Modeling Images using Transformed Indian Buffet Processes

Yuening Hu
Ke Zhai†
Department of Computer Science, University of Maryland, College Park, MD USA

Jordan Boyd-Graber
iSchool and UMIACS, University of Maryland, College Park, MD USA

Sinead Williamson
Department of Machine Learning, Carnegie Mellon University, Pittsburgh, PA USA

Abstract
Latent feature models are attractive for image modeling, since images generally contain multiple objects. However, many latent feature models ignore that objects can appear at different locations or require pre-segmentation of images. While the transformed Indian buffet process (tIBP) provides a method for modeling transformation-invariant features in unsegmented binary images, its current form is inappropriate for real images because of its computational cost and modeling assumptions. We combine the tIBP with likelihoods appropriate for real images and develop an efficient inference, using the cross-correlation between images and features, that is theoretically and empirically faster than existing inference techniques. Our method discovers reasonable components and achieve effective image reconstruction in natural images.

1. Introduction
Latent feature models assume data are generated by combining latent features shared across the dataset and aim to learn this latent structure in an unsupervised manner. Such models typically assume all properties of a feature are common to all data points—i.e., each feature appears in exactly the same way across all observations. This is often a reasonable assumption. For example, microarray data are designed so each cell consistently corresponds to a specific condition.

This does not hold for images. Consider a collection of images of a rolling ball. If a model must create new features to explain the ball’s every position, it will devote less attention to other aspects of the image and will be unable to generalize across the ball’s path. Instead, we would like some properties of a feature, e.g., shape, to be shared across data points but properties, e.g., location, to be observation-specific.

Models that generalize across images to discover transformation-invariant features have many applications. Image tracking, for instance, discovers mislaid bags or illegally stopped cars. Image reconstruction restores partially corrupted images. Movie compression recognizes recurring image patches and caches them across frames.

We argue that latent feature models of images should:

• Discover features needed to model data and add additional features to model new data.
• Generalize across transformations so features can have different locations, scales, and orientations.
• Handle properties of real images such as occlusion.

A nonparametric model that comes close to our goals is the noisy-OR transformed Indian buffet process (NO-tIBP, Austerweil & Griffiths, 2010); however, its likelihood model is inappropriate for real images. Existing unsupervised models that handle realistic likelihoods (Jojic & Frey, 2001; Titsias & Williams, 2006) are parametric and cannot discover new features. In Section 2, we further describe these and other models that meet some, but not all, of our criteria.

In Section 3, we propose models that fulfill these properties by combining realistic likelihoods with nonparametric frameworks. In Section 4, we introduce novel inference algorithms that dramatically improve inference for transformed IBPs in larger datasets (Section 5). In Section 6, we show that our models can discover features and model data better than existing models. We discuss relationships with other nonparametric models and extensions in Section 7.
from location to location or where a person may be in either
the foreground or background. Naive models would learn
different features for each location a car appears in; a more
appropriate model would learn that each observation is in
fact a transformation of a common feature.

The transformed IBP (tIBP, Austerweil & Griffiths, 2010)
extends the IBP to accommodate data with varying locations.
In the tIBP, each column of an IBP-distributed matrix \( Z \) is
(as before) associated with a feature. In addition, each non-
zero element of \( Z \) is associated with a transformation \( r_{nk} \).
Transforming the features and combining them according
to a likelihood model produces observations. In the original
tIBP paper, features were generated and combined using
noisy-OR (Wood et al., 2006); we refer to this model as the
noisy-OR tIBP (NO-tIBP), which allows the same feature
to appear in different locations, scales, and orientations.

2.3. Likelihoods for Latent Feature Image Models

In addition to the noisy-OR, another likelihood that has
been used with the IBP is a linear Gaussian model, which
assumes images are generated via a linear superposition of
features (Griffiths & Ghahramani, 2005). Each IBP row
selects a subset of features and generates an observation by ad-
ditively superimposing these features and adding Gaussian
noise. This is demonstrated in Figure 1(a). This model can
be extended by adding weights to the non-zero elements of
the IBP-distributed matrix (Knowles & Ghahramani, 2007)
and incorporating a spiky noise model (Zhou et al., 2011)
appropriate for corrupted images.

