Teodor Fredriksson
January 18, 2025

Supplementary Information

1. Unsupervised classification

Machine Learning algorithms are divided into supervised and unsupervised algorithms. It is interesting to study algorithms that cobnes supervised an unsupervised learning. Consider a system where the input is a vector $\mathbf{x}$ and the output is a probability vector $\mathbf{y(x)}$. The components $y_i(\mathbf{x}), i = 1, ..., N_c$ where N_c are the number of classes represent the probability that the label has class i. Given unlabelled data $\mathbf{u}_j,\ j = 1, 2, ..., N$ we wish to improve the supervised model.

1.1. Mutual Information Criterion.

Suppose that the marginal label distribution:

$$p(c) = \int p(x, c)dx = \int p(c|x)p(x)\, dx, \tag{1}$$

is uniform. An unsupervised classifier is good if is:

- **Decisive:** The output of the classifier can tell what class an input vector $\mathbf{x}$ belongs to,
- **Fair:** The model predicts each class with equal frequency given the training data. (We dont know that the label distribution is balanced on test data).

It is necessary to discover a way to measure decisivness and fairness. The general idea is for the predictions to retain as much information about $\mathbf{x}$ as possible. Therefore we measure the *mutual information.* betwen input $\mathbf{x}$ and output c:

$$\mathcal{I}(c, \mathbf{x}) = \int\int p(c, \mathbf{x}) \log \frac{p(c, \mathbf{x})}{p(c)p(\mathbf{x})}\, dc\, d\mathbf{x}, \tag{2}$$

$$= H\left(\mathbb{E}[p(y|x)]\right) - \mathbb{E}\left[H(p(y|x))\right] \tag{3}$$

The term $H\left(\mathbb{E}[p(y|x)]\right)$ is the emtropy minimization which makes the model output have low entropy and the term $-\mathbb{E}\left[H(p(y|x))\right]$ will make the predictions fair.

2. Semi-Supervised Learning

A C-classification problem in semi-supervised learning means utilziing a training set $\mathcal{X}$ containig labeled and unlabeld instances:

$$\mathcal{X} = \mathcal{L} \cup \mathcal{U}, \tag{4}$$

where :

$$\mathcal{L} = \{(x_i, y_i) \colon i = 1, 2, ..., N_L\}, \tag{5}$$

$$\mathcal{U} = \{u_i \colon i = 1, 2, ..., N_U\}, \tag{6}$$

to find the predictive predictive label distribution produced by x denoted $p(y|x)$. The number of labeled instances and the number of unlabeled instances different so it is important to balance them. The total loss will be

$$\ell = \ell_s + \lambda \ell_u, \tag{7}$$

where ℓ_s is the supervised loss and ℓ_u is the unsupervised loss. For loss function we choose the *cross-entropy*:

$$H(p, q) = -p \log q, \tag{8}$$

2.1. **Pseudo-Labeling.** Pseudo-labels are predicted labels for the unlabeled instances. These pseudo-labels are calculated by training a supervised model that poutputs the predictive probability $p(y|x)$ and then assign the unlabeled instance u with the class that has maximum predicted probability, i.e:

$$\hat{y} = \arg\max_{y} p(y|x). \tag{9}$$

The loss functions are defined as:

$$\ell_s = \frac{1}{N_L}\sum_{i=1}^{N_L} H(y_i, f(x_i)), \tag{10}$$

where y is the true label and f is the outout of the neural network. The unsupervised loss is given by:

$$\ell_u = \frac{1}{N_U}\sum_{i=1}^{N_U} H(y_i', f'(x_i)), \tag{11}$$

y_i' is the pseudo-label and f' is the unlabeled instance. This method is equivalent to *entropy regularization* [**?**]. Entropy Regularization is a method where unlabeled data is used to improve generalization performance using MAP estimation. The MAP estinate is computed by maximizing the posterior distribution

$$C(\theta, \lambda) = \sum_{i=1}^{n} \log p(y_i|x_i) - \lambda H(y|x), \tag{12}$$

where $H(y|x)$ is the conditional entropy of the class probabilities for unlabeled data:

$$H(y|x) = -\frac{1}{N_U}\sum_{i=1}^{N_U} p(y_i|x_i)\log p(y_i|x_i), \tag{13}$$

which measures the class overlap, as the class overlap decreases the denisty of data points get lower at the decision boundary. The conditional entropy is minimized in the pseudo-labeling method and is therefore equivalent to entropy minimization and wil improve generalization performance.

2.2. **Distribution Alignment.** To make predictions fair in semi-supervised learning it is necessary to minimize the mutual information Eq 1 By minimizing $H\left(\mathbb{E}[p(y|x)]\right)$ we ensure that the model predict each class label with equal frequencey given the training data but this is not useful if the marginal distribution $p(c)$ is not uniform. .Distribution alignment is another method to make the predictions of a model fair and works when $p(y)$ is non-uniform. During the training procedure we update the average of the model predictions on unlabeled data denoted $\widetilde{p}(y)$. Given the output of the model $q = p_m(y|u)$ distribution alignment scales q with $p(y)/\widetilde{p}(y)$ and normalizes the final distribution:

$$\widehat{q} = \mathrm{DA}(q) = \mathrm{Normalize}\left(q \cdot \frac{p(y)}{\widetilde{p}(y)}\right), \tag{14}$$

where Normalize is defined as:

$$\mathrm{Normalize}(x)_i = \frac{x_i}{\sum_j x_j}. \tag{15}$$

2.3. **Uniform Alignment.** Uniform alignment has the same purpose as distribution alignment. If define the distribution of the pseudo labels as an expectation $\mathbb{E}_{\mathcal{D}_U}[p(y|u)]$ wich is estimated using the sample mean denoted $\widehat{\mathbb{E}}_{N_U}[p(y|u)]$ duing the EMA of batch predictions. To normalize each prediction q on unlabeled data we formulate the UA operation as:

$$\mathrm{UA}(q) = \mathrm{Normalize}\left(q \cdot \frac{u(C)}{\widehat{\mathbb{E}}_{N_U}[p(y|u)]}\right) \tag{16}$$

where $u(C)$ is a uniform distrubution. The difference between DA and UA is the way they are used in the unsupervised loss. Normalization makes the predictions favor less-predicted classes. In DA the normalized predictions are used to create pseudo-labels that used in the cross-entropy. However, normalization might generate more errorous pseudo-labels. In UA the original predictions are used to produce pseudo-labels and the normalized predictions are used to calculate weights so that both quantity and quality is preserved.

2.4. **Consistency Regularization.** Assumes that the the model provides the similar predictions when fed peturbated data.. It is comonly used in semi-supervised learning to utilize unlabeled data by applying weak and strong augmentations to it.

2.5. **FixMatch.** The FixMatch algorithm requires the user to provide a threshold τ, weak and strong augmentations $\alpha(\cdot)$ and $\mathcal{A}(\cdot)$ respectively. The algorithm the utlizies labeled instances (x, y), the weakly augmented unlabeled instances $\alpha(u)$ and the strongly augmented unlabeled instances $\mathcal{A}(u)$. Supervised loss is set to the cross-entropy between the labels y and the predictive label distribution given the weakly augmented labeled instances $\alpha(u)$:

$$\ell_s = \frac{1}{B}\sum_{i=1}^{B} H(y_i\ p_m(y_i|\alpha(x_i))), \tag{17}$$

where H is the cross-entropy. The pseudo-labels are computed as $y^w = \arg\max_y p(y|\alpha(u))$, but only the pseudo-lebels hose predictive porbability is above the threshold will be used in the unsupervised loss that is defined as:

$$\ell_u = \frac{1}{\mu B}\sum_{i=1}^{\mu B} 1(\max p_m(y_i|\alpha(u_i)) > \tau) \cdot H(y_i^w, p_m(y_i|\mathcal{A}(u_i)), \tag{18}$$

where $1(\cdot)$ is the indicator function:

$$1(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}. \tag{19}$$

2.6. **SimMatch.** The SimMatch algorithms extends upon the FixMatch in many aspects. First weak and strong augmentations are applied to the unlabeled data u. We then train convolutional encoders to extract features from the augmented instances $\mathbf{h}^w = F_t(\alpha(u))$ and $\mathbf{h}^s = F_s(\mathcal{A}(u))$ as well as labeld instances $\mathbf{h}^\ell = F_t(x)$. We the use student and teacher encoders φ_t and φ_s respectively to compute the sematic similarities:

$$p^w = \mathrm{DA}\left(\varphi_s\left(\mathbf{h}^w\right)\right), \tag{20}$$

$$p^s = \varphi_s\left(\mathbf{h}^s\right), \tag{21}$$

$$p^\ell = \varphi_s\left(\mathbf{h}_s^\ell\right), \tag{22}$$

where DA stands for distribution alignment. We assume that that there is a nonlinear functions $g_t(\cdot)$ and $g_s(\cdot)$ that maps a representation $\mathbf{h}$ to a low-dimensional embedding $\mathbf{z}$, and we compute:

$$\mathbf{z} = g_t\left(\mathbf{h}^\ell\right), \tag{23}$$

$$\mathbf{z}^w = g_t\left(\mathbf{h}^w\right), \tag{24}$$

$$\mathbf{z}^s = g_s\left(\mathbf{h}^s\right). \tag{25}$$

It is assuned that we have access to K (where K is provided by the user) weakly augmented embeddings for many different samples so that the instance similarity can be computed using similarity distributions:

$$q_i^w = \frac{\exp\left(sim\left(\mathbf{z}^w, \mathbf{z}_i\right)\right)/t}{\sum_{k=1}^{K}\exp\left(sim\left(\mathbf{z}^w, \mathbf{z}_k\right)\right)/t}, \quad i = 1, 2, .., K \tag{26}$$

$$q_i^s = \frac{\exp\left(sim\left(\mathbf{z}^s, \mathbf{z}_i\right)\right)/t}{\sum_{k=1}^{K}\exp\left(sim\left(\mathbf{z}^s, \mathbf{z}_k\right)\right)/t}, \quad i = 1, 2, .., K. \tag{27}$$

where the *sim* operation is defined as:

$$sim(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^T\mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}. \tag{28}$$

Next we use the semantic similarity to adjust the instnace pseudo-label. This is done by unfolding p^w into K dimensional space to calibrate it with q^w:

$$q^w \mapsto q^{unfold}, \quad \mathbb{R}^L \to \mathbb{R}^K, \tag{29}$$

this is performed by matching semantic and instance similarities so that they have the same ground truth label:

$$p_i^{unfold} = p_j^w, \quad \text{where} \quad class(q_i^w) = class(p_j^w). \tag{30}$$

The final step of the calibration is to scale q^w with p^{unfold}:

$$\widehat{q}_i = \frac{q_i^w p_i^{unfold}}{\sum_{k=1}^K q_k^w p_k^{unfold}}. \tag{31}$$

Similarly the instance similarity is used to adjust the sematic pseudo-label. This is done by first projecting q^w to L dimensional space by matching the ground truth label of semantic and instance similarities, the projected instance similarities are denoted q^{agg}:

$$q_i^{agg} = \sum_{j=0}^{K} 1(class\,(p_i^w) = class\left(q_j^w\right))q_j^w. \tag{32}$$

Finally, smoothing is applied to p^w and q^{agg} to calculate the adjusted pseudo label:

$$\hat{p}_i = \alpha p_i^w + (1-\alpha) q_i^{agg}, \tag{33}$$

where α is a hyperparameter that determinees that weight of semantic and instance information. The supervised loss function becomes:

$$\ell_s = \frac{1}{B}\sum_{i=1}^{B} H\left(y, p^\ell\right), \tag{34}$$

and unsupervised loss function becomce:

$$\ell_u = \frac{1}{\mu B}\sum_{i=1}^{\mu B} 1(\max \hat{p} > \tau) H\left(\hat{p}, p^s\right). \tag{35}$$

SimMatch also adds an instance loss:

$$\ell_{instance} = \frac{1}{\mu B}\sum_{i=1}^{\mu B} H\left(\hat{q}, q^s\right), \tag{36}$$

so the total loss function is:

$$\ell = \ell_s + \lambda_u \ell_u + \lambda_s \ell_{instance} \tag{37}$$

2.7. **SoftMatch.** The FixMatch algorithm and SimMatch algorithms all incorportate a constant threshold over the course of training.then the algorithm yields high-quality labels because many labels are discarded. If a dynamically growing threshold like in Dash and AdaMatch. Like in FixMatch we wish to minimize the total loss function:

$$\ell = \ell_s + \lambda_u \ell_u, \tag{38}$$

where the supervised loss function is:

$$\ell_s = \frac{1}{N_L}\sum_{i=1}^{N_L} H(y_i | p(y_| x_i)). \tag{39}$$

The unsupervised loss is different from FixMatch and SimMatch:

$$\ell_u = \frac{1}{N_U}\sum_{i=1}^{N_U} \lambda(p_i) H(\widehat{p}, p(y|\mathcal{A}(u_i))), \tag{40}$$

and is known as the *weighted cross-entropy* where $\lambda(p)$ is called *sample weight function*, $\widehat{p} = \arg\max(p)$ with range $[0, \lambda_{max}]$ and $p = p(y|\alpha(u))$. $\lambda(p)$ is the distribution of p and its probability mass function is defined as:

$$\overline{\lambda}(p) = \frac{\lambda(p)}{\sum_j \lambda(p_j)}. \tag{41}$$

To access quantity and quality of pseudo-labels the following definitions are formulated. If $u \sim \mathcal{D}_{\mathcal{U}}$, then the the qualntitfy of pseudo labels denoted $f(p)$ is defined as the expectated value of the sample weight function:

$$f(p) = \mathbb{E}_{\mathcal{D}_{\mathcal{U}}}[\lambda(p)]. \tag{42}$$

The quality of labels $g(p)$ is defined as the expectation of the weight $0-1$ error of the pseudo-labels:

$$g(p) = \sum_{i=1}^{N_U} 1(\widehat{p}_i = y_i^u)\overline{\lambda}(p_i). \tag{43}$$

Since $\lambda(p)$ is dependent on the marginal distribution of $p(y|\alpha(u))$ it is important to design $\lambda(p)$ so that it will incorporate both quantity and quality. For intsnace in the Pseudo-labeling algorithm we would have $\lambda_{pseudo}(p) = \lambda_{max}$ and this would give:

$$f_{pseudo}(p) = \mathbb{E}_{\mathcal{D}_{\mathcal{U}}}[\lambda(p)] = \lambda_{max} \tag{44}$$

$$\overline{\lambda}_{pseudo}(p) = \frac{1}{N_U} \tag{45}$$

as well as:

$$g_{pseudo}(p) = \frac{1}{N_U}\sum_{i=1}^{N_U} 1(\widehat{p}_i = y_i^u), \tag{46}$$

which means that all pseudo-labels are equally correct. For the fixmatch algorithm:

$$\lambda(p) = \begin{cases} \lambda_{max} & \text{if } \max(p) \geq \tau \\ 0 & \text{otherwise} \end{cases}, \tag{47}$$

and let $\widetilde{N}_U^{\tau} = \sum_{i=1}^{N_U} 1(\max(p_i) \geq \tau)$ then:

$$f_{fixmatch}(p) = \lambda_{max}\frac{\widetilde{N}^{\tau}}{N_U} \tag{48}$$

$$\overline{\lambda}_{fixmatch}(p) = \begin{cases} \frac{1}{N_U} & \text{if } \max(p) \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

as well as:

$$g_{fixmatch}(p) = \frac{1}{\widetilde{N}_U}\sum_{i=1}^{\widetilde{N}_U} 1(\widehat{p}_i = y_i^u), \tag{50}$$

so we can see that FixMatch provides high quality labels than Pseudo-labeling and Pseudo-labeling provides higher quantity but lower quality labels than FixMatch. In SoftMatch, the pmf $\overline{\lambda}(p)$ is assumed to be a Gaussian truncated distrubution with mean μ_t and variance σ_t.

$$\lambda(p) = \begin{cases} \lambda_{max}\text{exp}\left(-\frac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right), & \text{if } \max(p) < \mu_t \\ \lambda_{\max} & \text{otherwise,} \end{cases} \tag{51}$$

