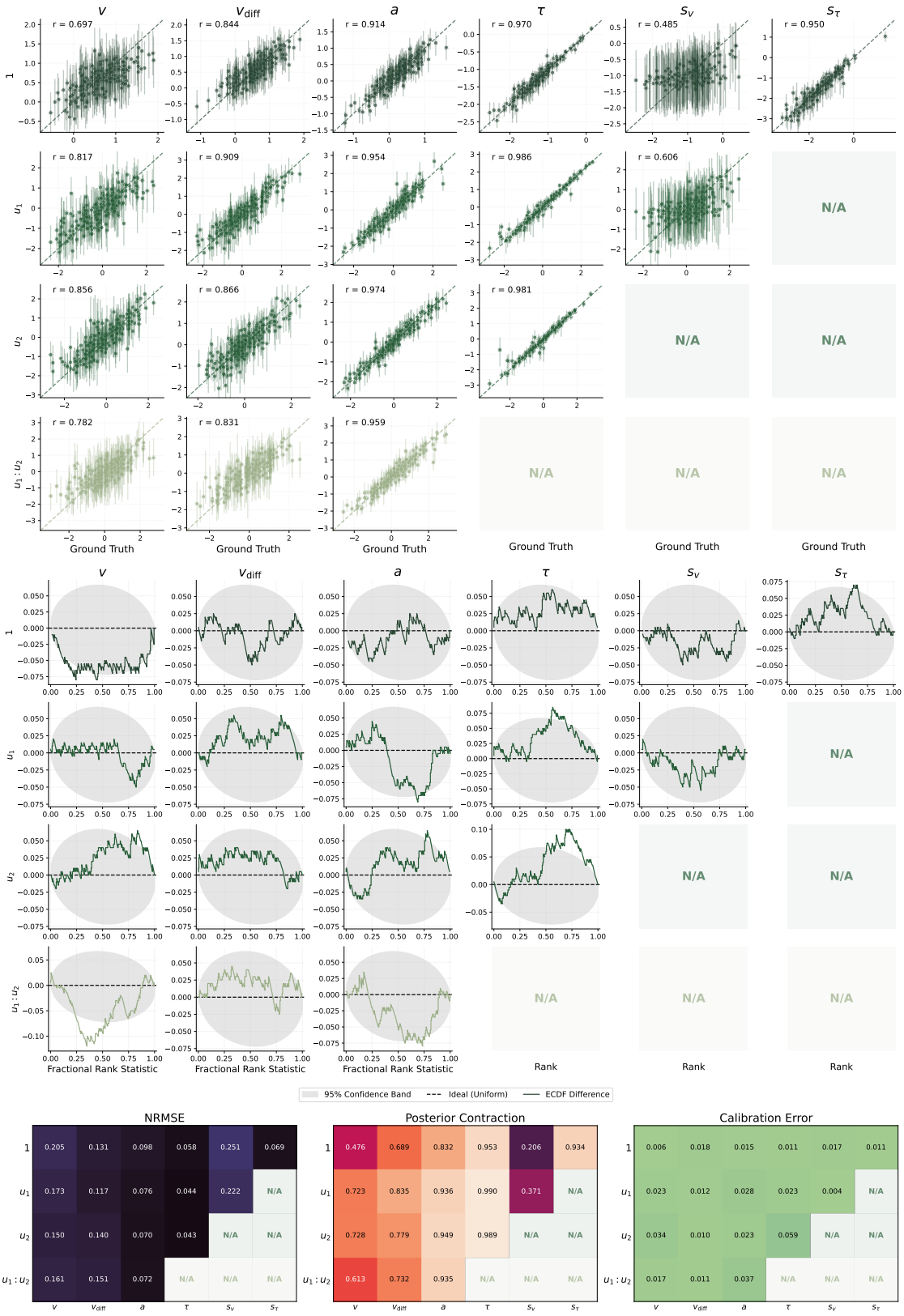

Figure 34. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM baseline (Case **interaction**).

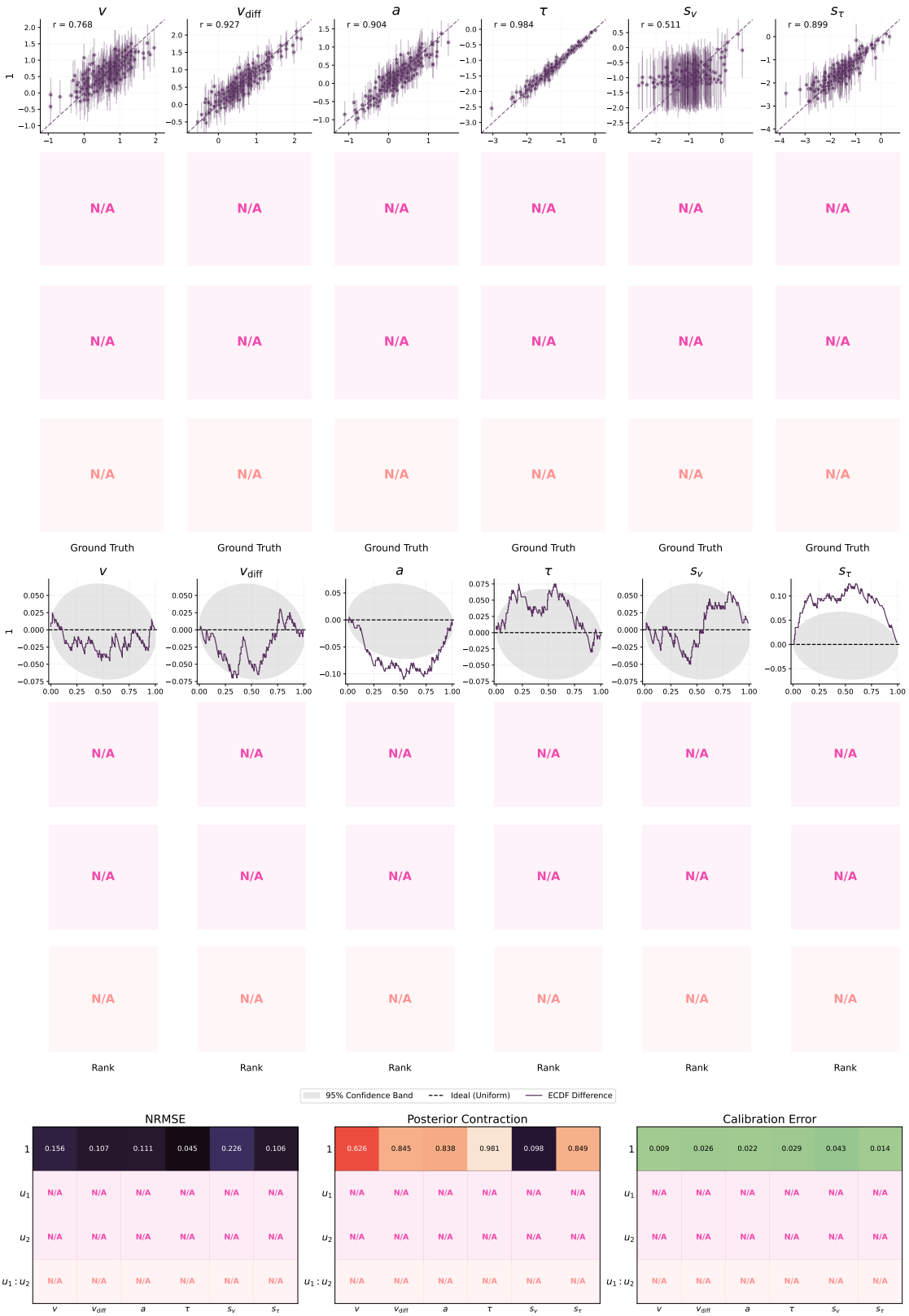

Figure 35. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM model family (Case **intercept_only**).

Figure 36. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM model family (Case **fixed**).