If we want to model images where features can occlude
each other, linear Gaussian models are inappropriate. In
the vision community, images are often represented via
overlapping layers (Wang & Adelson, 1994), including in
generative probabilistic models (Jojic & Frey, 2001; Titsias
& Williams, 2006). In these “sprite” models, features are
Gaussian-distributed, and an ordering is defined over a set
of features. In each image, every active feature has a trans-
formation (as in the tIBP) and a binary mask for each pixel.
Given the feature order, the image is generated by taking
the value, at each pixel, of the uppermost unmasked feature.

This model is appealing. It is an intuitive occlusion model;
features have a consistent ordering; and only the topmost
feature is visible. However, this likelihood model has only
been used for parametric feature sets and on data where the
number of features is known a priori.

3. Modeling Real-valued Images

While the NO-tIBP likelihood model is incompatible with
real images, it provides a foundation for nonparametric
models with transformed features. In this section, we use
the tIBP to build models that combine nonparametric feature
models with more useful and realistic likelihood functions for real images.

We begin by providing a general representation for the transformed IBP with an arbitrary likelihood.

1. Sample a binary matrix $Z \sim \text{IBP}(\alpha)$, determining the features (columns) present in observations (rows).
2. For $k \in \mathbb{N}$, sample a feature $\phi_k \sim p(\phi)$.
3. For each image $n \in \{1, \ldots, N\}$
   - For $k \in \mathbb{N}$, sample a transformation $r_{nk} \sim p(r)$.
   - Sample an image $x_n \sim p(x|\Phi, z_n, r_n)$.

The distribution over transformations $p(r)$, the feature likelihood $p(\phi)$, and the image likelihood $p(x|\Phi, z_n, r_n)$ can be defined in various ways. In the remainder of this section, we will use this generic framework to define concrete models with a parameterization of transformations and two different likelihood models.

**Transformations** Following Austerweil & Griffiths (2010), we consider three categories of transformation: translation, rotation, and scaling. We parameterize a transformation $r : \mathbb{R}^D \rightarrow \mathbb{R}^D$ using a vector $(r_x, r_y, r_x, r_y)$. The parameters $(r_x, r_y)$ parameterize translations, and the transformed feature $r(a_k)$ is obtained by shifting each pixel in $a_k$ by $(r_x, r_y)$. Rotations are parameterized by $r_s \in [0, 2\pi)$, and scaling is parameterized by $r_s \in \mathbb{R}^+$. In practice, we restrict the possible rotations and scaling factors to a finite set, and assume a uniform prior on transformations.

**Linear Gaussian transformed IBP** Our first attempt to define a likelihood applicable to real data is based on the linear Gaussian likelihood for the IBP described in Section 2.3. Each feature $\phi_k$ is represented using a real-valued vector $a_k \sim \mathcal{N}(0, \sigma^2_\phi I)$. In each image, the transformed features are combined using superposition,

$$x_n \sim \mathcal{N}(\sum_{k=1}^{\infty} z_n, r_{nk}(a_k), \sigma^2_\phi I). \tag{1}$$

We refer to the resulting model as the linear Gaussian transformed IBP (LG-tIBP).

**Masked transformed IBP** While the LG-tIBP model is appropriate for real-valued data, it cannot handle feature occlusion. To address this problem, we propose a masked transformed IBP (M-tIBP), based on the sprite model (Section 2.3). In this model, each feature $\phi_k$ is represented by a Gaussian feature $a_k$ and a shape vector $\pi_k$. Let $\omega$ be a permutation of $\mathbb{N}$ that imposes an ordering on the features. We can interpret feature $i$ being “behind” feature $k$ if $\omega_i < \omega_k$. Each time a feature appears in an image, we sample a mask $s_{n,k}$ from the Bernoulli probabilities in the corresponding shape vector $\pi_k$. These masks “occlude” lower layers so that at each pixel; only the uppermost unmasked feature contributes to the final image.