The parameters μ_t and σ_t are estimated from historical predictions of the model, at the tth iteration we compute the emprical mean and the variance as:

$$\hat{\mu}_b = \frac{1}{N_U}\sum_{i=1}^{N_U} \max(p_i), \tag{52}$$

$$\hat{\sigma}_b^2 = \frac{1}{N_U}\sum_{i=1}^{N_U} (\max(p_i) - \hat{\mu}_b)^2. \tag{53}$$

we then aggregate the batch statistics for more stable estimation, using Exponential Moving Average (EMA) with momentum m over previous batches:

$$\hat{\mu}_t = m\hat{\mu}_{t-1} + (1-m)\hat{\mu}_b, \tag{54}$$

$$\hat{\sigma}_t^2 = m\hat{\sigma}_{t-1}^2 + (1-m)\frac{N_U}{N_U-1}\hat{\sigma}_b^2. \tag{55}$$

Here the unbiased variance is used for the EMA and the intial conditions are:

$$\begin{cases} \hat{\mu}_0 & = \frac{1}{C}, \\ \hat{\sigma}_0^2 & = 1, \end{cases} \tag{56}$$

where C are the number of labels. The pmf, quality and quatity functions are derived to:

$$\lambda_{softmatch}(p) = \begin{cases} \dfrac{\exp\left(-\frac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right)}{\frac{N_U}{2}+\sum_{i=1}^{\frac{N_U}{2}} \exp\left(-\frac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right)} \\ \dfrac{1}{\frac{N_U}{2}+\sum_{i=1}^{\frac{N_U}{2}} \exp\left(-\frac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right)} \end{cases} \tag{57}$$

$$f(p) = \frac{\lambda_{max}}{2} + \frac{\lambda_{max}}{N_U}\sum_{i=1}^{N_U} \exp\left(-\frac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right) \tag{58}$$

$$g(p) = \frac{1}{2\hat{N}_U^{\mu_t}}\sum_{i=1}^{\hat{N}_U^{\mu_t}} 1(\hat{p}_i = y_j^u) + \frac{1}{2(N_U - \hat{N}_U^{\mu_t})}\sum_{i=1}^{N_U-\hat{N}_U^{\mu_t}} 1(\hat{p}_i = y_j^u)\exp\left(-\frac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right) \tag{59}$$

which indicates that SoftMatch has both high quality and quantity. Sometimes, if learning difficulty of each label class is different then the generated pseudo-label distribution is imbalanced it limits the generalization of the PMF assimption (WHAT IS THIS?). To encourage a more uniform label distribution we use uniform alginment UA. If UA is plugged in then the final sample weight function becomes:

$$\lambda(p) = \begin{cases} \lambda_{max}\exp\left(-\frac{(\mathrm{UA}(\max(p))-\mu_t)^2}{2\sigma_t^2}\right), & \text{if } \max(p) < \mu_t \\ \lambda_{\max} & \text{otherwise,} \end{cases} \tag{60}$$

2.8. **FreeMatch.** FreeMatch utilizes supervised loss ℓ_s and unsupervised loss ℓ_u just as fixmatch. In addition FreeMatch uses a fairness loss $\ell_f = u(C) \cdot \log E_{\mu B}[q_b]$ where $u(C)$ is a uniform prior distribution. The self-adaptive thresholding (SAT) method is used to balance the quantity-quality trade-off. SAT automatically defines and adjusts the threshold. When training starts the threshold is low to include more potentialy wrong labels. As training progresses the threshold increases as the model is becoming more confident. The threshold for each label class c at time t is defined denoted as $\tau_t(c)$. The global threshold is related to the models confidence on unlabeled data and should increase over time in order to exclude potenial wrong labels. The global threshold is computed during the learning process using EMA. The global threshold is defined as:

$$\tau_t = \begin{cases} \frac{1}{C}, & \text{if } t = 0 \\ \lambda\tau_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B}\max(q_b), & \text{otherwise} \end{cases}. \tag{61}$$

where $\lambda \in (0, 1)$ is the momentum decay of EMA and C is the number of class labels in the dataset. Next we define a local treshold which is class-speific:

$$\tilde{p}_t(c) = \begin{cases} \frac{1}{C}, & \text{if } t = 0 \\ \lambda \tilde{p}_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} q_b(c), & \text{otherwise} \end{cases} \tag{62}$$

The global and local thresholds are combined to defined the self-adaptive threshold $\tau_c(t)$, defined as:

$$\tau_t(c) = \text{MaxNorm}\left(\tilde{p}_t(c)\right) \cdot \tau_t \tag{63}$$

$$= \frac{\tilde{p}_t(c)}{\max\left\{\tilde{p}_t(c) : c \in [C]\right\}} \cdot \tau_t \tag{64}$$

The unuspuervised loss becomes:

$$\ell_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} 1(\max p_m(y|\alpha(u) > \tau_t(c)) H(y^w, p_m(y|\mathcal{A}(u)). \tag{65}$$

where $1(\cdot)$ is the indicator function:

$$1(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise.} \end{cases} \tag{66}$$

The fairness loss is included to make the model do diverse predictions for each class. To maximize fairness we need to optimize $-E[p(y|u)]$, see section 1. To estimate the expected predicitve probability given the unlabeled data we approximate it as the cross-entropy between the local threshold $\tilde{p}_t$ and the expected predictive probability given the strongly-augmented unlabeled instances $\overline{p} = E[p(y|\mathcal{A}(u))]$. Furthermore we estimate process both $\overline{p}$ and $\tilde{p}$ as below.

$$\overline{p} = \frac{1}{\mu B}\sum_{i=1}^{\mu B} 1(\max(q_b) \geq \tau_t(\arg\max(q_b)) p(y|\mathcal{A}(u)), \tag{67}$$

$$\overline{h} = \text{Hist}_{\mu B}\left(1\left(\max(q_b) \geq \tau_t(\arg\max(q_b)) y^s\right)\right). \tag{68}$$

The normalization $\tilde{h}$ of $\tilde{p}$ is calculated using smoothing.

$$\tilde{h}_t = \lambda \tilde{h}_{t-1} + (1-\lambda)\text{Hist}_{\mu B}(\hat{q}_b), \tag{69}$$

so the fairness loss is given by:

$$\ell_f = -H\left(\text{Normalize}\left(\frac{\tilde{p}_t}{\tilde{h}_t}\right), \text{Normalize}\left(\frac{\overline{p}}{\overline{h}}\right)\right). \tag{70}$$

The total loss that is minimized by the FreeMatch algorithm is:

$$\ell = \ell_s + w_u \ell_u + w_f \ell_f. \tag{71}$$

3. Bayesian Data Analysis

This section outlines how the Bayesian Data Analysis was performed. We follow the recommendations and guidlines of [1, 2, 3] amd asses the Markov Chain Monte Carlo (MCMC) chains and Posterior Predictive Checks.

3.1. Assesing the MCMC chains.

All analysis were conducted using the `R` programming language and the `cmdstanr` library. Each model was computed using four parallel chains using 2000 iterations and an additional 200 warmup iterations. None of the iteration diverged as can be seen in the traceplots Figures 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 and therefore the MCMC simulation is considered valid.

In addition we investigate diagnostics of the posterior estimates, such as *Gelman-Rubin Potential Scale Reduction*$(\widehat{R})$ [4] the number of *efficient samples* (n_{eff}). Tables 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 contains three columns. The first column contains $\widehat{R}$ and the other two contains two different estiamtes of the number of effective samples. As a rule of thumb we should have $\widehat{R} < 1.01$ which is satisfied in all scenarios. The number of efficient samples should be atleast 200, which is satisfied in all cases.

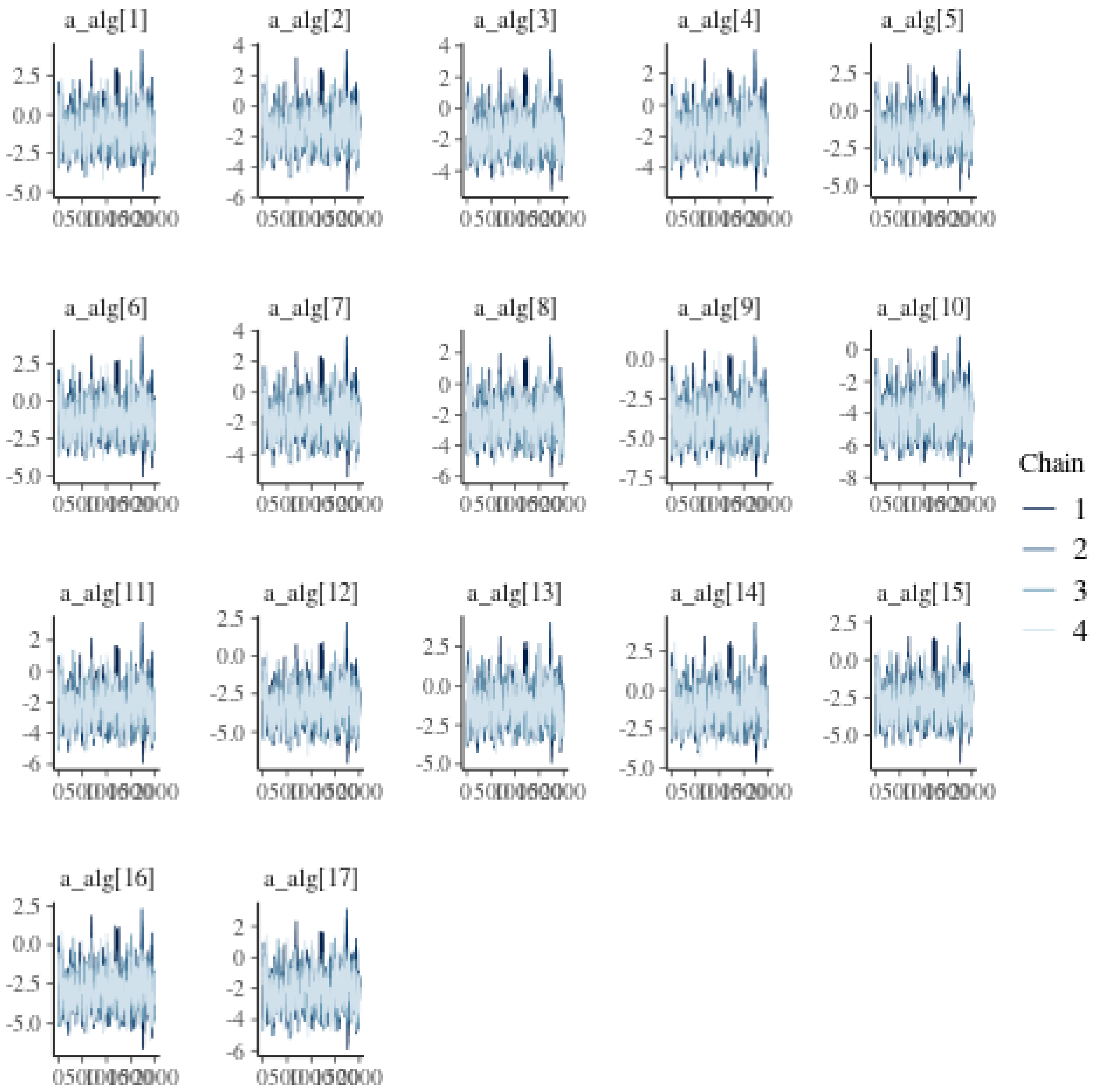

Figure 1. Simulated MCMC chains for the strength parameters (aggregated data).

4. Posterior Predictive Cheks

For posterior predictive checks, we investigate how well the mean error rate is captured by the posterior distribution. Figures 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24 shows the posterior distribution of the mean for each algorithm and the Figures show that the mean is well captured by the posterior distribution.

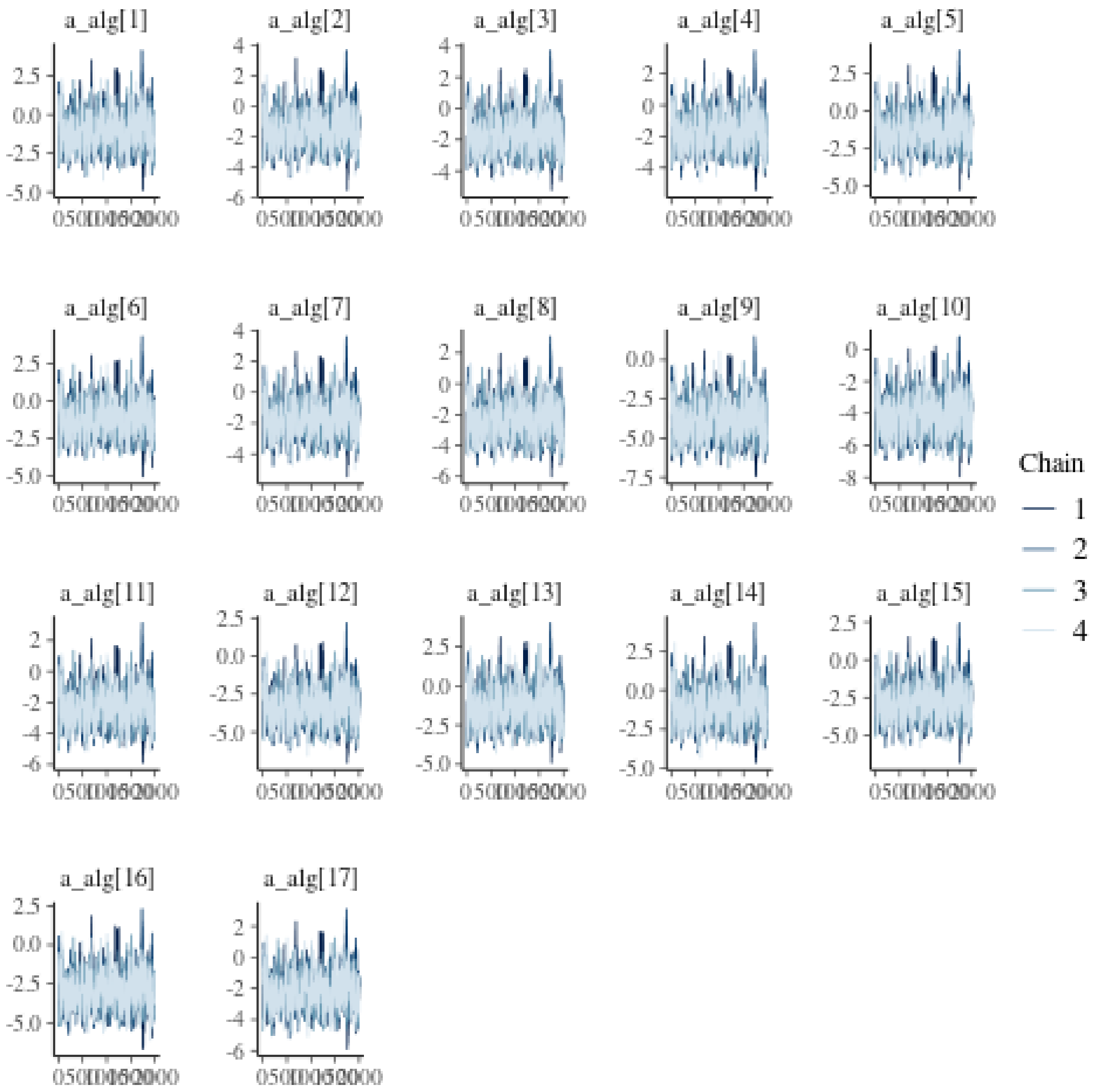

Figure 2. Simulated MCMC chains for the effect parameters (aggregated data).