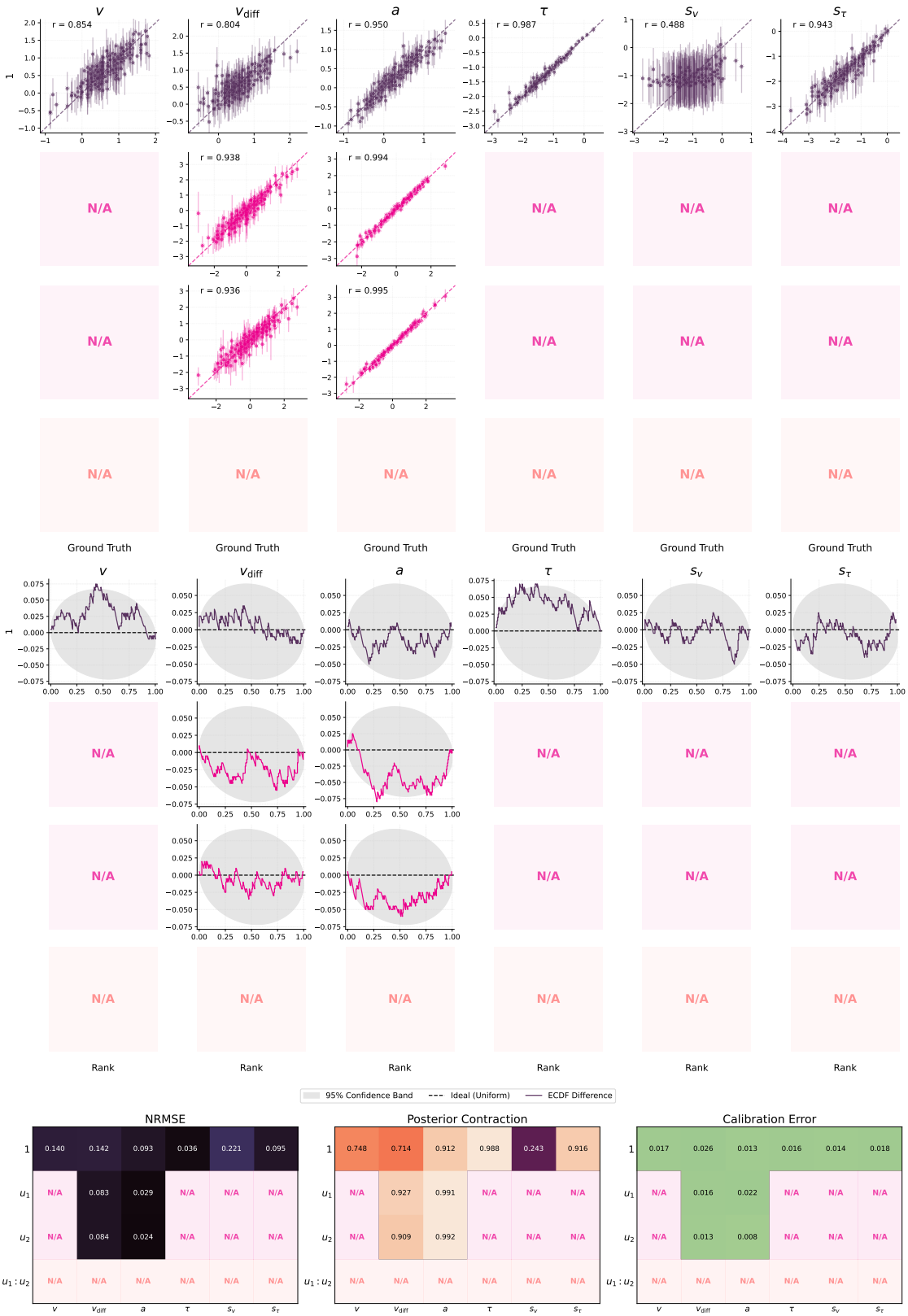

Figure 37. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM model family (Case **regressed**).

Figure 38. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM model family (Case **fixed_regressed**).

Figure 39. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for RDM model family (Case **interaction**).

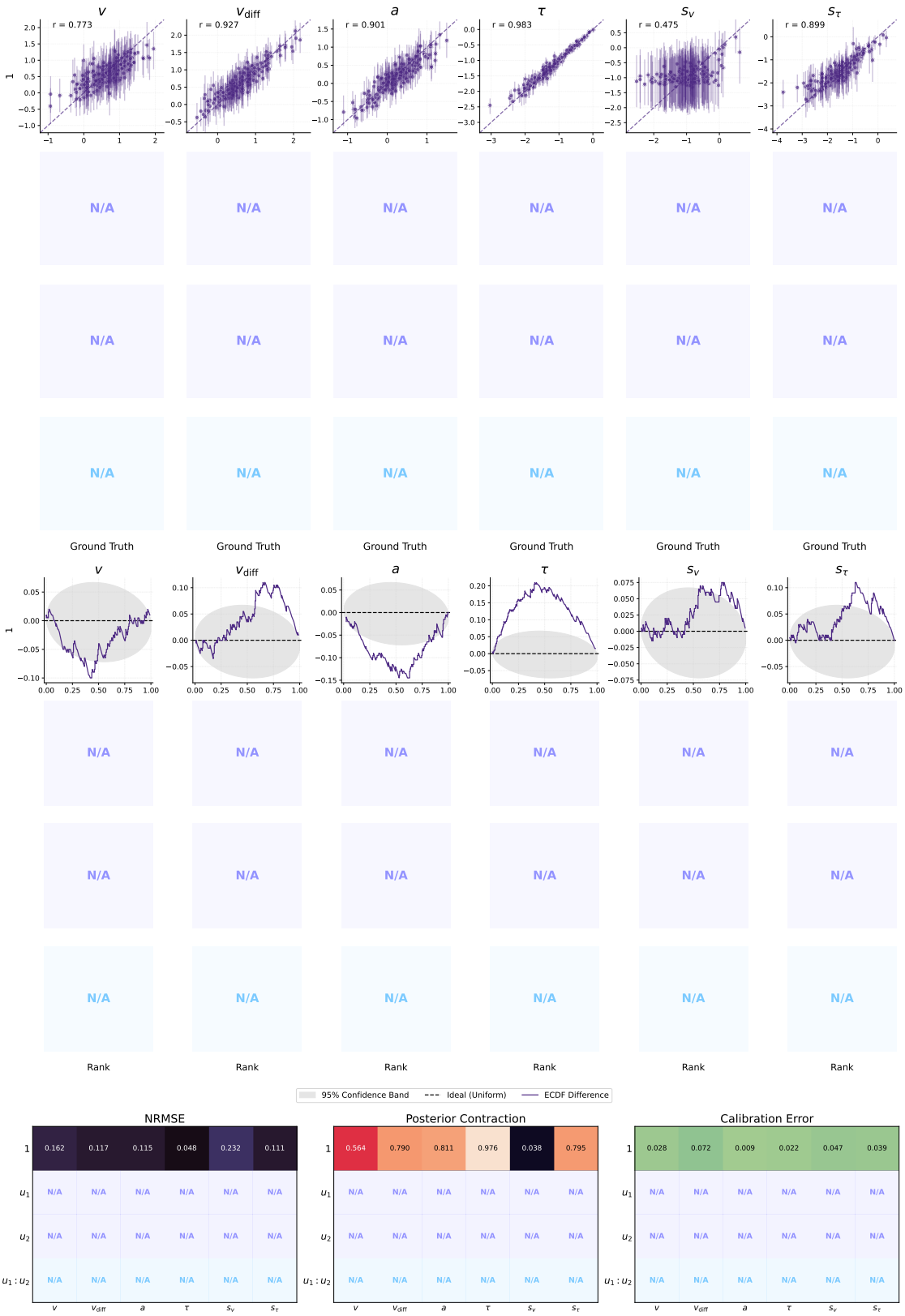

Figure 40. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for RDM model class (Case **intercept_only**).