The generative process can be described as follows. For each image $n$ and feature $k$, define an auxiliary variable $M_{n,k}$, the visibility indicator.

$$M_{n,k} = \begin{cases} 1 & \text{if } \arg\max_j \left[ \omega_j z_{n,j} \left( r_{n,k}^{-1}(d) \right) \right] = k \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The visibility indicator $M_{n,k}$, is 1 when feature $k$ is the uppermost unmasked feature at pixel $d$ in image $n$. The image and feature likelihoods for the M-tIBP are

$$a_k \sim \mathcal{N}(0, \sigma^2_\alpha I)$$
$$\pi_k^d \sim \text{Beta}(\beta, 1 - \beta)$$
$$\phi_k := (a_k, \pi_k, \omega_k)$$
$$s_{n,k}^d \sim \text{Bernoulli}(\pi_k^d)$$
$$x_n \sim \mathcal{N}(\sum_{k=1}^{\infty} s_{n,k}^d \cdot [r_{nk}(a_k) \circ M_{n,k}], \sigma^2_\phi I),$$

where the operator $\circ$ is the Hadamard product on matrices.

Figure 1 shows how the IBP-distributed matrix $Z$ and other transformations variables combine features to form images for the IBP, LG-tIBP, and M-tIBP.

**4. Inference**

We perform inference of both LG-tIBP and M-tIBP using MCMC. At each iteration, we sample the Gaussian-distributed features $A$, the IBP-distributed binary matrix $Z$, the transformations $R$, the hyperparameters $\alpha, \sigma_\phi$, and $\sigma_\omega$, and, for M-tIBP, the binary masks $S$ and ordering $\omega$.

**4.1. Sampling Indicators, Transformations, and Masks**

In all models, the binary indicator matrix $Z$, the matrix of transformations $R$, and (where appropriate) the feature masks $S$ are all closely coupled. Austerweil & Griffiths (2010) sampled each $z_{nk}$ of $Z$ by explicitly marginalizing over $r_{nk}$, and then sampling $r_{nk}$. However, explicitly computing the conditional distribution for all transformations for each feature cannot scale to even moderate-sized images (as discussed in Section 5). Instead, we sample $z_{nk}, r_{nk}$ and $s_{n,k}$ jointly via a Metropolis-Hastings step.

The efficacy of a Metropolis-Hastings sampler depends on the quality of the proposal distribution. We design a data-driven proposal distribution (Tu & Zhu, 2002) based on an established pattern matching technique that assigns high probability to plausible states.

**Feature Indicator Proposal Distribution** Let $K_+$ be the highest feature index represented in the data, excluding the
current data point. Our proposal distribution for $z_{nk}, k \leq K_k$ is

$$q(z_{nk} \rightarrow z'_{nk}) = \begin{cases} 1 & \text{if } z'_{nk} \neq z_{nk} \\ 0 & \text{otherwise.} \end{cases}$$

(4)

Our proposal distribution for previously unseen features follows Griffiths & Ghahramani (2005): sample $K^*$ new features according to Poisson($\alpha/N$), where $N$ is the number of observations.

Transformation Proposal Distribution To obtain a proposal distribution for translations $r_{nk}$ that matches our intuitions about the true posterior, we look at the cross-correlation between the feature $a_k$ and the residual $\tilde{x}_{n,k}$ obtained by removing all but that feature from the image $x_n$.

Cross-correlation (Duda & Hart, 1973) is a standard tool in classical image analysis and pattern-matching. The cross-correlation $u \ast v$ between two real-valued images $u$ and $v$ is a measure of the similarity between $u$ and a translated version of $v$, i.e., $(u \ast v)(t) := \sum_{\tau=-T}^{T} u(\tau)v(t+\tau)$.