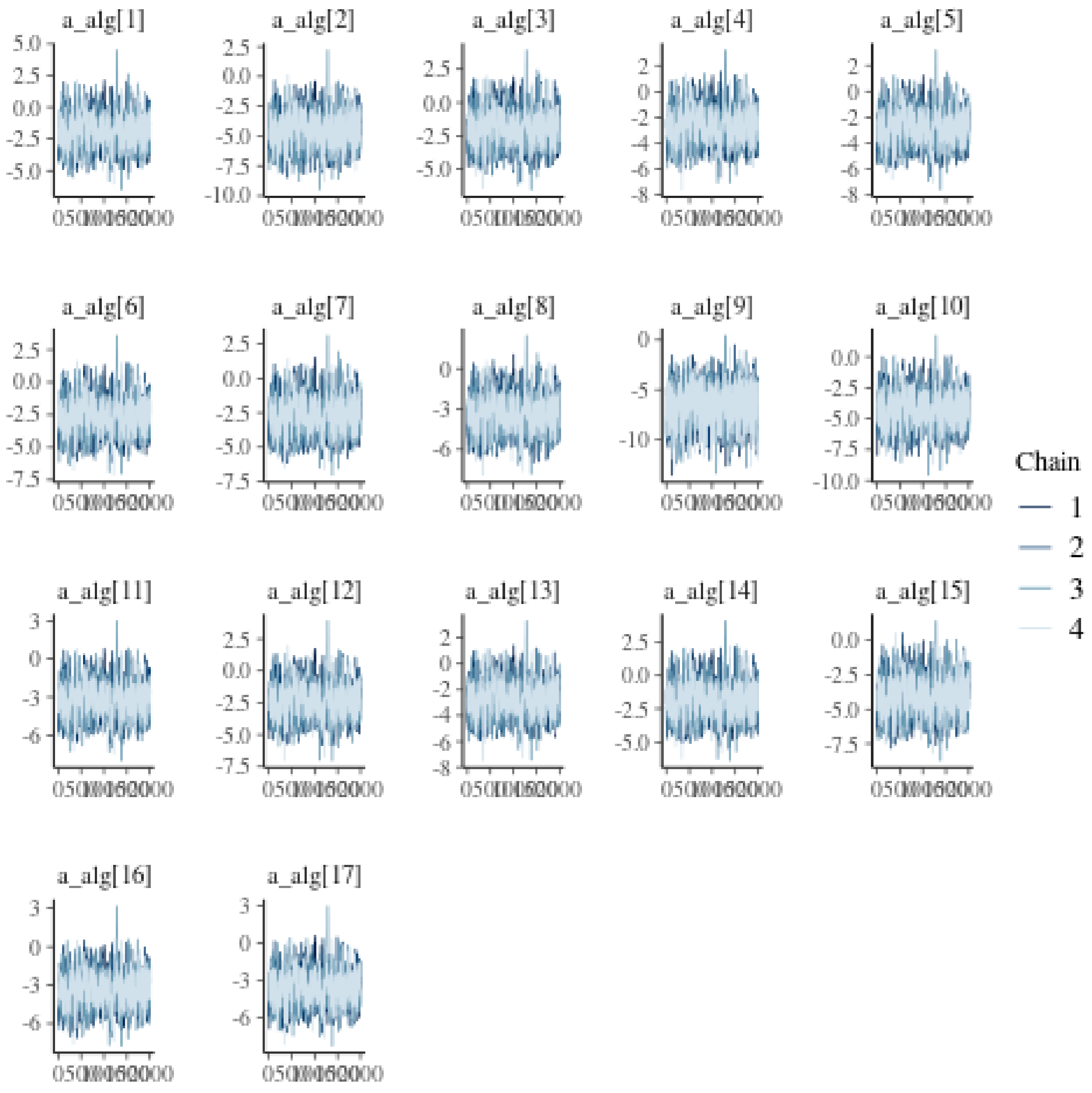

Figure 3. Simulated MCMC chains for the strength parameters (image data).

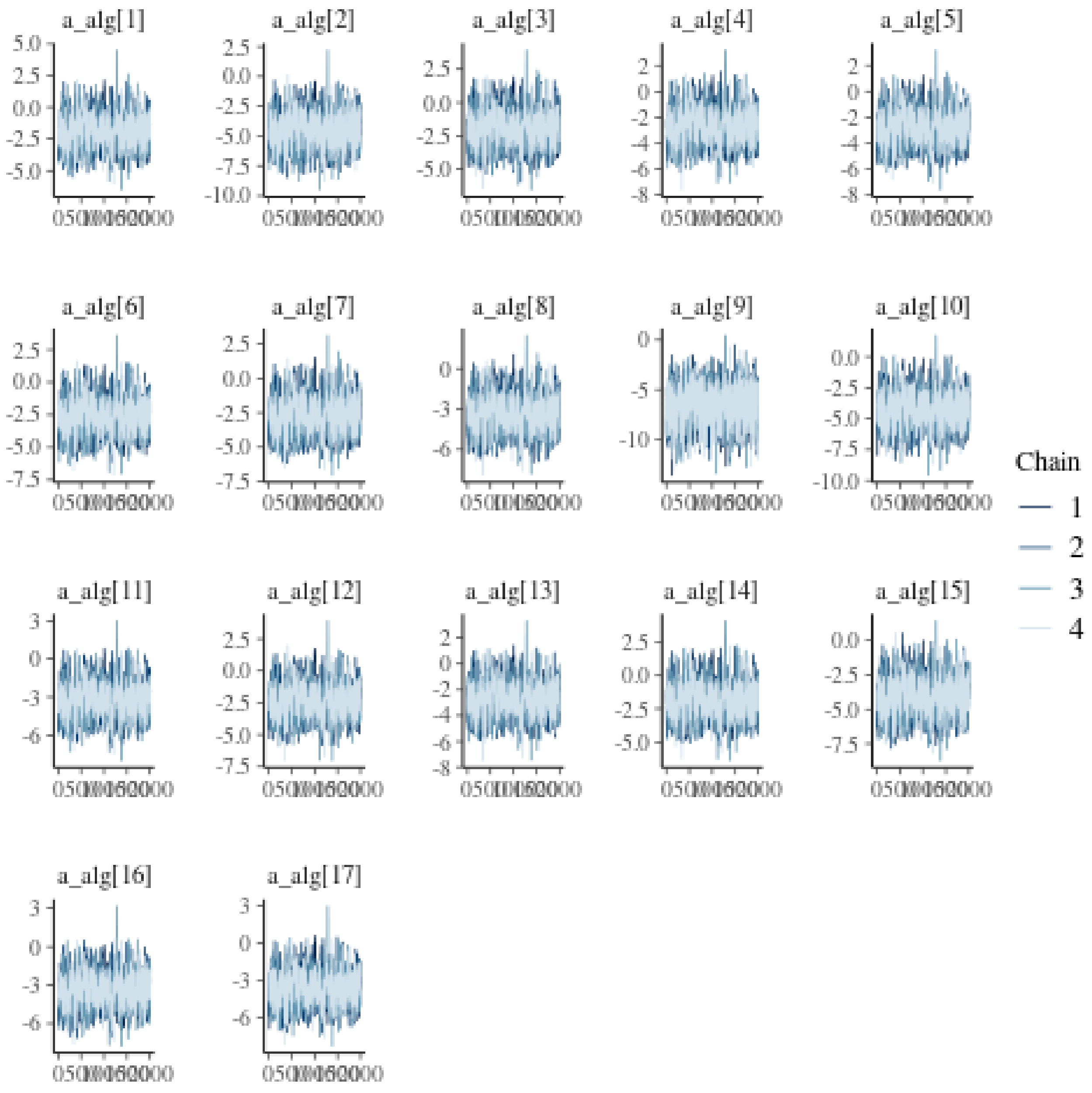

Figure 4. Simulated MCMC chains for the effect parameters (image data).

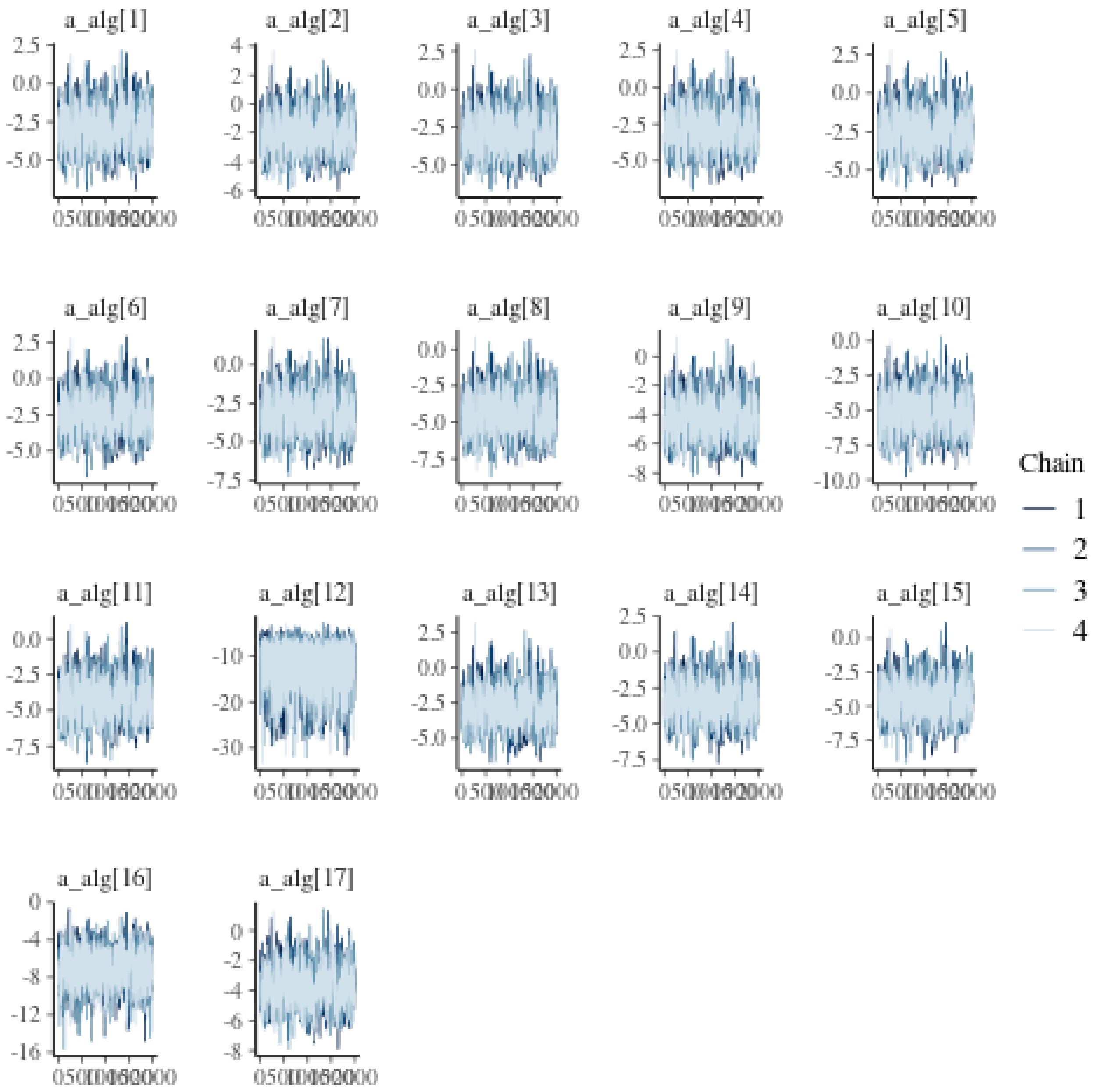

FIGURE 5. Simulated MCMC chains for the strength parameters (text data).

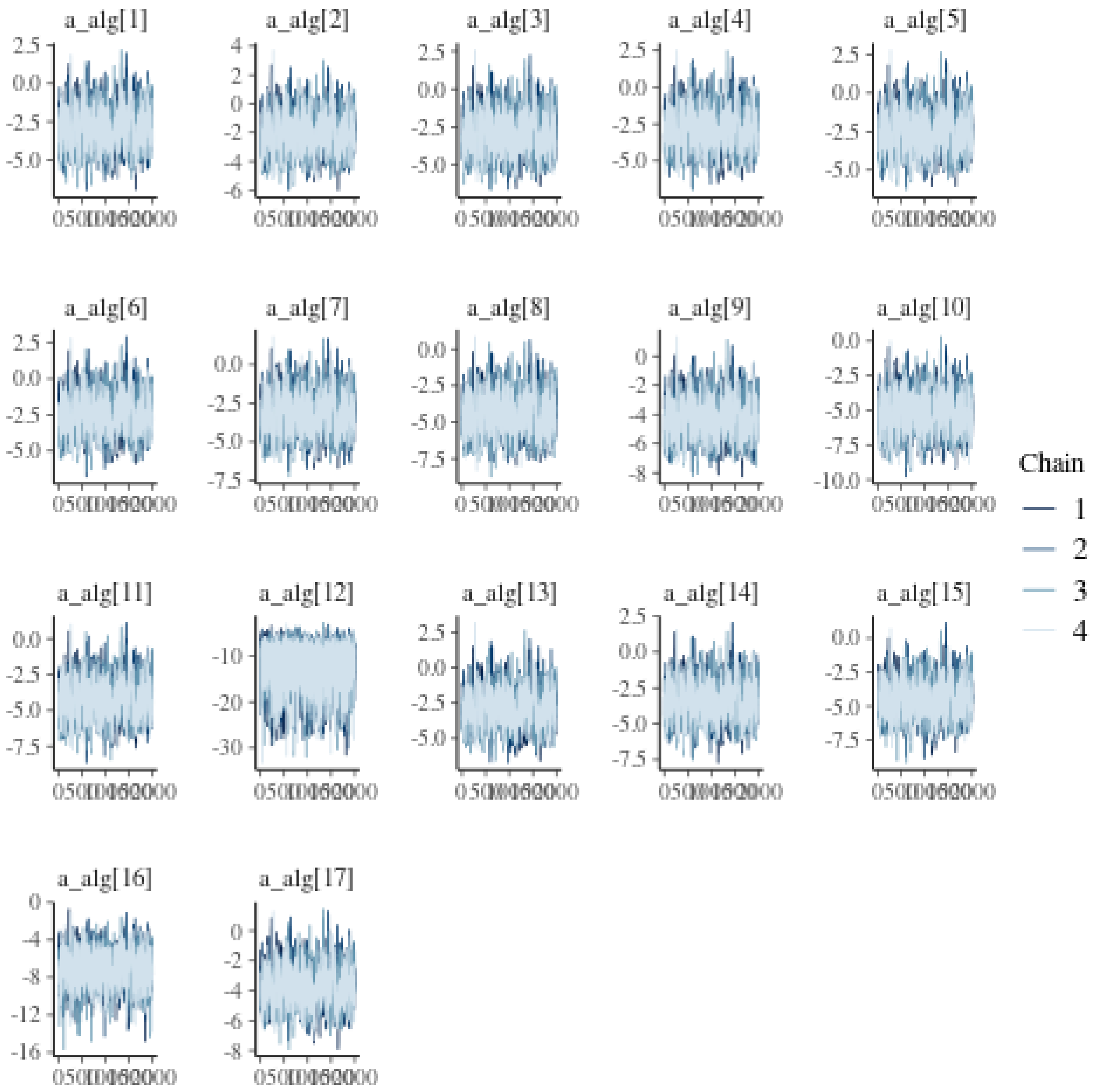

FIGURE 6. Simulated MCMC chains for the effect parameters (text data).