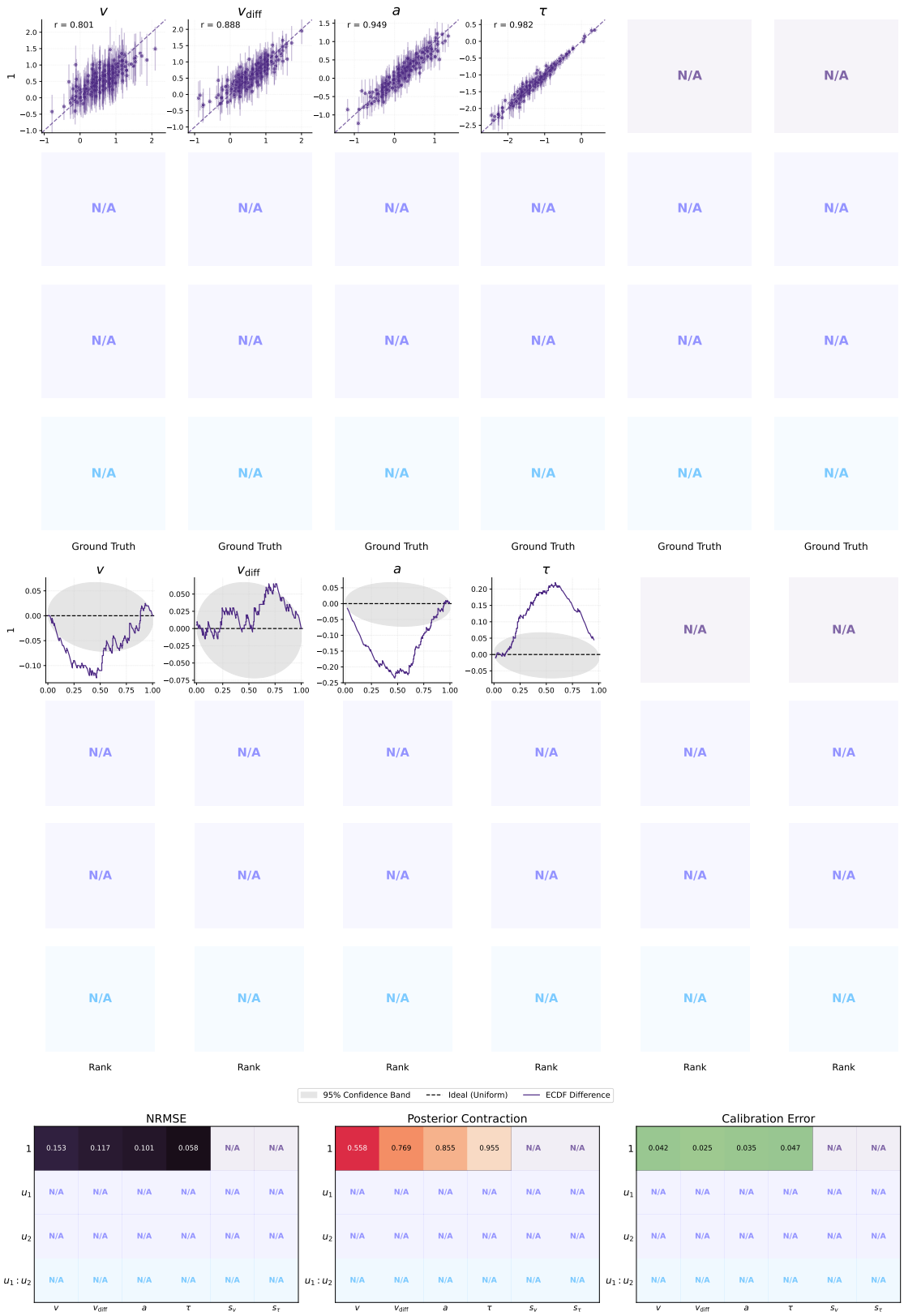

Figure 41. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for RDM model class (Case **fixed**).

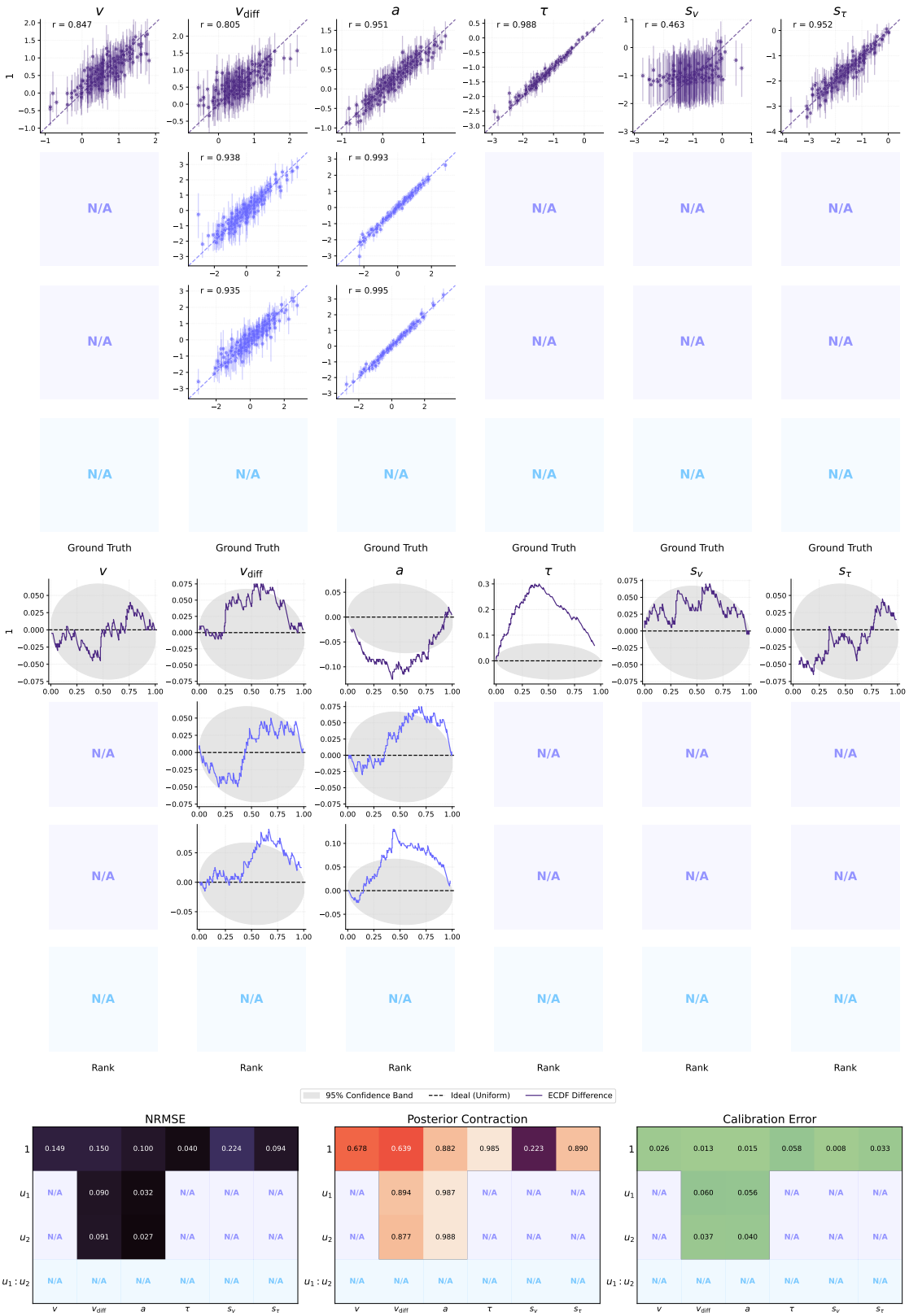

Figure 42. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for RDM model class (Case **regressed**).

Figure 43. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for RDM model class (Case **fixed_regressed**).