Since our proposal distribution for $r_{nk}$ must be strictly positive, we use the exponentiated function

$$q(r|a_k, \tilde{x}_{n,k}) \propto \exp \{(\tilde{x}_{n,k} \ast a_k)(r)\},$$

(5)

for our proposal distribution, and define the residual

$$\tilde{x}_{n,k} = \sum_{j:w_{nj} < \omega_k} M_{nj} \circ x_n$$

(6)

for M-tIBP, and

$$\tilde{x}_{n,k} = x_n - \sum_{j \neq k} z_{nj} r_{nj}(a_k)$$

(7)

for LG-tIBP.

In Figure 2, we show the proposal distribution for $r_{nk}$ for a feature and three data points. The proposal distribution peaks in the locations that best match the pattern of pixels in the feature. If no locations match the feature, the proposal distribution is relatively entropic. Thus, the cross-correlation proposal distribution will cause us to consider good candidates for $r_{nk}$.

To incorporate scaling and rotation in addition to translation, we must increase the space over which we define our Metropolis-Hastings proposal. For a small transformation space (e.g., multiples of $\pi$ rotation and half / double scaling) it remains practical to extend the proposal distribution to include all possible scaling and rotation combinations. We separately obtain cross-correlations of these transformed features with the residual image, and concatenate the resulting vectors to obtain a distribution over all possible transformations. For new features, $r_{nk}$ is set to be the identity transformation.

For newly seen features, the binary mask $s_{nk}$ is sampled from a uniform distribution between two real-valued images $u$ and $v$.

$$q(s_{nk}) = \prod_{d=1}^{D} p(s_{nk}^d = v|s_{-(n,k)})$$

(8)

$$p(s_{nk}^d = 1|s_{-(n,k)}) = \frac{\sum_{m \neq n} s_{mk}^d + \beta}{\sum_{m \neq n} s_{mk} + 2\beta}.$$  

(9)

Unseen Features For previously unseen features, we sample a new feature $a_k \sim \mathcal{N}(0, \sigma^2)$. Our proposal distribution for the corresponding mask is obtained by normalizing $a_k$ and sampling each pixel of the proposed mask $s_{nk}^d$ according to a series of Bernoulli distributions parameterized by the normalized entries of $a_k$.

4.2. Resampling Transformation and Masks

In addition to sampling $z_{nk}, r_{nk}$ and $s_{nk}$ jointly, we also resample $r_{nk}$ (and, for M-tIBP, $s_{nk}$) for values of $n$ and $k$ for which $z_{nk} = 1$. We jointly resample $r_{nk}$ using a Metropolis-Hastings step with proposal distribution $q_r(r_{nk})$ (or $q_r(r_{nk})q_s(s_{nk}))$. For the M-tIBP, we also Gibbs sample the binary masks using the conditional distribution

$$p(s_{nk}^d|s_{-(n,k)}, x_n, z, r, A) \propto p(x_n|s_{nk}^d, z, r_n, A) \cdot p(s_{nk}^d | s_{-(n,k)}),$$

(10)

where $p(s_{nk}^d | s_{-(n,k)})$ is given in Eqn. (9).

4.3. Sampling the Feature Order

We assume the feature order $\omega$ is sampled from a uniform distribution over permutations. We sample the feature order using a Metropolis-Hastings step where we uniformly choose two consecutive features and propose an order swap.

4.4. Sampling Features and Hyperparameters

Conjugacy eases the sampling of $a_k$. For the M-tIBP, we sample the $d^{th}$ pixel of the $k^{th}$ feature as

$$a_{kd}|Z, R, S, X \sim \mathcal{N}(\frac{\beta}{\sum_{n=1}^{N} M_{nk}x_n r_{nk}(d)}, F),$$

(11)
where $F = (\sigma^{-2} + \sigma^{-2} \sum_{n=1}^{N} M_{n,k}^2)^{-1}$.