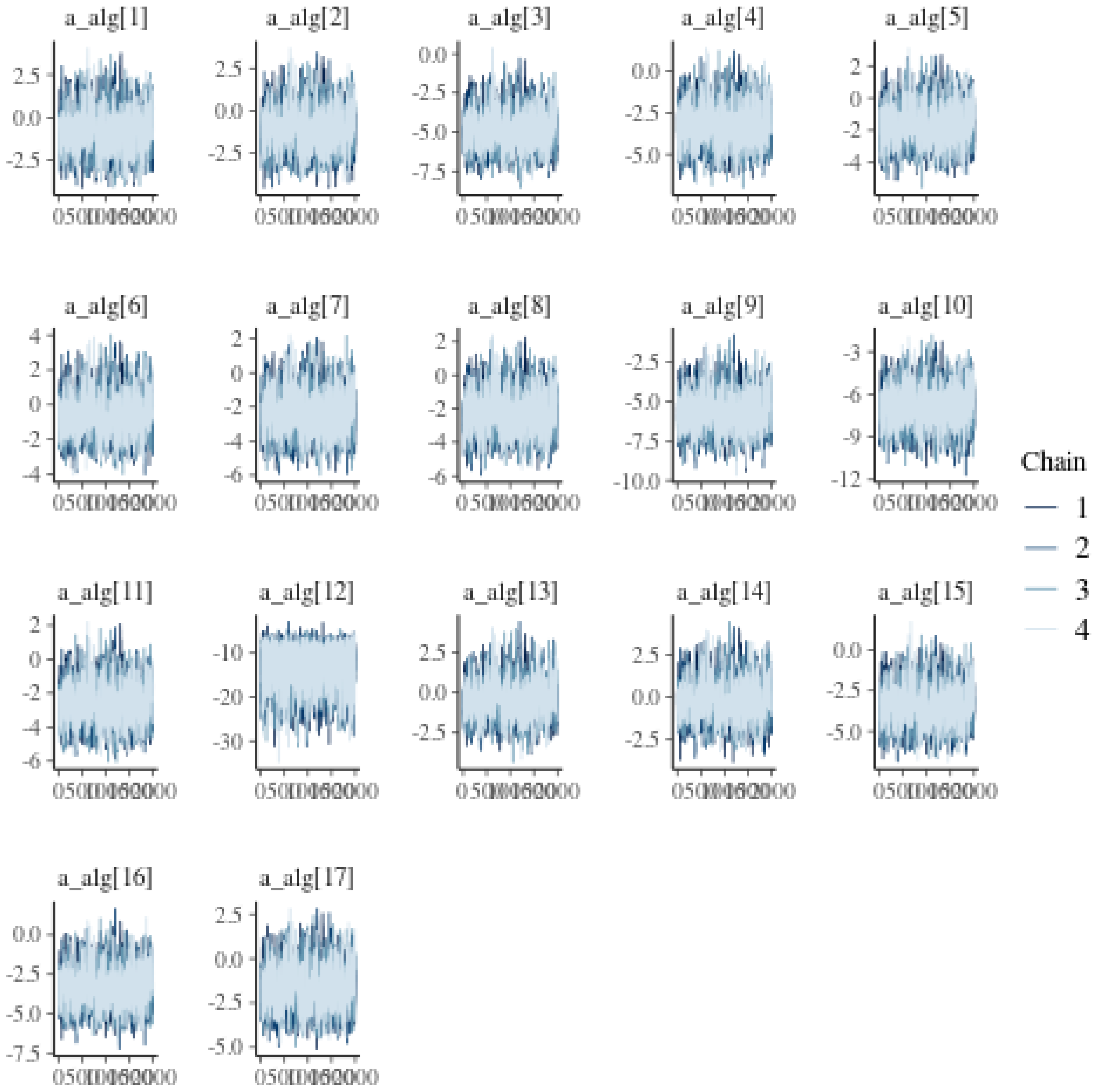

FIGURE 7. Simulated MCMC chains for the strength parameters (audio data).

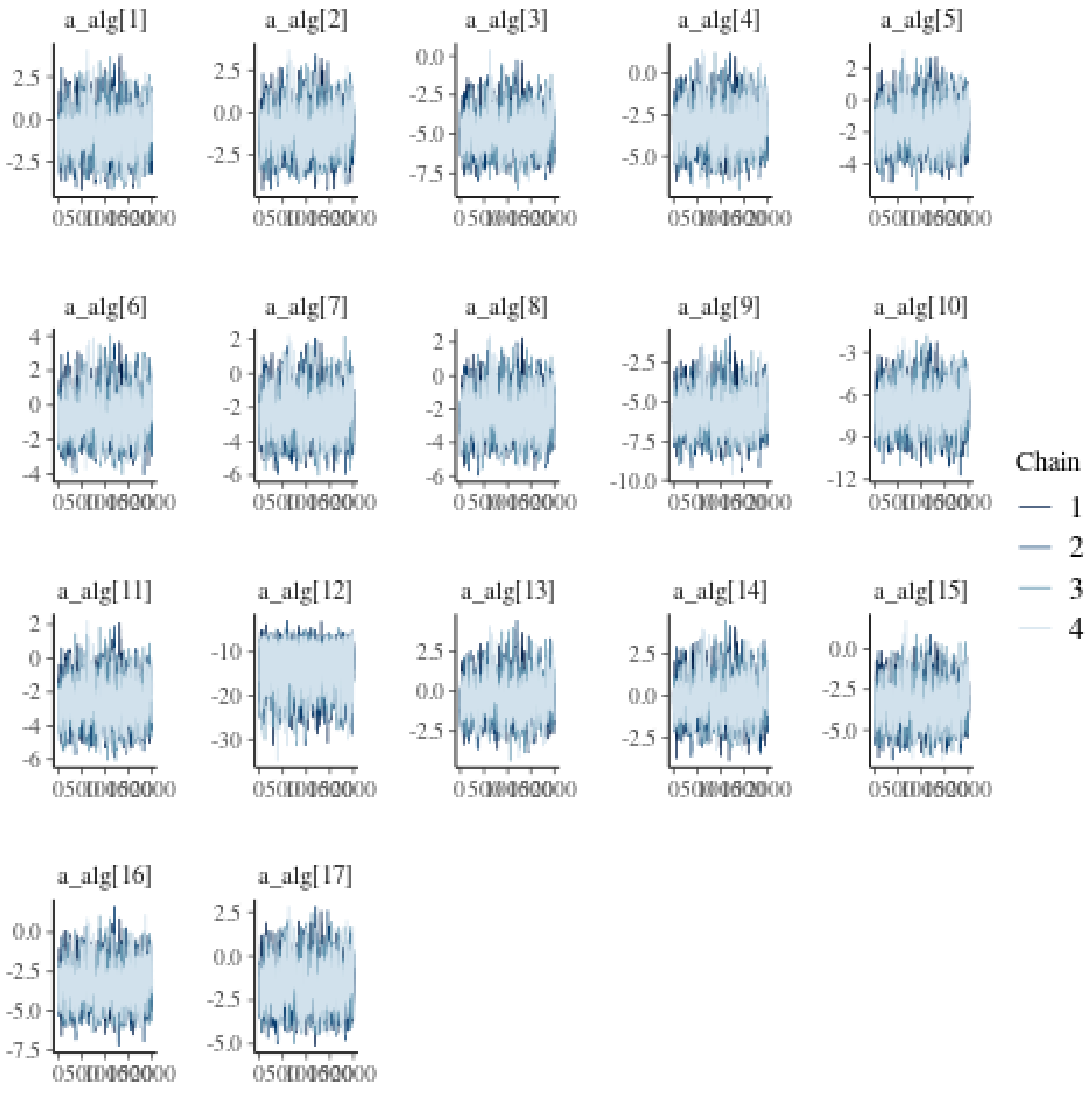

FIGURE 8. Simulated MCMC chains for the effect parameters (audio data).

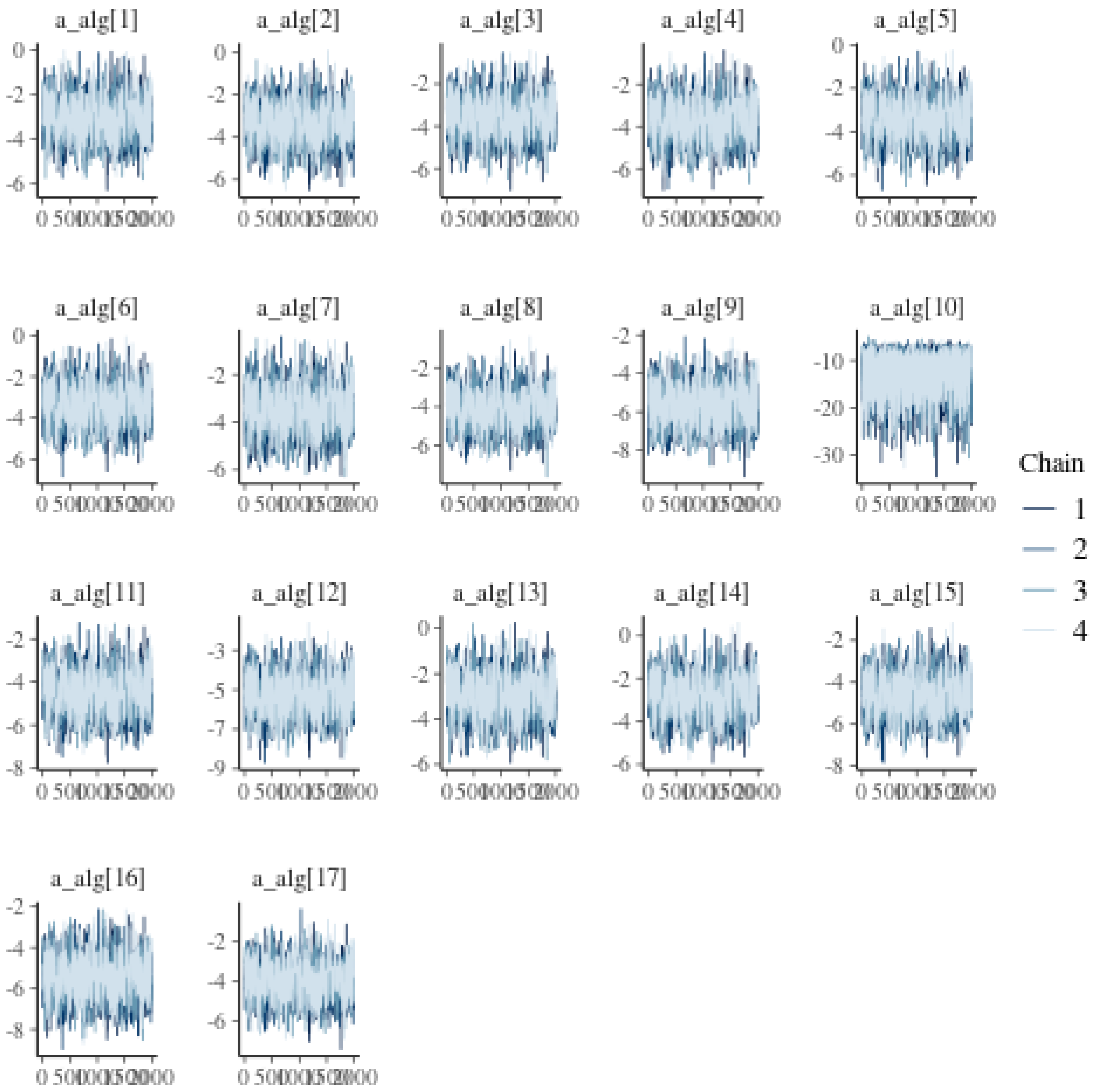

FIGURE 9. Simulated MCMC chains for the strength parameters (small allocation of data).

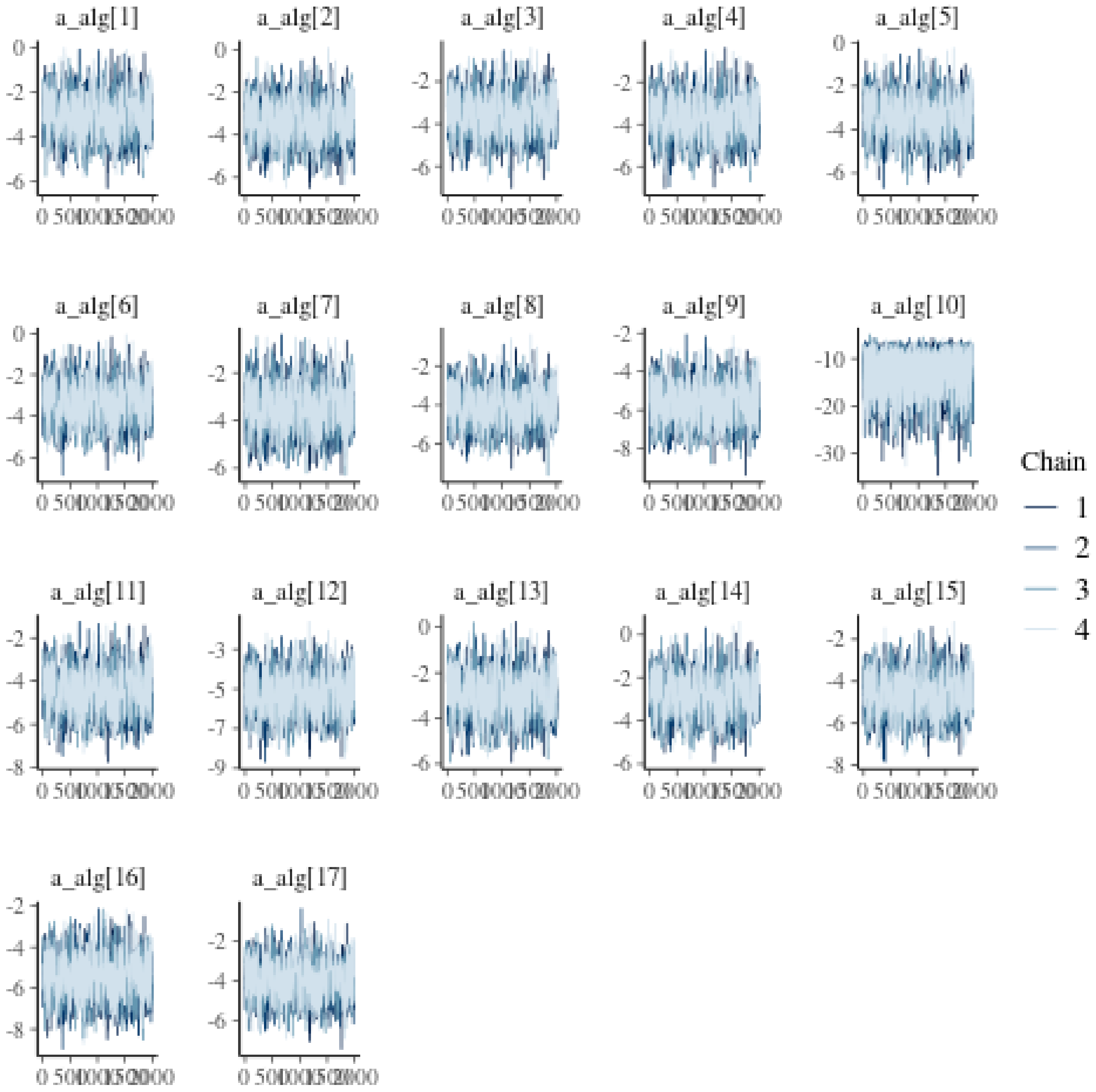

FIGURE 10. Simulated MCMC chains for the effect parameters a_i (small allocation of data)

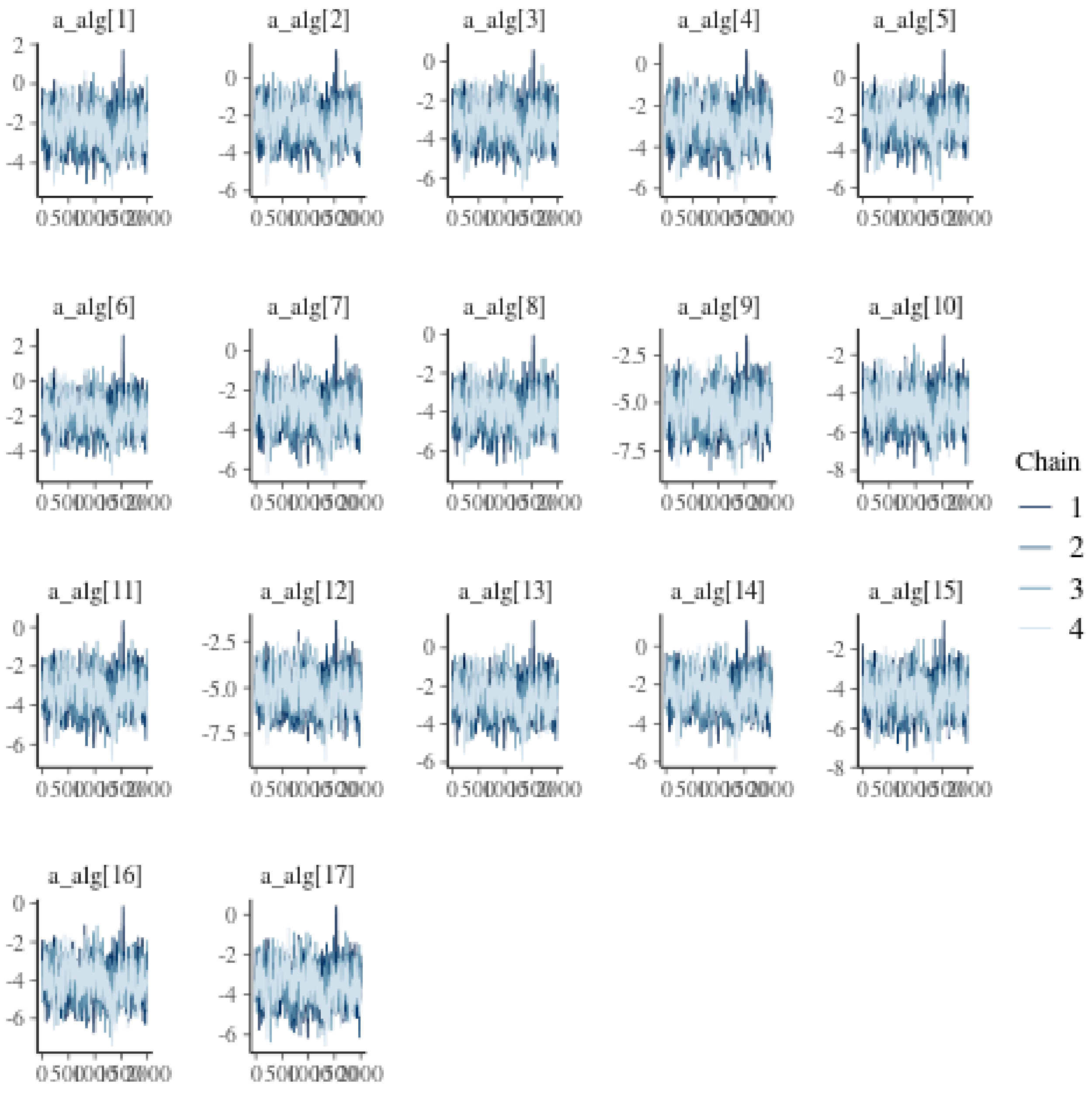

FIGURE 11. Simulated MCMC chains for the strength parameters (large allocation of data).