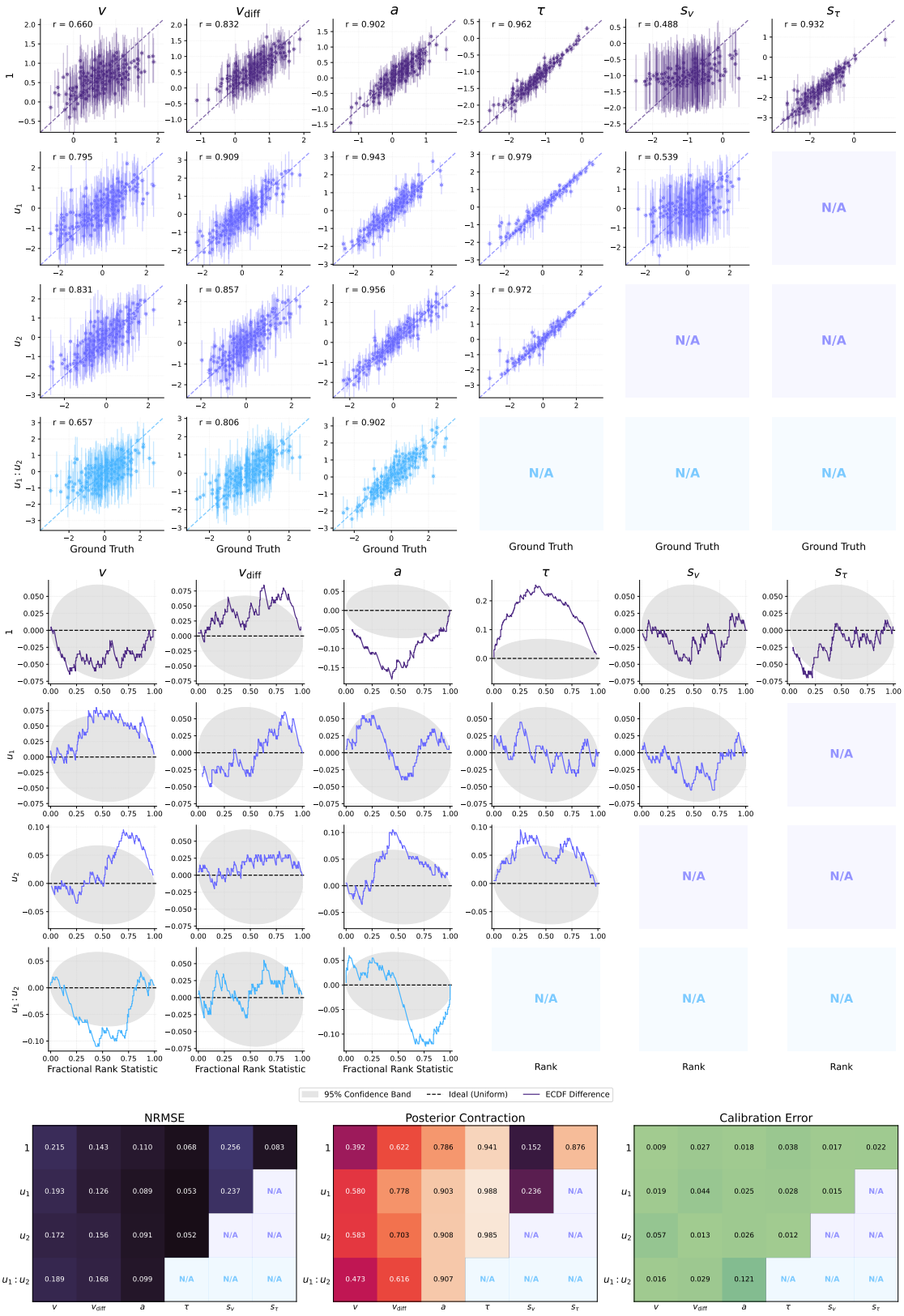

Figure 44. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for RDM model class (Case **interaction**).

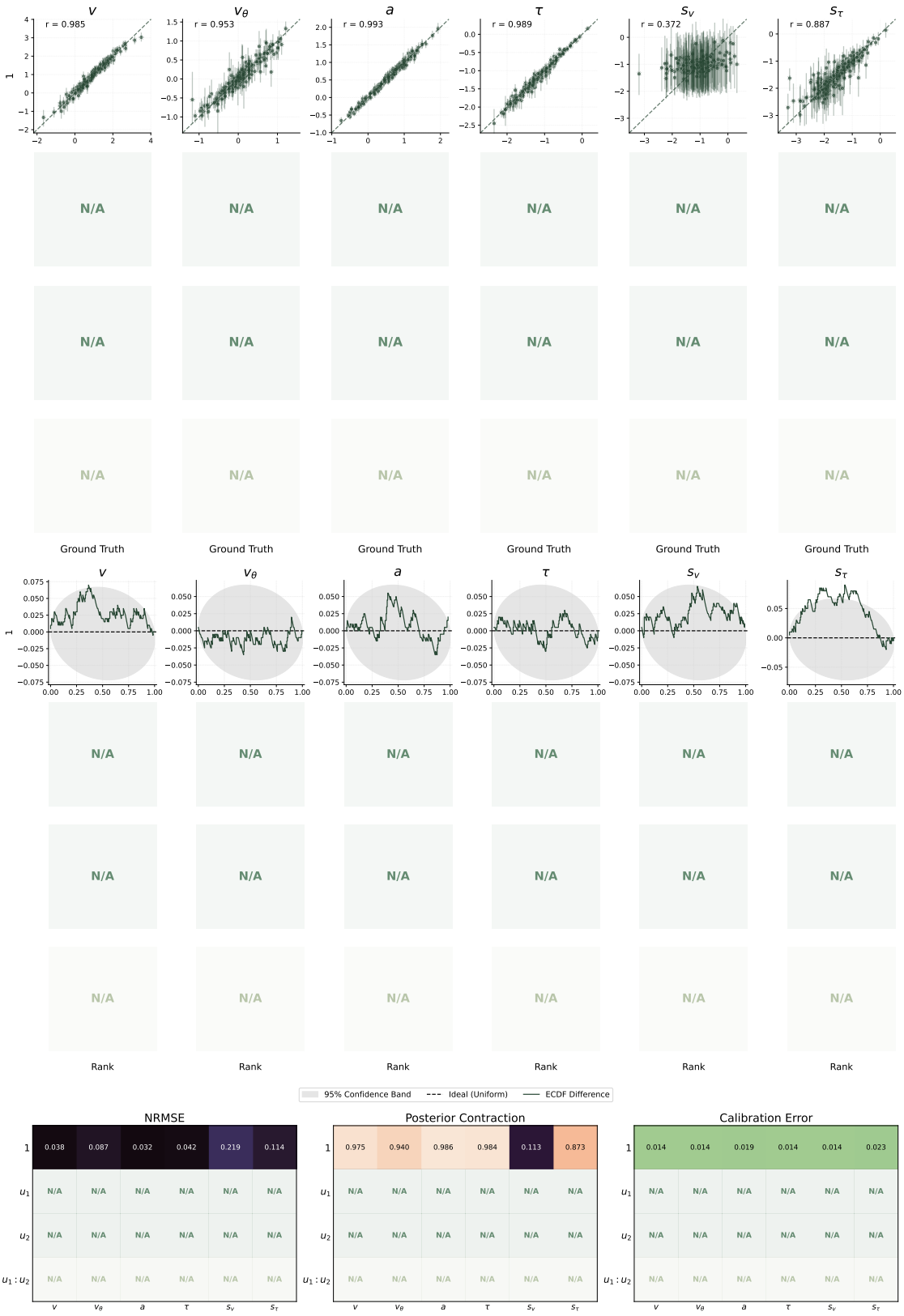

Figure 45. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM baseline (Case **intercept_only**).

Figure 46. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM baseline (Case **fixed**).

Figure 47. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM baseline (Case **regressed**).