The hyperparameters $\alpha$, $\sigma_{\alpha}$ and $\sigma$ can be Gibbs sampled via closed form equations (Doshi-Velez, 2009).

### 4.5. Modeling Color Images

The derivation above assumes that each pixel is a single real number. However, natural images are typically have color information, represented as a three-dimensional vector for each pixel. In our model, all colors contribute to the image likelihoods. Similarly, the proposal distribution is an element-wise sum over all possible channels,

$$q(r|a_{k}, \tilde{x}_{n,k}) \propto \exp \left\{ \sum_{c} (\tilde{x}_{n,k}^c \ast a_{k}^c)(r) \right\},$$

where $\tilde{x}_{n,k}$ and $a_{k}$ are $c$-channel contribution of $\tilde{x}_{n,k}$ and $a_{k}$, respectively.

In the M-tIBP case, for feature $k$ in image $n$, we assume all channels share a common mask $s_{n,k}$.

### 5. Computational Complexity

The main motivation behind the algorithm proposed in Section 4 is to allow the transformed IBP to be applied to large data. Austerweil & Griffiths (2010) calculate the likelihood of the data for every possible transformation. Replacing this naive approach with the sampler presented above can achieve a speed-up of at least $O(D \min(SR, K/\log D))$, where $R$ is the number of rotations considered, $S$ is the number of scales considered, $D$ is the number of pixels, and $K$ is the number of non-zero elements in $z_n$.

Evaluating the LG-tIBP and M-tIBP likelihoods for a single image requires $O(DK)$ computations. Since the number of possible translations is $O(D)$, calculating the likelihood for all possible translations in $O(SRD^2K)$, yielding a total per-iteration complexity of $O(NDK^2)$ for the inference method used by Austerweil & Griffiths (2010). If we were to also sum over values of $s_{n,k}$, this would scale as $O(2^D)$.

By contrast, calculating the cross-correlation between a feature and an image residual can be done using the fast Fourier transform in $O(D \log D)$, so the proposal distribution described in Section 4.1 can be calculated in $O(SRD \log D)$. The likelihood need only be evaluated twice in the Metropolis-Hastings step, so our sampler scales as $O(NSRDK \max(K, \log D))$.

### 6. Experimental Evaluation

We evaluate the LG-tIBP and M-tIBP models on both simulated and real-world data against the linear Gaussian IBP

---

3 Since features can be centered outside the image, the total number of translations is in fact greater than the number of pixels.

Figure 4. A run-time and likelihood comparison of using a Metropolis Hastings sampling of R and Z vs. Gibbs Sampling of R and Z. MH9 and Gibbs9 used a synthetic dataset with 9 by 9 pixel images, while MH15 and Gibbs15 used 15 by 15 pixels.

(a) True features  (b) True data
(c) LG-tIBP features  (d) LG-tIBP reconstruction
(e) M-tIBP features  (f) M-tIBP reconstruction

Figure 5. Evaluation of LG-tIBP and M-tIBP on synthetic data (image size 15 × 15) with translation, rotation and scaling.

ized in practice; while convergence requires slightly more iterations, it requires far less total CPU time.

In addition, we trained LG-tIBP and M-tIBP on a dataset where features have been scaled, rotated, and translated. This was not implemented by Austerweil & Griffiths (2010), presumably due to the computational cost. Figure 5 shows that our two models successfully detected the underlying features. The ordering learned by M-tIBP matches the true order, except in the case of the green “▽” and the blue “×”, which did not often overlap.

Real-world data To show that the performance on simulated data in Section 6 carries over to real images, we evaluated LG-tIBP and M-tIBP on four image datasets, chosen to reflect various levels of complexity from simple video games with static/dynamic background to real-world scenes.