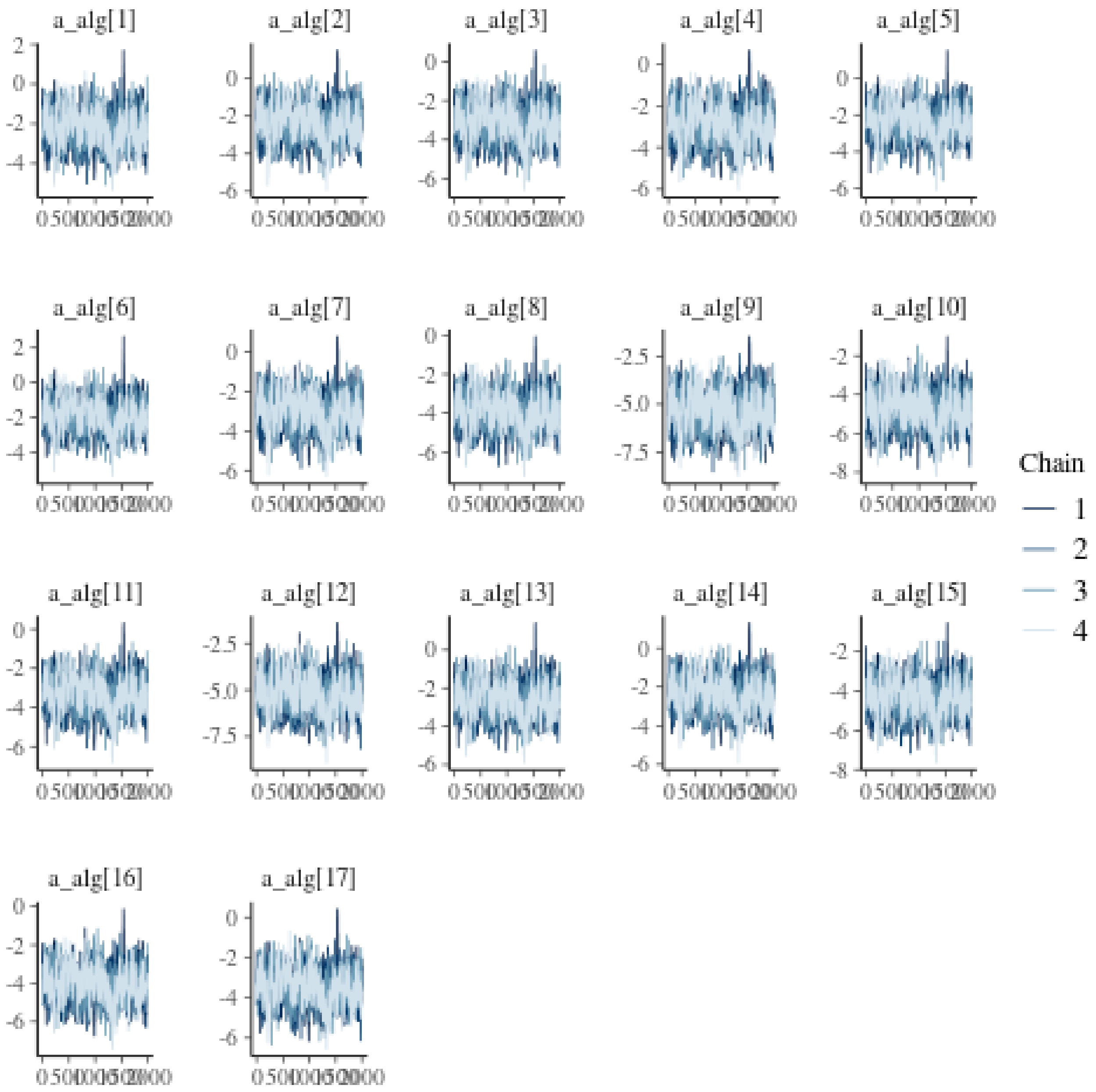

Figure 12. Simulated MCMC chains for the strength parameters (large allocation of data).

Table 1. Diagnostics for the posterior ranks (aggregated)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.000	4363.596	4389.268
a_alg[2]	1.002	4310.666	4704.940
a_alg[3]	1.002	4506.676	4641.121
a_alg[4]	1.000	4511.913	5175.336
a_alg[5]	1.001	4179.552	5407.158
a_alg[6]	1.000	4421.190	4853.736
a_alg[7]	1.000	4431.419	4697.169
a_alg[8]	1.000	4429.212	4462.071
a_alg[9]	1.001	4518.639	4809.640
a_alg[10]	1.001	4581.093	4876.947
a_alg[11]	1.001	4409.779	5259.614
a_alg[12]	1.001	4903.602	5323.972
a_alg[13]	1.001	4561.385	5199.520
a_alg[14]	1.000	4635.099	4744.294
a_alg[15]	1.001	4431.587	5018.196
a_alg[16]	1.000	4328.026	4618.103
a_alg[17]	1.001	4493.896	4915.240

Table 2. Diagnostics for the posterior ranks (image datatype)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.001	5569.827	4926.560
a_alg[2]	1.000	5384.131	5833.872
a_alg[3]	1.000	5453.536	5380.084
a_alg[4]	1.000	4758.031	5284.870
a_alg[5]	1.000	4738.275	4432.089
a_alg[6]	1.000	5421.547	5321.455
a_alg[7]	1.000	5538.582	5961.318
a_alg[8]	1.000	5458.517	5340.038
a_alg[9]	1.000	5314.767	5418.474
a_alg[10]	1.000	5526.175	5679.398
a_alg[11]	1.000	5530.860	6031.030
a_alg[12]	1.000	4838.689	5512.851
a_alg[13]	1.000	5119.155	5921.655
a_alg[14]	1.000	5301.094	5466.499
a_alg[15]	1.000	5222.151	5542.210
a_alg[16]	1.000	4757.067	5744.937
a_alg[17]	1.001	5019.732	5372.783

Table 3. Diagnostics for the posterior ranks (text datatype)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.001	3950.014	5012.816
a_alg[2]	1.001	3967.102	4947.424
a_alg[3]	1.001	4034.655	5302.354
a_alg[4]	1.001	4110.053	4768.079
a_alg[5]	1.000	4124.764	5175.381
a_alg[6]	1.001	4085.174	4895.489
a_alg[7]	1.001	3927.361	4756.549
a_alg[8]	1.001	4303.844	4836.293
a_alg[9]	1.001	4407.529	5267.432
a_alg[10]	1.000	4110.240	4805.624
a_alg[11]	1.001	4178.923	5332.661
a_alg[12]	1.001	4659.343	5859.878
a_alg[13]	1.000	4142.509	5047.237
a_alg[14]	1.001	3822.927	4777.553
a_alg[15]	1.001	3859.451	4303.521
a_alg[16]	1.000	4199.281	5302.039
a_alg[17]	1.001	4283.987	5291.580

Table 4. Diagnostics for the posterior ranks (audio datatype)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.002	8472.718	5296.022
a_alg[2]	1.000	9903.935	5834.777
a_alg[3]	1.000	9027.025	5998.672
a_alg[4]	1.000	9146.187	5669.856
a_alg[5]	1.001	8125.924	5011.553
a_alg[6]	1.001	8687.339	5981.227
a_alg[7]	1.000	8628.567	5128.435
a_alg[8]	1.000	8184.104	5895.332
a_alg[9]	1.001	8197.165	6069.792
a_alg[10]	1.001	7761.948	5446.681
a_alg[11]	1.001	8807.185	5967.099
a_alg[12]	1.001	8913.463	6830.820
a_alg[13]	1.001	8633.998	5469.254
a_alg[14]	1.000	8748.116	5787.659
a_alg[15]	1.001	8118.402	5409.835
a_alg[16]	1.001	8752.863	6160.065
a_alg[17]	1.000	9061.207	5681.185

Table 5. Diagnostics for the posterior ranks (small allocation)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.000	4735.897	4112.013
a_alg[2]	1.000	4793.097	4387.702
a_alg[3]	1.001	4694.239	4751.959
a_alg[4]	1.001	4675.820	4336.875
a_alg[5]	1.001	4835.159	4383.011
a_alg[6]	1.000	4435.438	4177.937
a_alg[7]	1.000	4555.595	4329.844
a_alg[8]	1.000	4474.490	4002.949
a_alg[9]	1.000	4748.329	4499.903
a_alg[10]	1.000	4955.164	4542.496
a_alg[11]	1.001	5020.975	4230.675
a_alg[12]	1.001	5478.078	4698.688
a_alg[13]	1.001	4471.626	3814.573
a_alg[14]	1.001	4459.145	4289.903
a_alg[15]	1.000	4488.908	4743.348
a_alg[16]	1.001	4503.813	4560.218
a_alg[17]	1.001	4666.474	3592.517

Table 6. Diagnostics for the posterior ranks (large allocation)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.000	9164.379	5867.408
a_alg[2]	1.001	9215.643	5816.067
a_alg[3]	1.000	9621.344	5915.346
a_alg[4]	1.000	9415.307	5461.207
a_alg[5]	1.000	9607.589	5958.987
a_alg[6]	1.001	8800.023	5865.614
a_alg[7]	1.000	9822.973	6074.739
a_alg[8]	1.001	8850.728	5606.480
a_alg[9]	1.000	8739.513	5854.369
a_alg[10]	1.003	9746.317	5910.605
a_alg[11]	1.001	8805.959	5627.071
a_alg[12]	1.001	9520.813	6643.876
a_alg[13]	1.001	8688.475	6033.553
a_alg[14]	1.001	8779.775	6017.763
a_alg[15]	1.000	9485.947	5613.972
a_alg[16]	1.001	9502.301	6621.634
a_alg[17]	1.001	9140.726	5586.761

Table 7. Diagnostics for the posterior effects (aggregated)

variable	rhat	ess_bulk	ess_tail
a_alg[1]	1.006	440.182	541.622
a_alg[2]	1.006	439.851	507.177
a_alg[3]	1.006	437.147	522.022
a_alg[4]	1.006	442.510	523.726
a_alg[5]	1.006	440.479	535.330
a_alg[6]	1.005	439.403	531.731
a_alg[7]	1.006	441.416	530.607
a_alg[8]	1.006	443.915	552.155
a_alg[9]	1.006	455.324	549.392
a_alg[10]	1.005	454.539	548.766
a_alg[11]	1.005	436.243	532.145
a_alg[12]	1.005	446.804	552.072
a_alg[13]	1.006	443.509	522.109
a_alg[14]	1.006	436.285	518.366
a_alg[15]	1.005	447.754	514.499
a_alg[16]	1.005	446.589	518.547
a_alg[17]	1.006	438.401	525.700

Table 8. Diagnostics for the posterior effects (image datatype)

Variable	Rhat	ess_bulk	ess_tail
a_alg[1]	1.002	694.306	1165.395
a_alg[2]	1.002	848.293	1373.391
a_alg[3]	1.002	711.569	1215.024
a_alg[4]	1.002	705.012	1163.763
a_alg[5]	1.002	714.125	1100.587
a_alg[6]	1.001	696.947	1189.039
a_alg[7]	1.002	696.537	1208.653
a_alg[8]	1.002	746.903	1307.071
a_alg[9]	1.001	1298.246	2323.478
a_alg[10]	1.001	753.645	1406.142
a_alg[11]	1.002	703.356	1172.664
a_alg[12]	1.002	684.666	1234.366
a_alg[13]	1.002	713.963	1315.901
a_alg[14]	1.002	716.643	1163.742
a_alg[15]	1.002	766.597	1409.595
a_alg[16]	1.002	728.488	1288.008
a_alg[17]	1.001	723.604	1322.541

Table 9. Diagnostics for the posterior effects (text datatype)

variable	rhat	ess_bulk	ess_tail
a_alg[1]	1.011	600.486	901.568
a_alg[2]	1.011	619.436	935.044
a_alg[3]	1.012	613.667	910.521
a_alg[4]	1.011	628.431	891.711
a_alg[5]	1.012	605.999	902.718
a_alg[6]	1.012	612.309	950.841
a_alg[7]	1.012	612.771	850.828
a_alg[8]	1.011	636.753	905.439
a_alg[9]	1.011	649.210	961.074
a_alg[10]	1.010	693.142	1043.443
a_alg[11]	1.010	629.357	961.024
a_alg[12]	1.000	4197.822	4444.723
a_alg[13]	1.011	599.856	811.524
a_alg[14]	1.011	626.391	906.396
a_alg[15]	1.009	646.797	888.692
a_alg[16]	1.005	1014.138	1357.790
a_alg[17]	1.012	620.882	895.637

Table 10. Diagnostics for the posterior effects (audio datatype)

variable	rhat	ess_bulk	ess_tail
a_alg[1]	1.003	870.761	1374.035
a_alg[2]	1.004	849.701	1351.964
a_alg[3]	1.003	770.054	1325.369
a_alg[4]	1.003	833.502	1442.465
a_alg[5]	1.003	842.617	1358.068
a_alg[6]	1.003	770.569	1398.594
a_alg[7]	1.002	862.501	1389.156
a_alg[8]	1.003	845.793	1565.068
a_alg[9]	1.003	841.420	1339.853
a_alg[10]	1.003	1028.792	1773.255
a_alg[11]	1.003	839.701	1495.961
a_alg[12]	1.001	4585.525	4785.559
a_alg[13]	1.003	764.878	1392.853
a_alg[14]	1.003	799.794	1362.592
a_alg[15]	1.003	832.628	1402.115
a_alg[16]	1.003	830.624	1467.762
a_alg[17]	1.003	879.117	1481.837

Table 11. Diagnostics for the posterior effects (small allocation)

variable	rhat	ess_bulk	ess_tail
a_alg[1]	1.003	614.066	1004.753
a_alg[2]	1.004	618.395	1045.639
a_alg[3]	1.003	627.256	1019.701
a_alg[4]	1.004	616.507	1038.961
a_alg[5]	1.004	617.869	955.493
a_alg[6]	1.004	615.979	1151.694
a_alg[7]	1.003	630.325	1006.059
a_alg[8]	1.004	632.262	1038.843
a_alg[9]	1.003	717.147	1206.127
a_alg[10]	1.001	3588.698	3392.161
a_alg[11]	1.003	623.126	1198.584
a_alg[12]	1.003	684.711	1284.887
a_alg[13]	1.003	623.283	1096.985
a_alg[14]	1.003	611.306	1008.557
a_alg[15]	1.003	648.600	1073.667
a_alg[16]	1.002	694.782	1184.188
a_alg[17]	1.004	629.064	1135.959