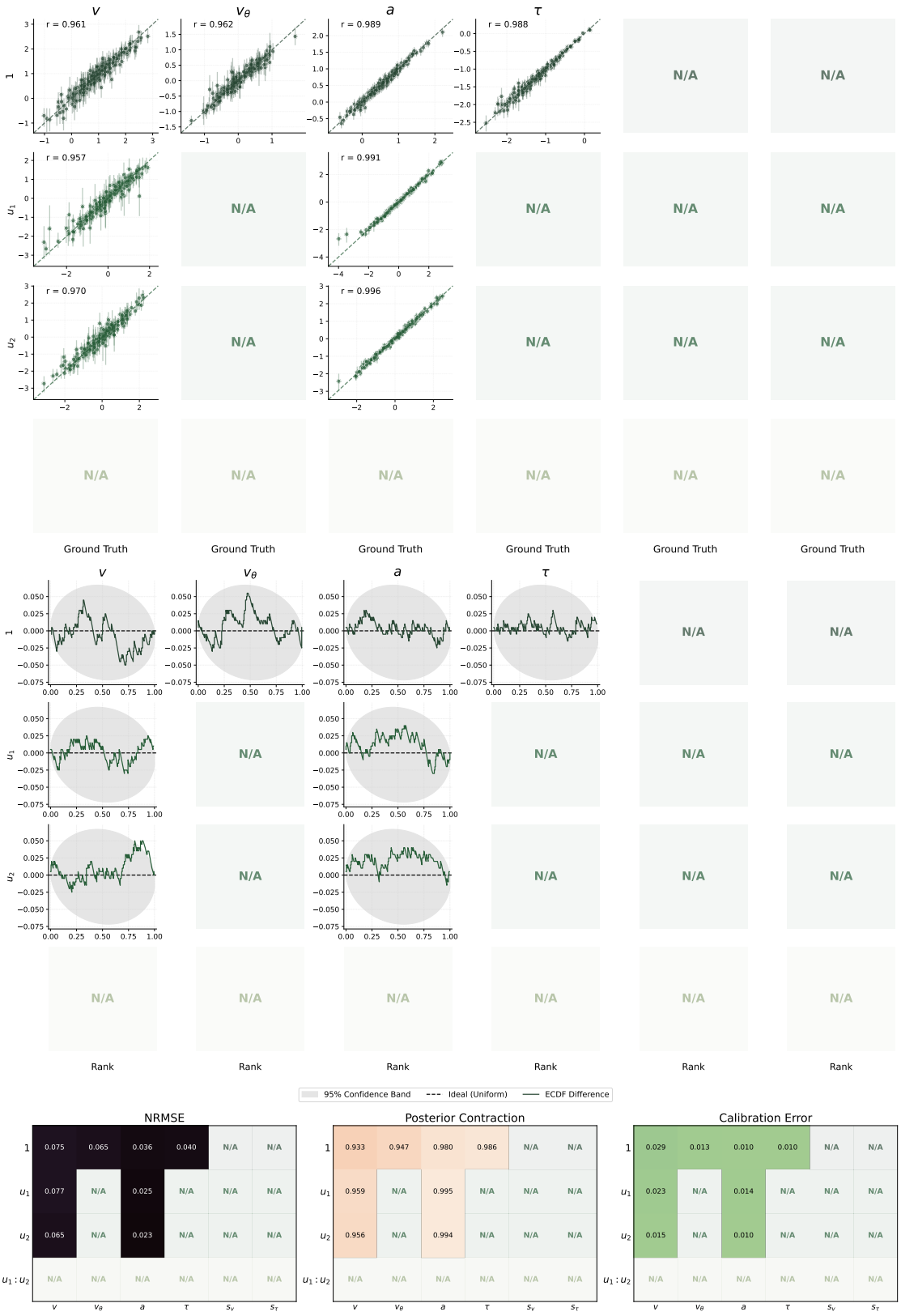

Figure 48. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM baseline (Case **fixed_regressed**).

Figure 49. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM baseline (Case **interaction**).

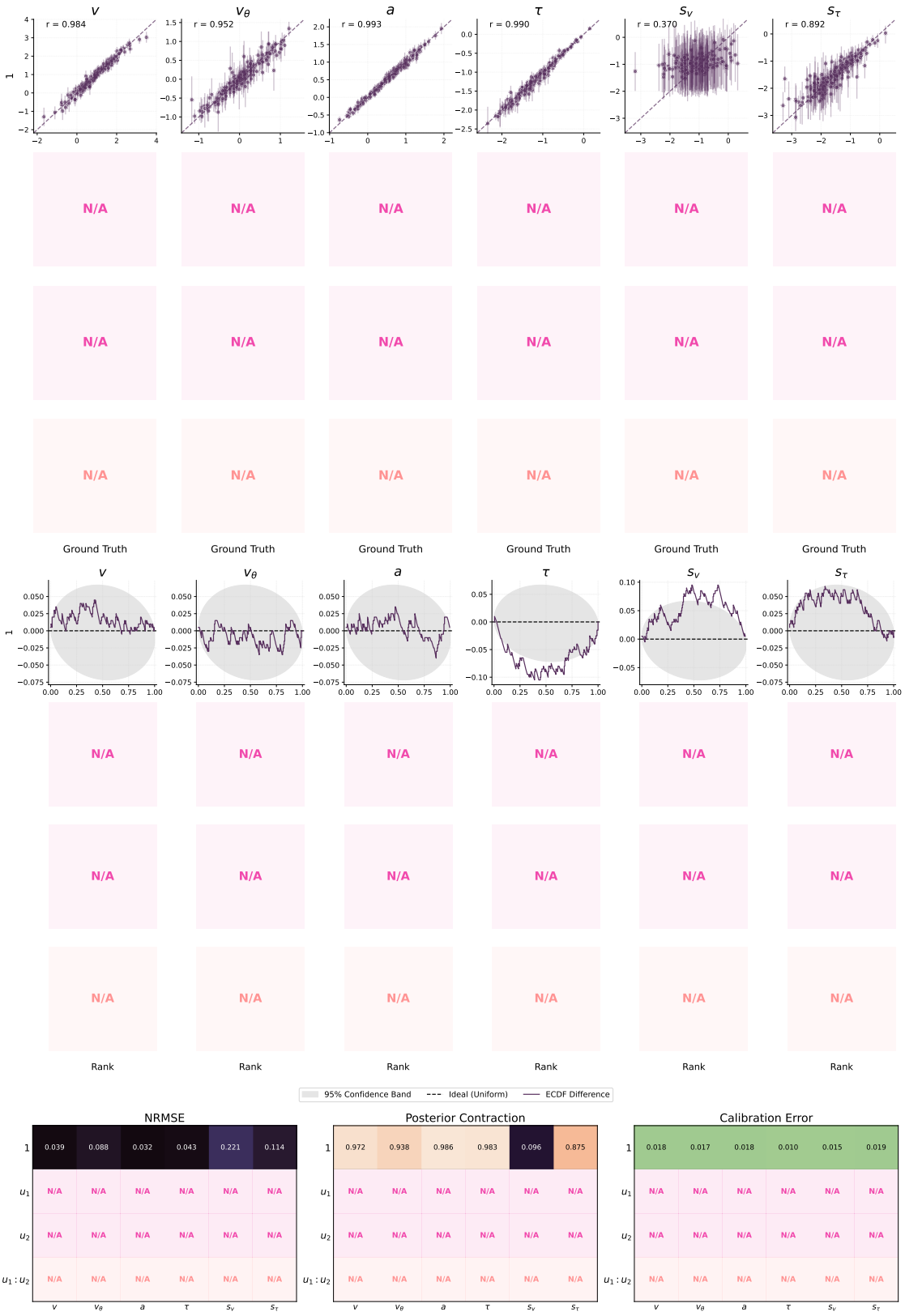

Figure 50. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM model family (Case **intercept_only**).