Table 12. Diagnostics for the posterior effects (large allocation)

variable	rhat	ess_bulk	ess_tail
a_alg[1]	1.023	258.317	463.035
a_alg[2]	1.021	268.310	538.120
a_alg[3]	1.021	253.876	458.608
a_alg[4]	1.023	248.612	503.228
a_alg[5]	1.022	258.077	465.426
a_alg[6]	1.023	255.613	533.282
a_alg[7]	1.022	252.546	457.869
a_alg[8]	1.021	263.317	584.556
a_alg[9]	1.019	301.839	709.768
a_alg[10]	1.018	289.796	539.473
a_alg[11]	1.021	264.149	504.409
a_alg[12]	1.021	281.814	665.613
a_alg[13]	1.022	257.686	480.975
a_alg[14]	1.022	252.151	576.368
a_alg[15]	1.020	262.272	499.345
a_alg[16]	1.020	266.885	570.303
a_alg[17]	1.023	259.107	516.268

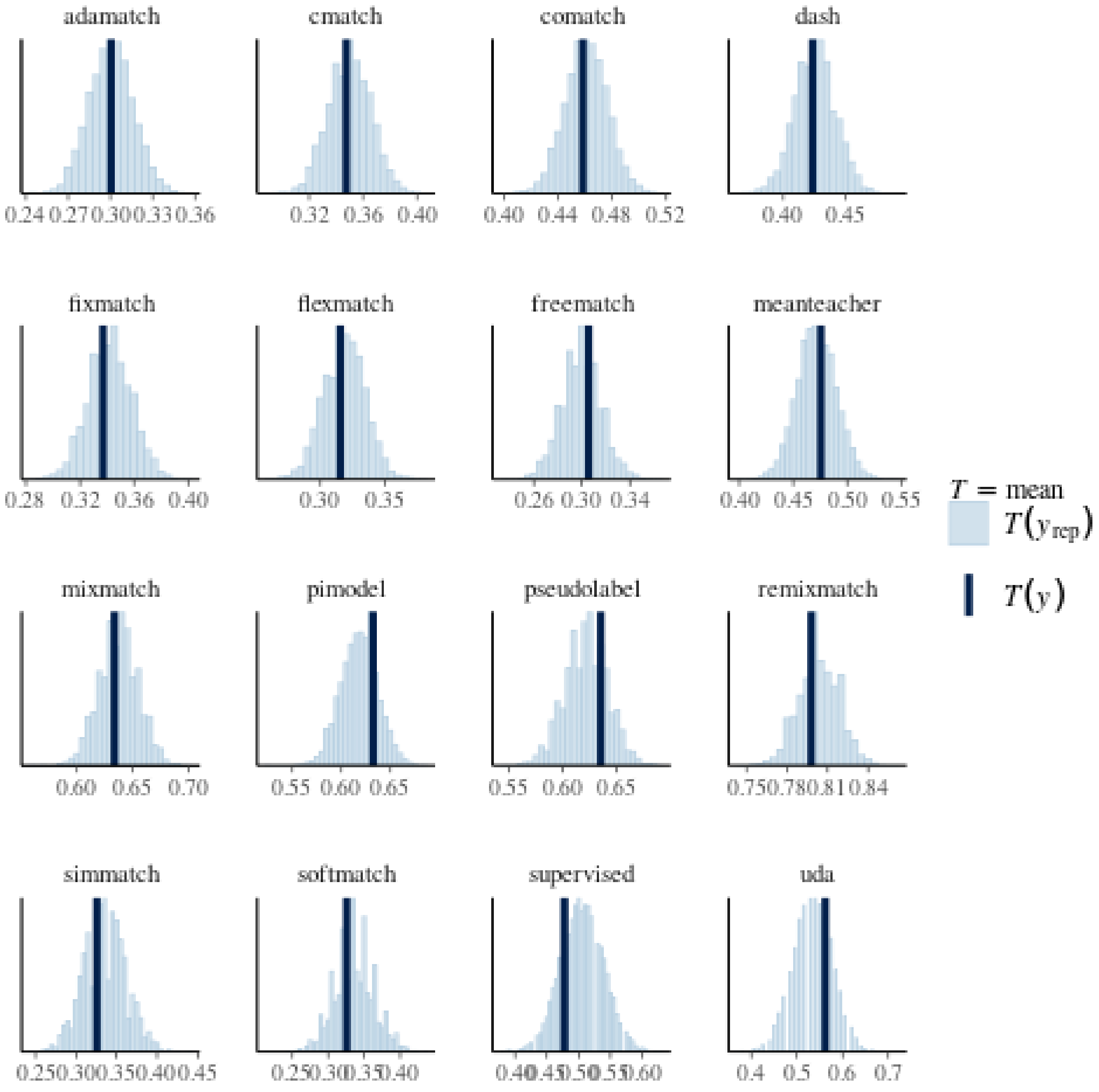

FIGURE 13. Posteririor predictice check of the ranks for each algorithm (aggregated)

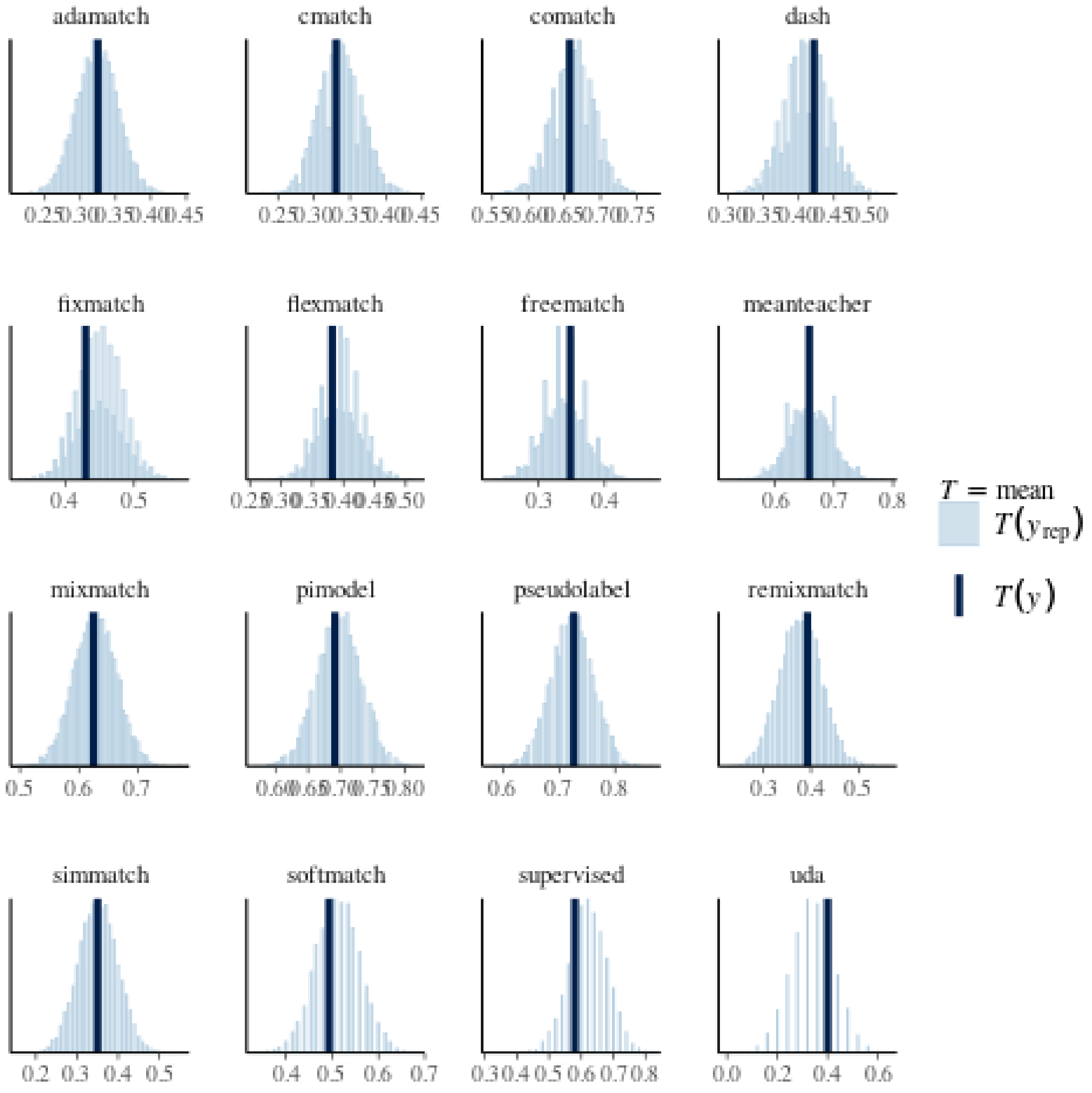

FIGURE 14. Posteririor predictice check of the ranks for each algorithm (image datatype)

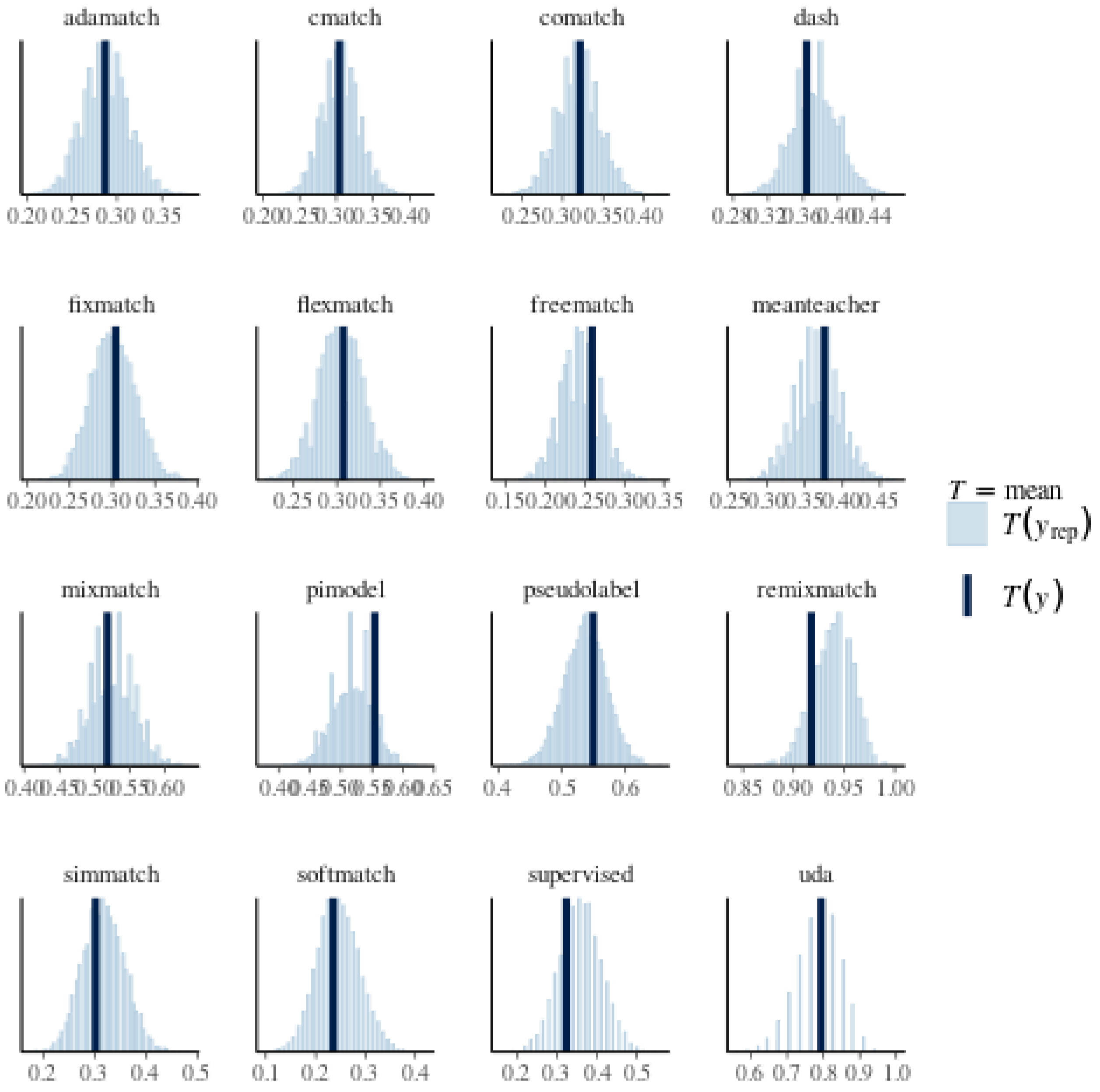

FIGURE 15. Posteririor predictice check of the ranks for each algorithm (text datatype)

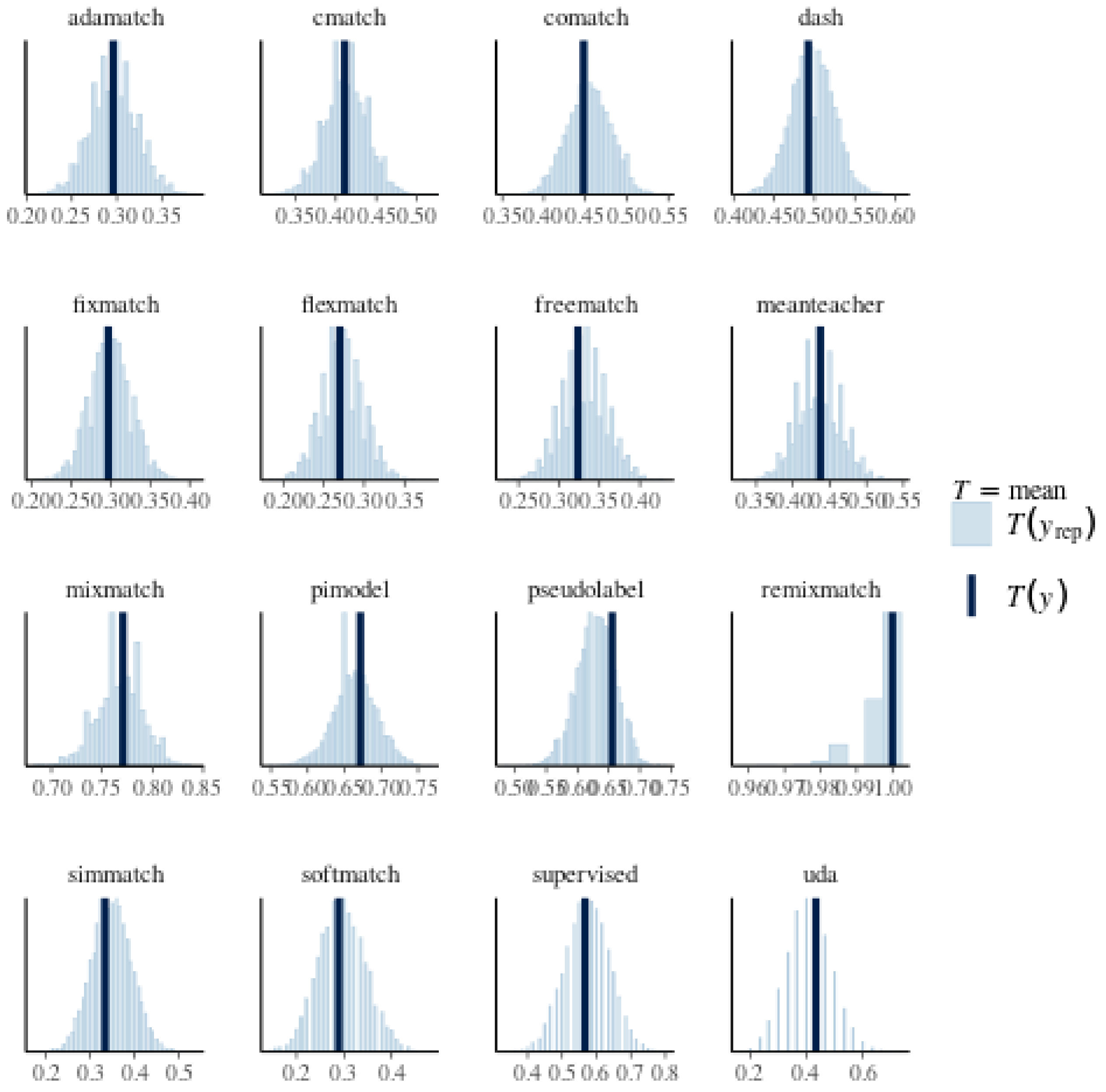

Figure 16. Posteririor predictice check of the ranks for each algorithm (audio datatype)

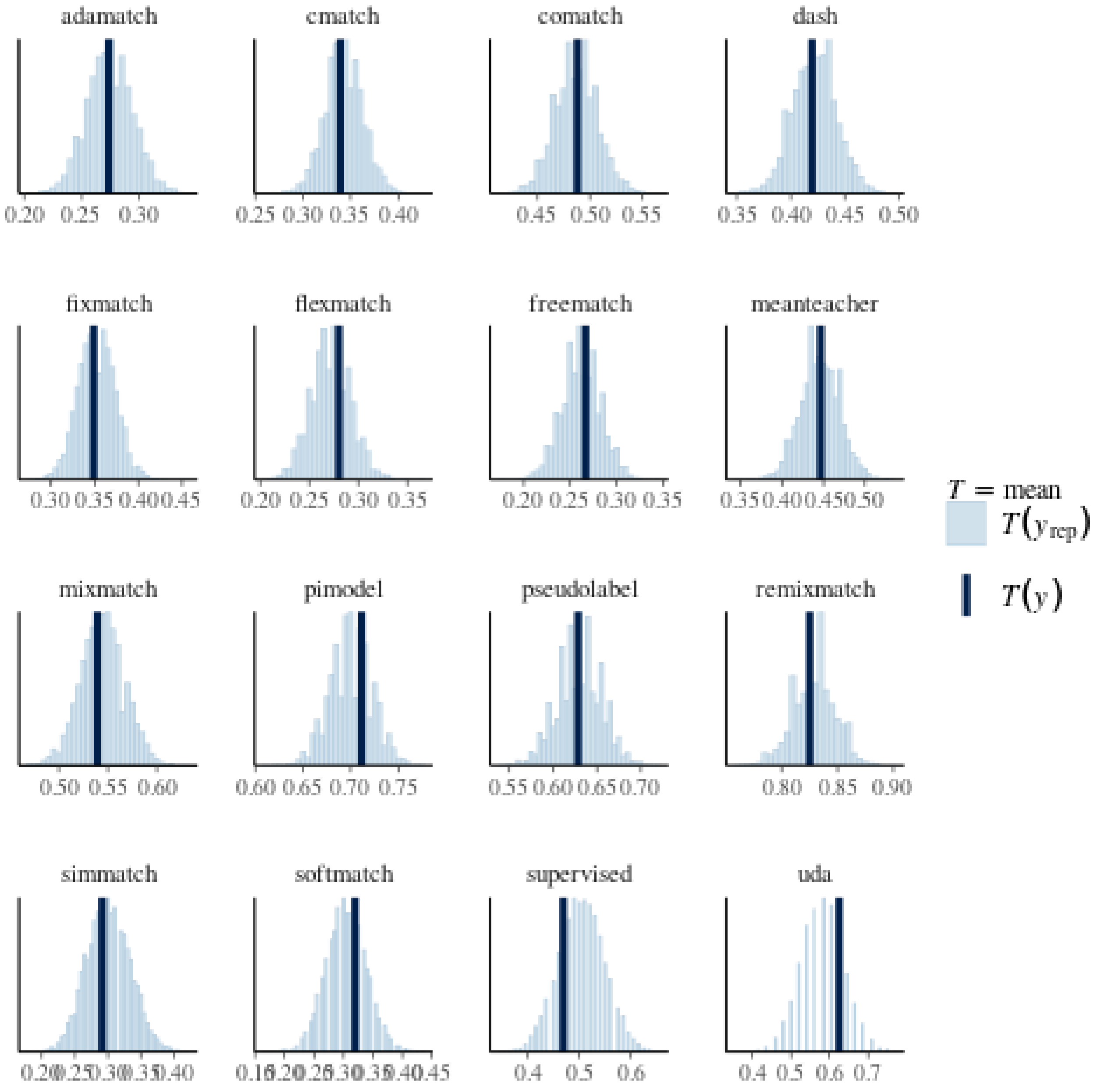

FIGURE 17. Posteririor predictice check of the ranks for each algorithm (small allocation)

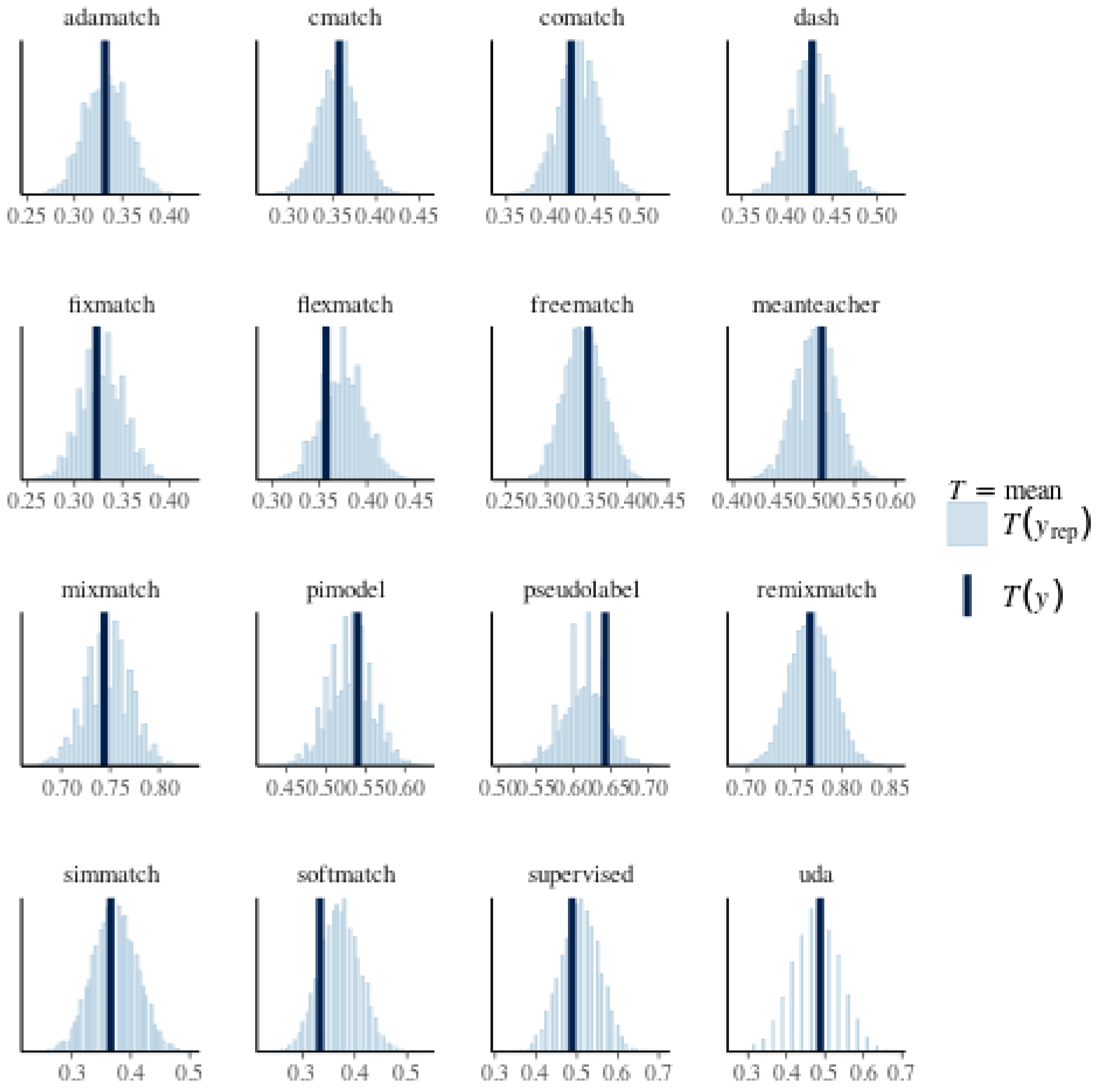

Figure 18. Posteririor predictice check of the ranks for each algorithm (large allocation)

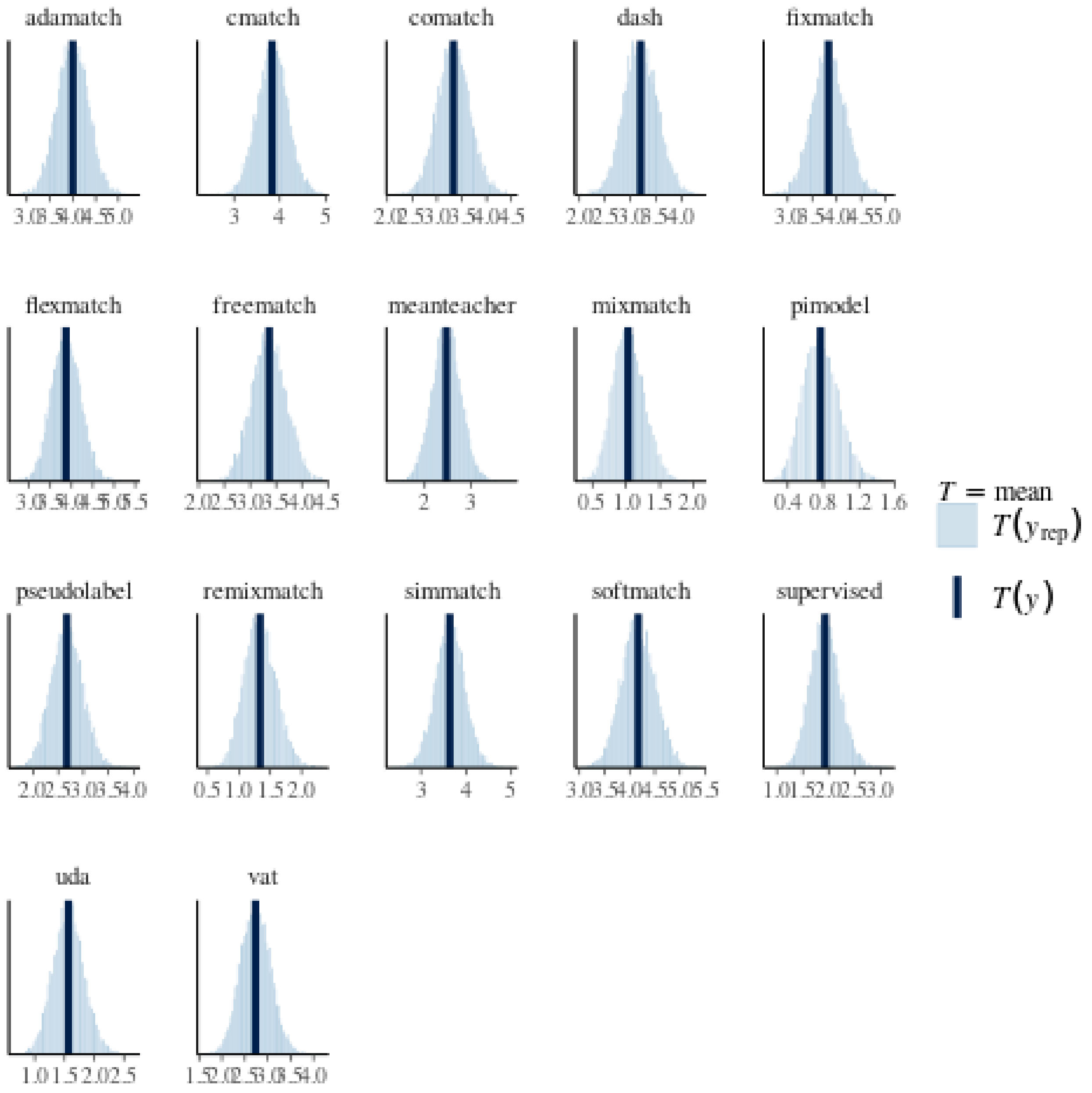

FIGURE 19. Posteririor predictice check of the effects for each algorithm (aggregated)

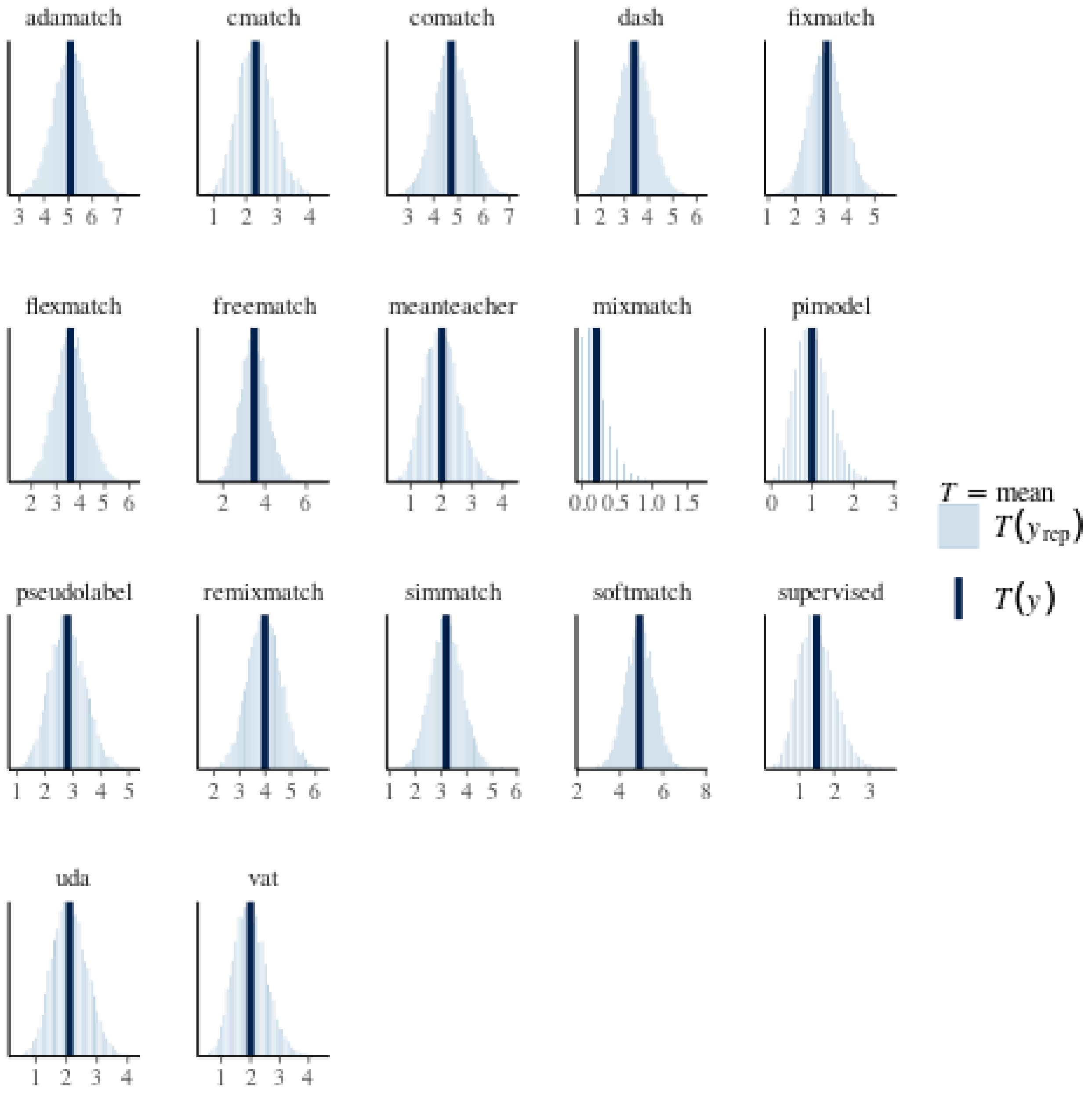

FIGURE 20. Posteririor predictice check of the effects for each algorithm (image datatype)