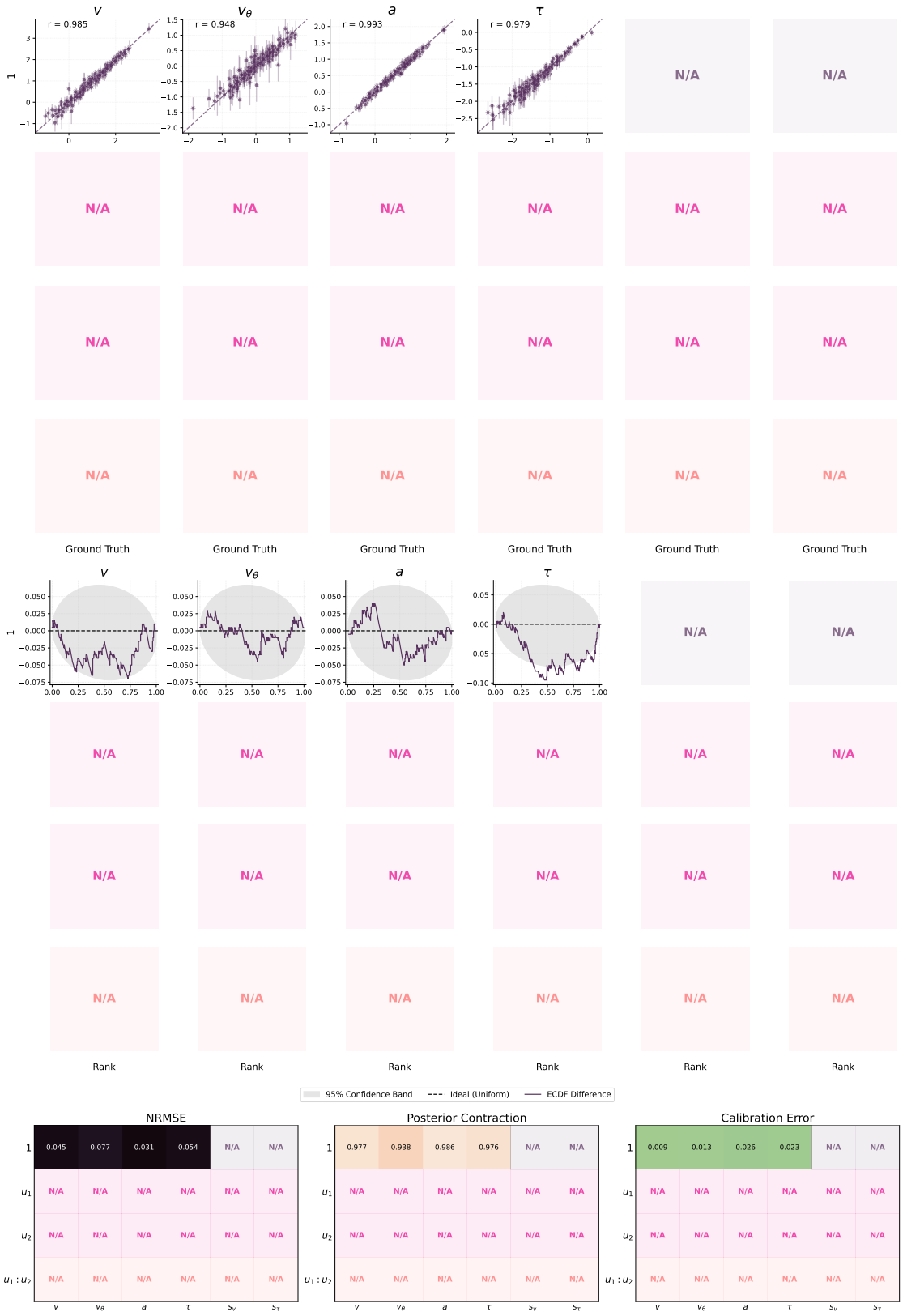

Figure 51. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM model family (Case **fixed**).

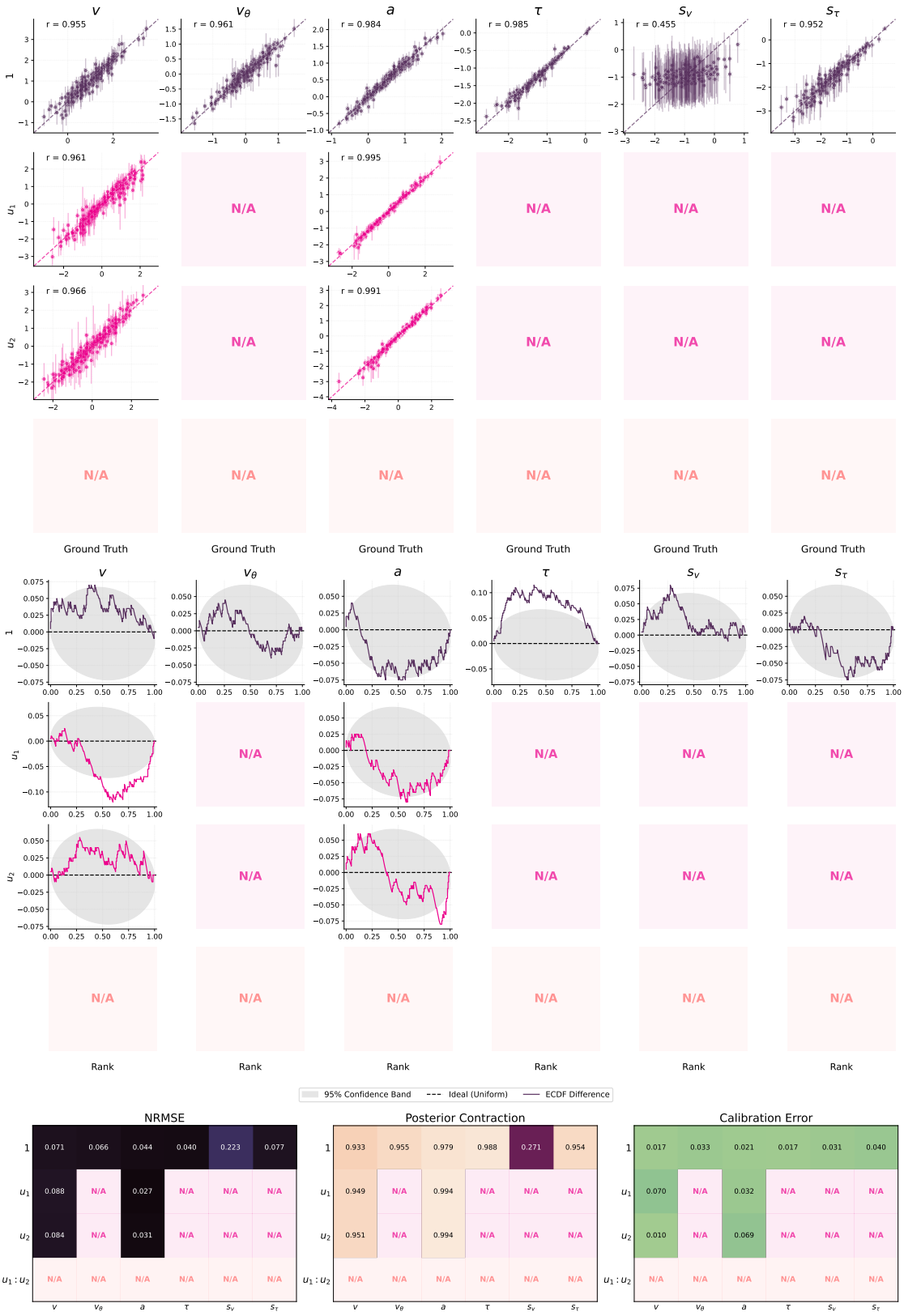

Figure 52. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM model family (Case **regressed**).

Figure 53. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM model family (Case **fixed_regressed**).

Figure 54. Parameter recovery (*top*), calibration ECDF (*middle*), and validation metrics (NRMSE, calibration error, and posterior contraction) for CDM model family (Case **interaction**).

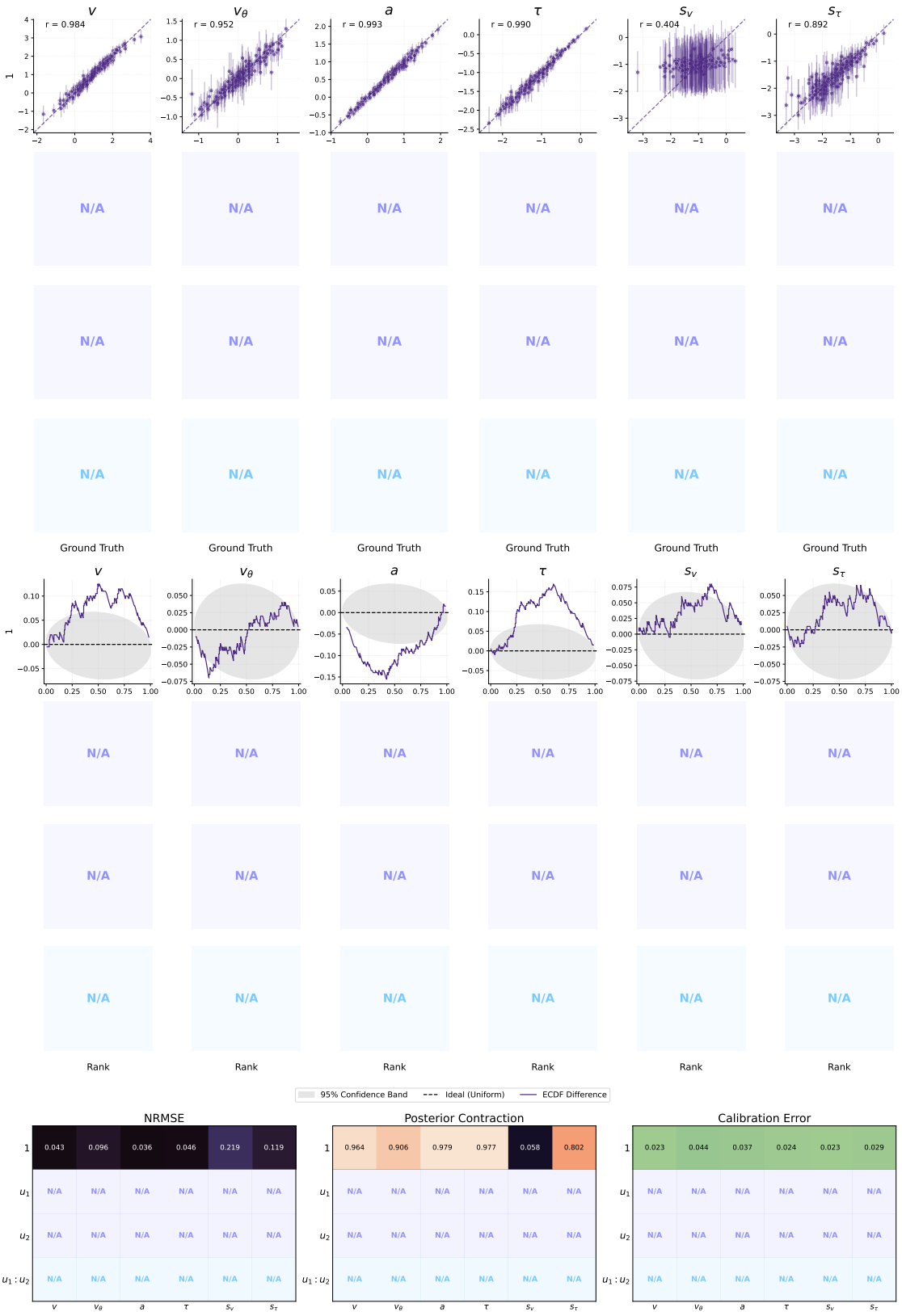

Figure 55. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for CDM model class (Case **intercept_only**).

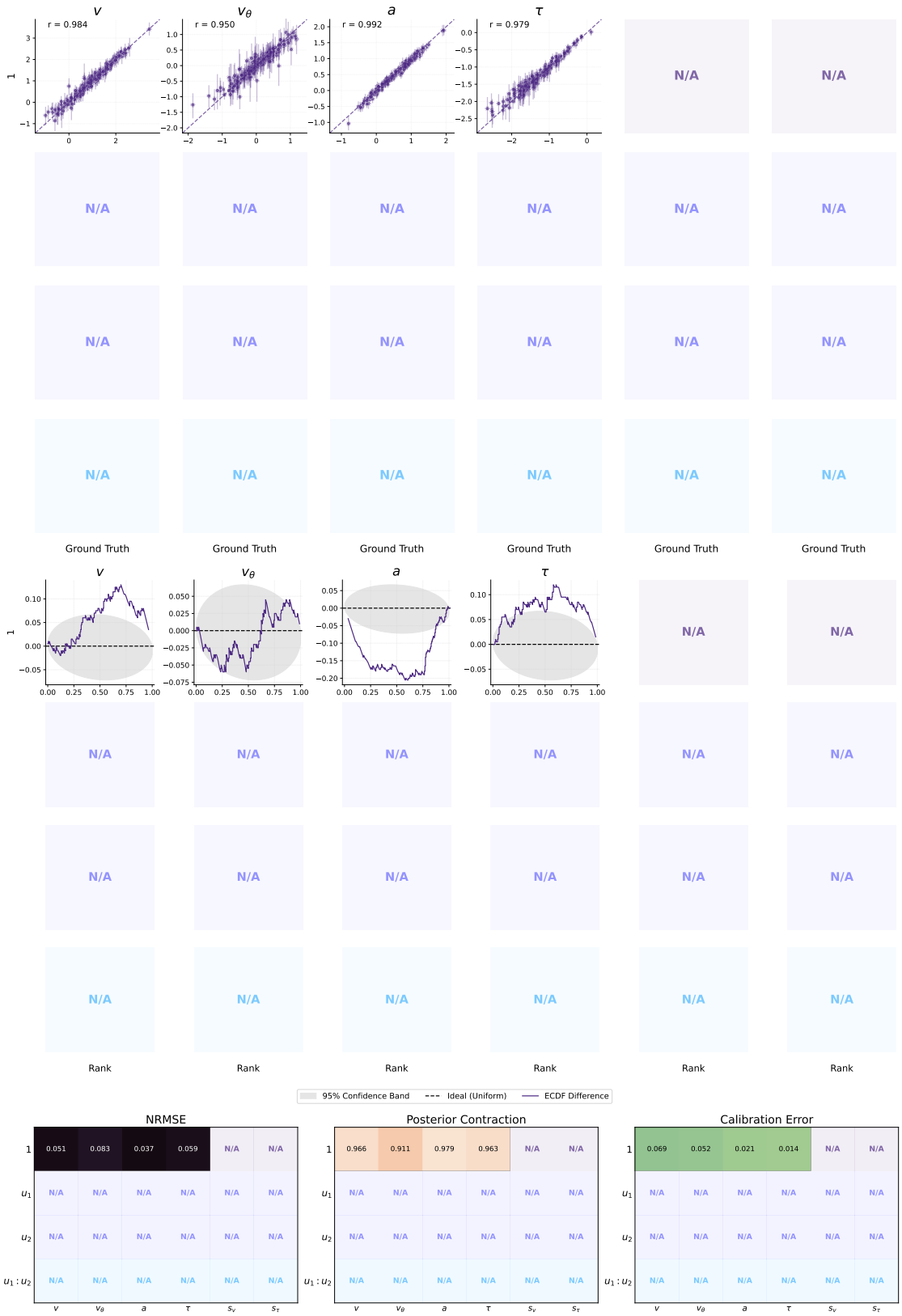

Figure 56. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for CDM model class (Case **fixed**).

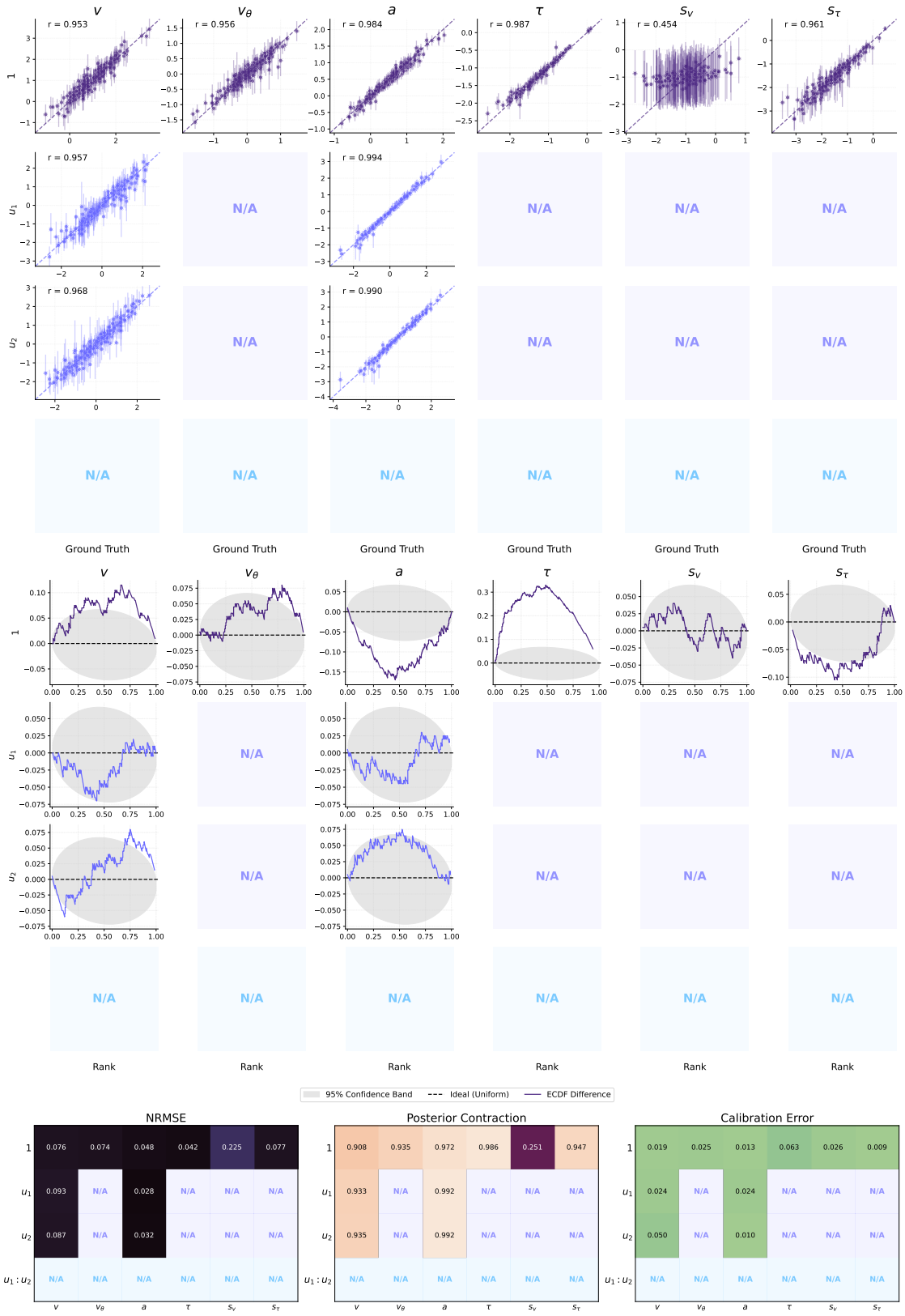

Figure 57. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for CDM model class (Case **regressed**).