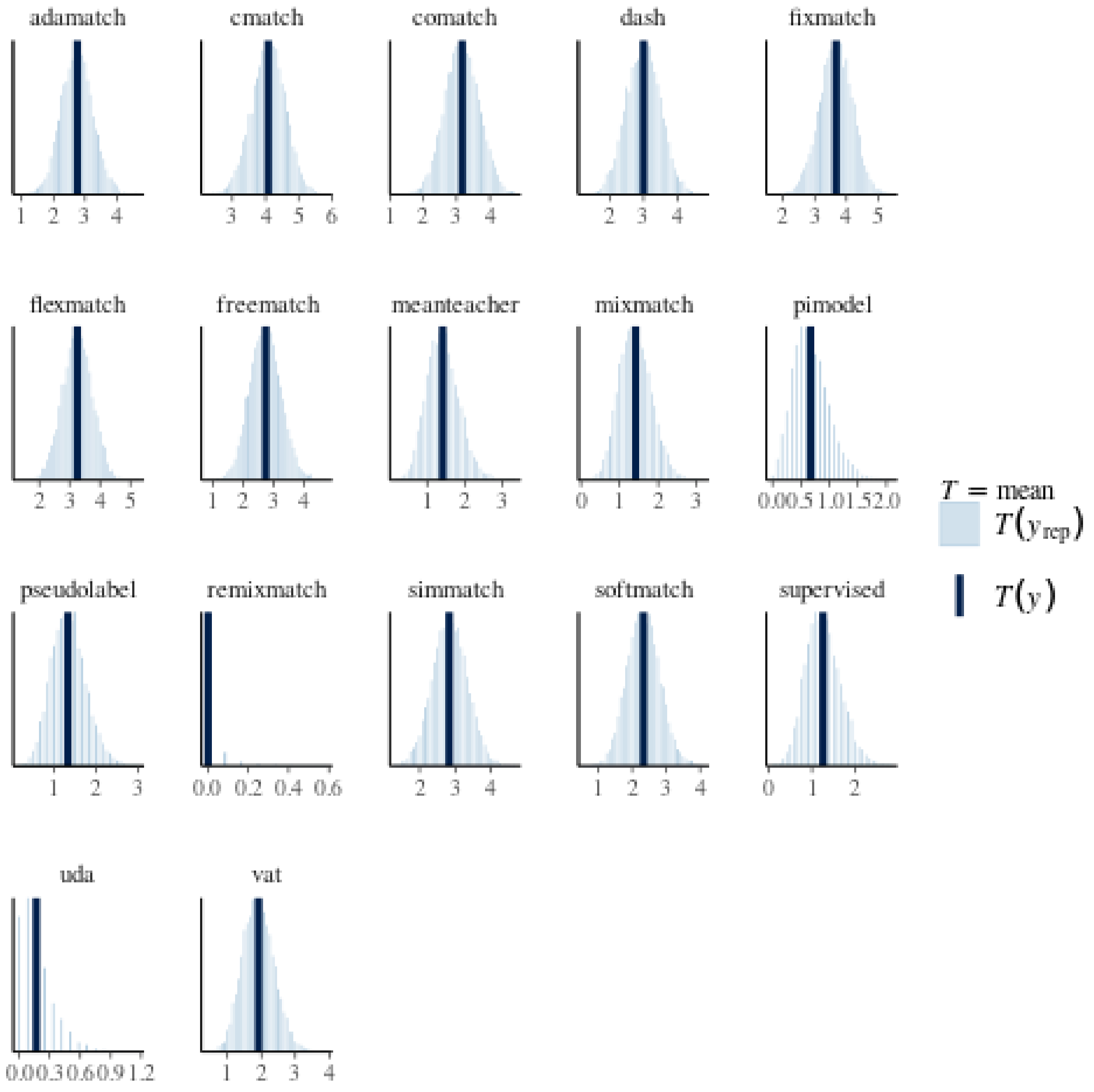

FIGURE 21. Posteririor predictice check of the effects for each algorithm (text datatype)

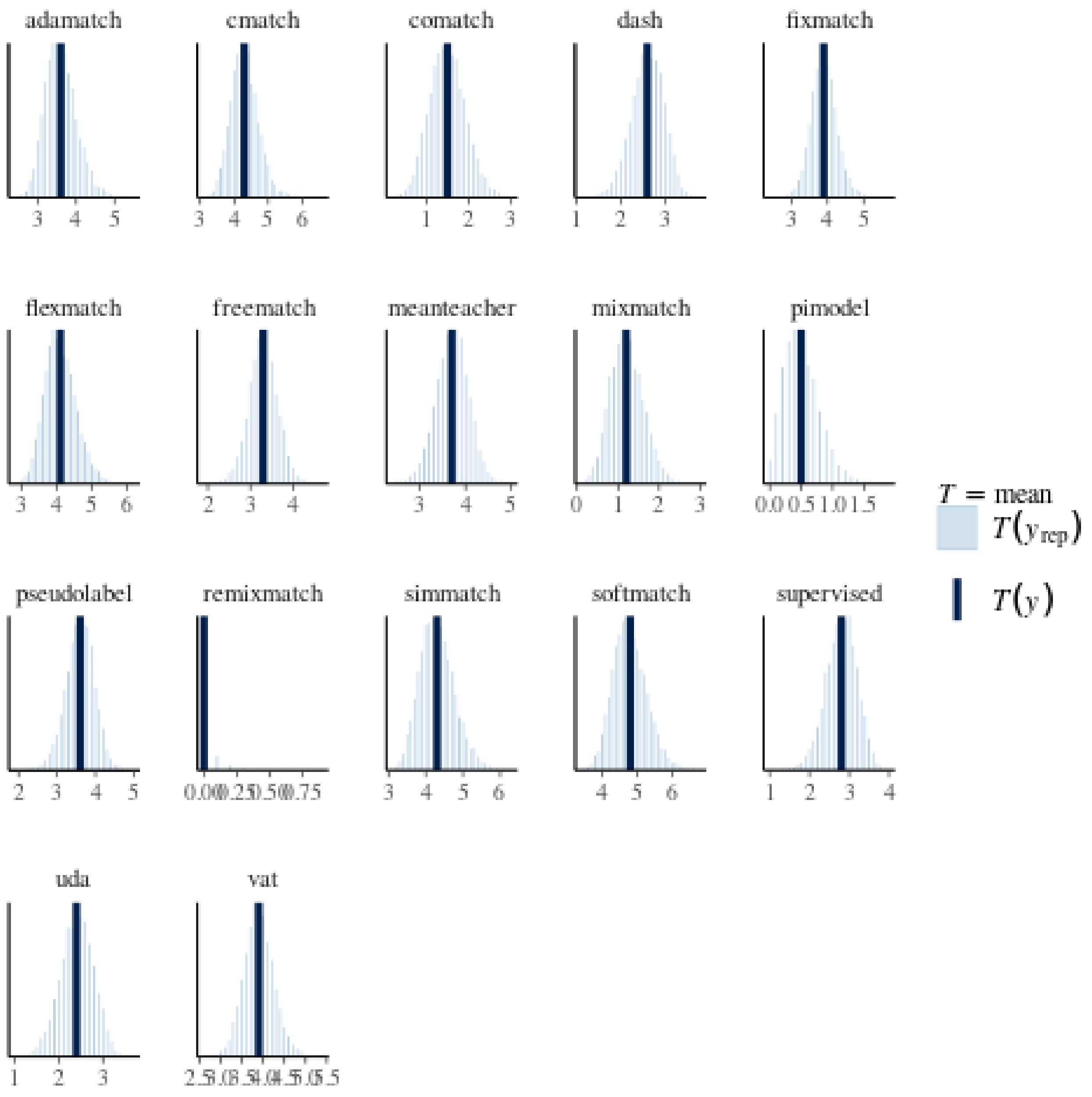

FIGURE 22. Posteririor predictice check of the effects for each algorithm (audio datatype)

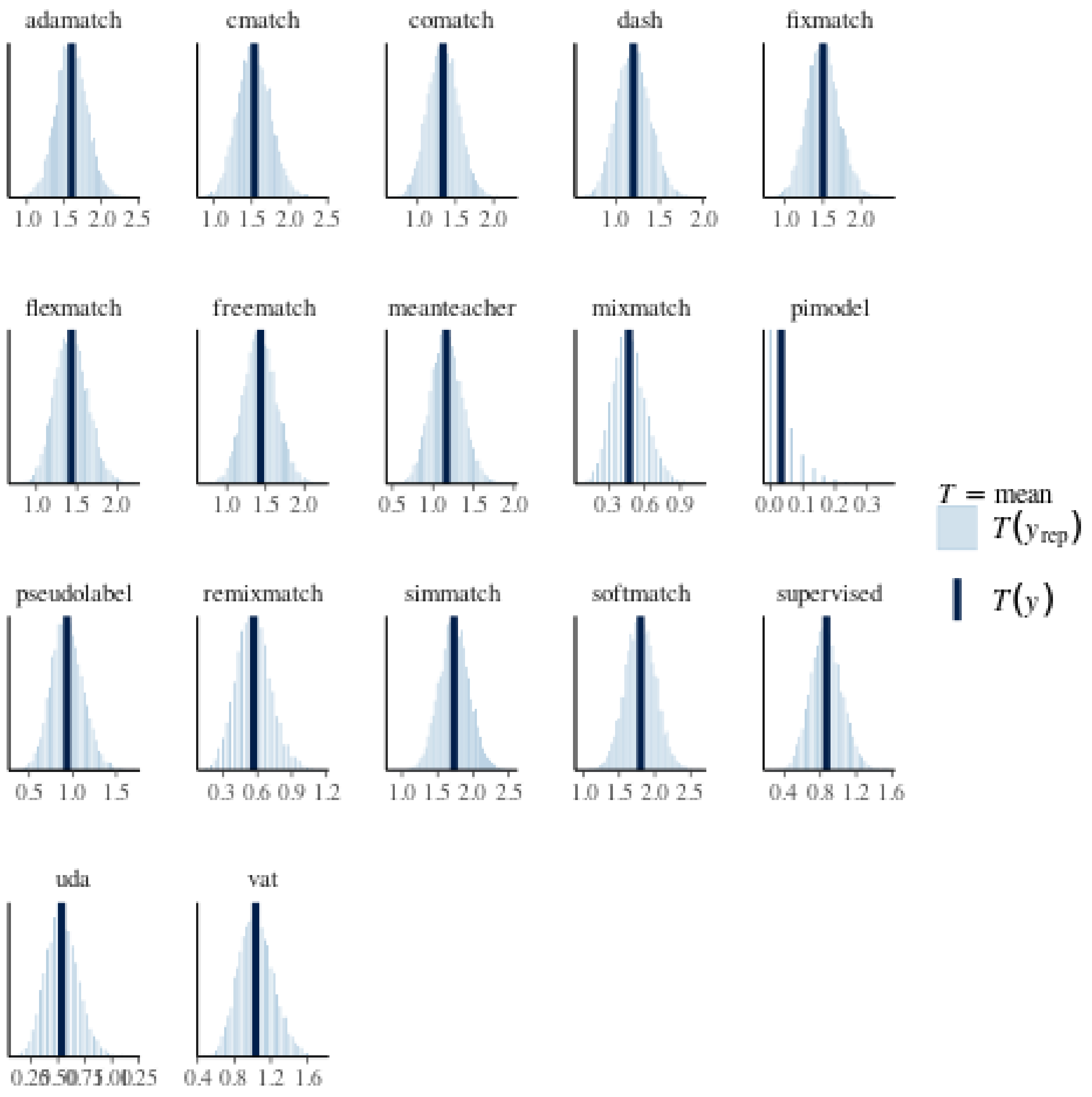

FIGURE 23. Posterior predictive check of the effects for each algorithm (small allocation)

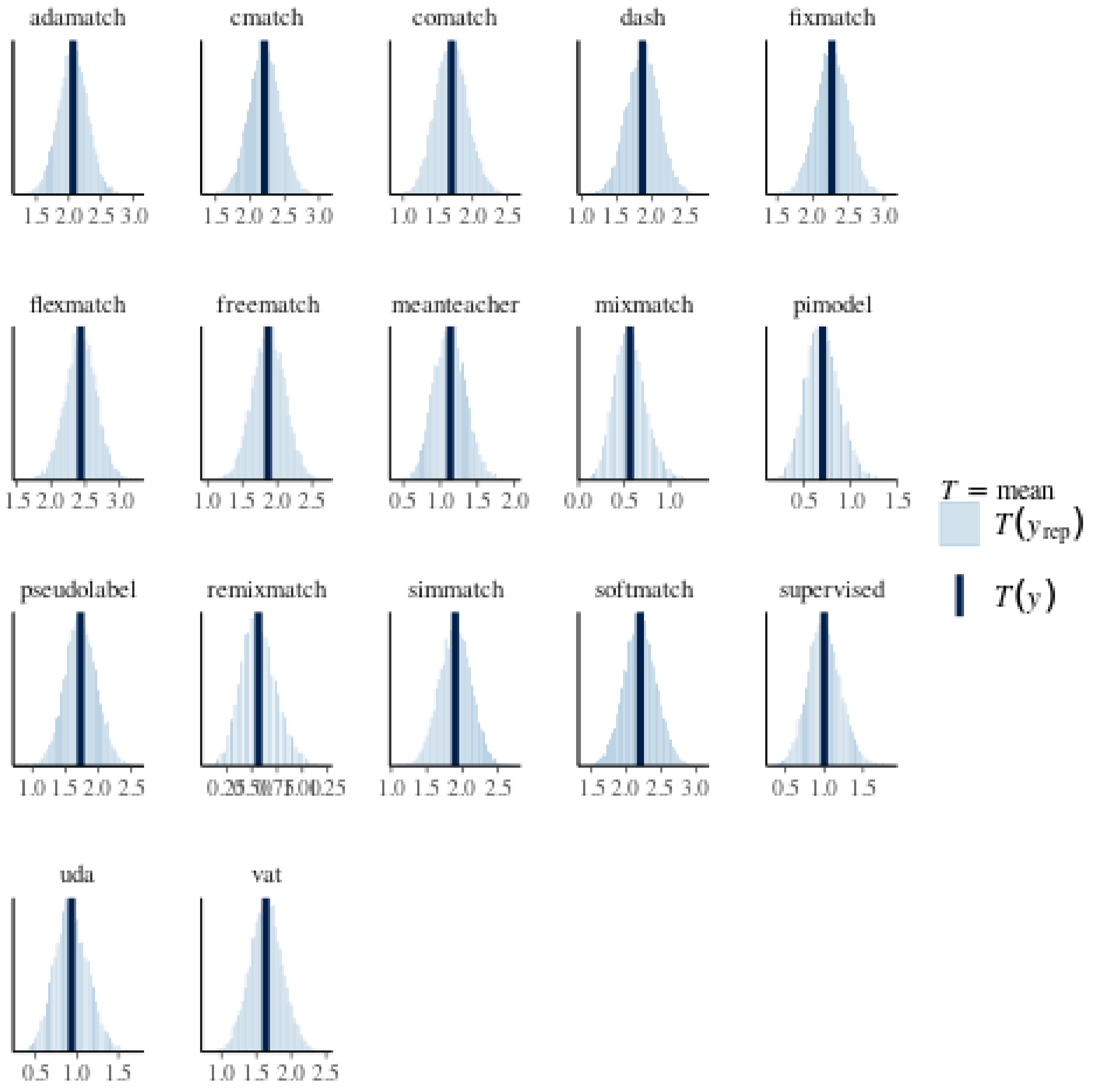

FIGURE 24. Posteririor predictice check of the effects for each algorithm (large allocation)

References

[1] Carlo A Furia, Robert Feldt, and Richard Torkar. Bayesian data analysis in empirical software engineering research. *IEEE Transactions on Software Engineering*, 47(9):1786–1810, 2019.

[2] Jonah Gabry, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman. Visualization in bayesian workflow. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 182(2):389–402, 2019.

[3] David Issa Mattos, Jan Bosch, and Helena Holmström Olsson. Statistical models for the analysis of optimization algorithms with benchmark functions. *IEEE Transactions on Evolutionary Computation*, 25(6):1163–1177, 2021.

[4] Richard McElreath. *Statistical rethinking: A Bayesian course with examples in R and Stan.* CRC press, 2020.