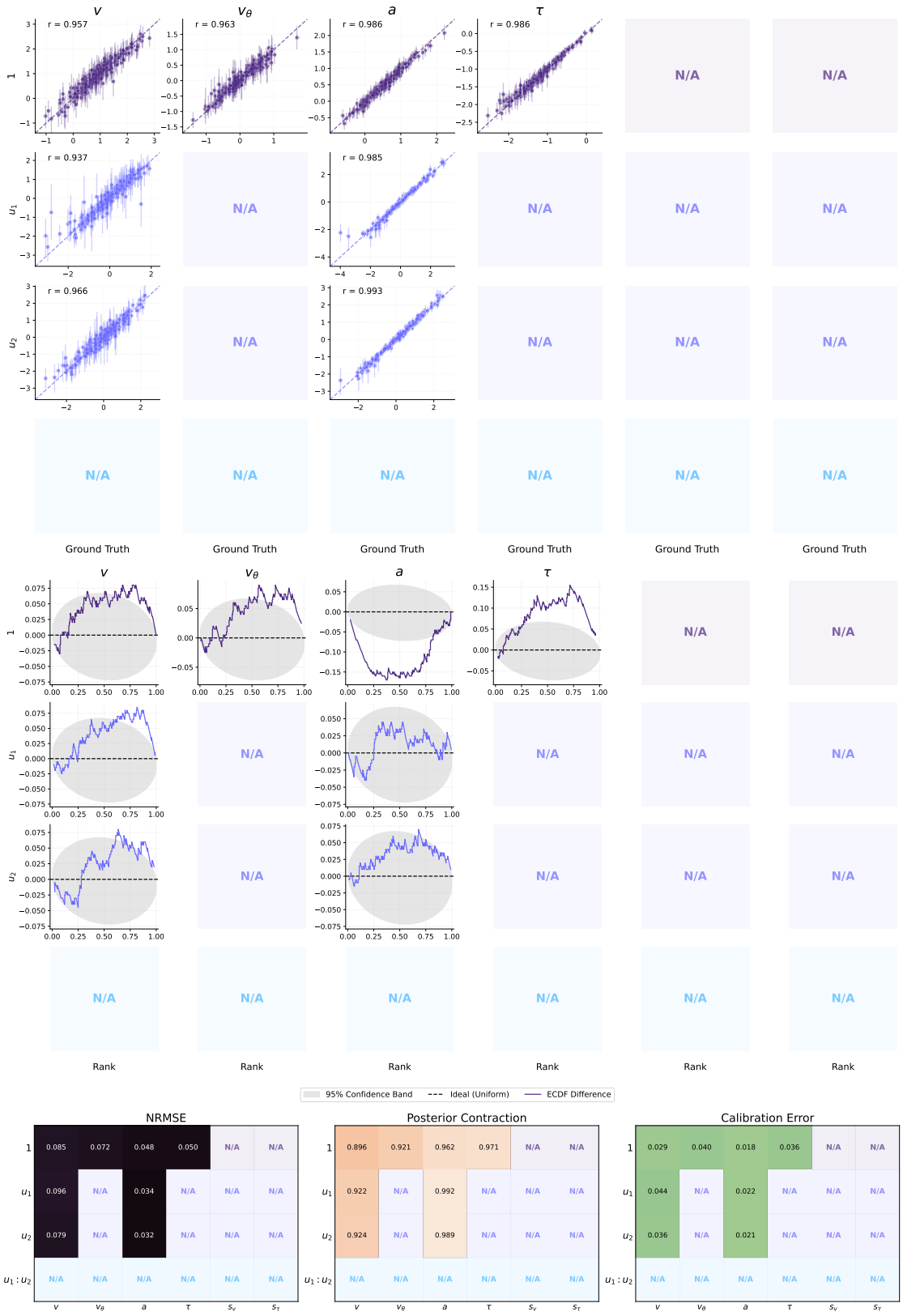

Figure 58. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for CDM model class (Case **fixed_regressed**).

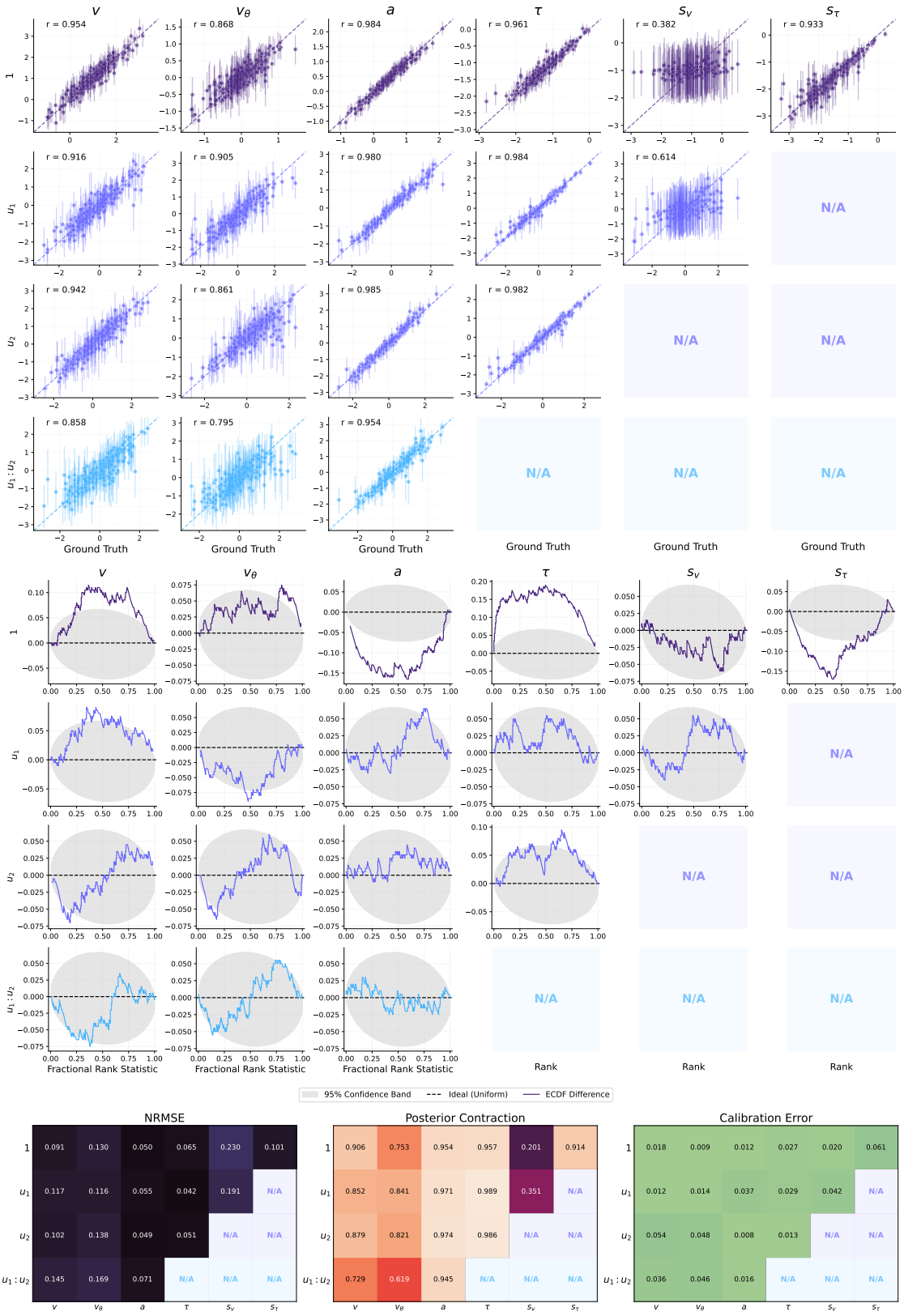

Figure 59. Parameter recovery (*top*), calibration ECDF (*middle*), and parameter-wise metrics (NRMSE, calibration error, and posterior contraction) for CDM model class (Case **interaction**).