1. DNK: 171 screen shots from the 1981 video game “Donkey Kong”.
2. SMB: 200 screen shots from the 1985 video game “Super Mario Brothers”.
3. TFC: 186 frames from an intersection traffic video.
4. WLK: 226 frames from a video of people walking in a Lisbon shopping center.\textsuperscript{3}

All images were resized to 101 × 101 pixels. We trained and tested the models using the full three-channel RGB data.

For each dataset, we trained LG-tIBP, M-tIBP, IBP and SPRITE\textsuperscript{5} on a randomly selected 80% of the images with the remaining 20% held out for testing. Since the NO-tIBP is only appropriate for binary data, we could not compare with this method. We used the features extracted from the training set to estimate Z and R on test data, and evaluated the reconstructions using test set RMSE. Table 1 shows that LG-tIBP and M-tIBP achieve better performance than IBP across all datasets; M-tIBP performs equally well as SPRITE on three datasets, and much better on the SMB dataset. M-tIBP performs better than LG-tIBP on SMB and WLK datasets, but worse than LG-tIBP on DNK. This is because DNK has limited occlusions and a black background, and so can be adequately represented using the simpler LG-tIBP.

Figure 6 shows reconstructions and features obtained using the IBP, SPRITE, LG-tIBP, and M-tIBP. The IBP only matches the image background. In contrast, both LG-tIBP and M-tIBP identify shapes that appear in different locations. For example, in the first column of Figure 6, LG-tIBP identifies Donkey Kong (cyan) and a fireball (yellow), in addition to the background (green). Interestingly, LG-tIBP mis-identifies a pie\textsuperscript{10} as a fireball but missed the actual fireball. Our M-tIBP model detected the pie (red) and the fireball (blue), while Donkey Kong (cyan) and background (yellow) are also clearly identified. Though M-tIBP has

Table 1. Test set per-pixel per-channel RMSE (lower is better) on four datasets. The number of features for SPRITE is set to be the true number of features of each dataset. LG-tIBP and M-tIBP outperforms IBP on all datasets. M-tIBP works better than LG-tIBP on SMB and WLK, equally well with LG-tIBP on TFC, worse on DNK. M-tIBP performs equally well with SPRITE on three datasets, and outperforms SPRITE on SMB.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IBP</th>
<th>SPRITE</th>
<th>LG-tIBP</th>
<th>M-tIBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNK</td>
<td>0.098</td>
<td>0.093</td>
<td>0.064</td>
<td>0.079</td>
</tr>
<tr>
<td>SMB</td>
<td>0.144</td>
<td>0.202</td>
<td>0.078</td>
<td>0.045</td>
</tr>
<tr>
<td>TFC</td>
<td>0.131</td>
<td>0.070</td>
<td>0.083</td>
<td>0.084</td>
</tr>
<tr>
<td>WLK</td>
<td>0.154</td>
<td>0.059</td>
<td>0.081</td>
<td>0.067</td>
</tr>
</tbody>
</table>

\textsuperscript{3}Raw WalkByShop1cor data available at \url{http://groups.inf.ed.ac.ukVISION/CAVIAR/CAVIARDATA1/}

\textsuperscript{5}The publicly available implementation of SPRITE could not detect any features in our datasets. To enable the fairest comparison possible, we compare against a finite version of M-tIBP with a fixed K (based on the “true” K based on inspecting the dataset, as in previous works using SPRITE) and an “always on” Z. We believe that this is equivalent to the SPRITE model, although the inference implementation has tweaks and tricks that restrict the kinds of features that can be learned.

\textsuperscript{10}“Pie” is the common name used for these sprites by Donkey Kong players; the designers’ intent was to depict troughs of cement
Modeling Images using Transformed Indian Buffet Processes

Donkey Kong (DNK) Super Mario (SMB) Traffic (TFC) Walker (WLK)

Original Data

Reconstructed by IBP

Reconstructed by LG-tIBP

Reconstructed by Sprite

Reconstructed by M-tIBP

Figure 6. Reconstructions of test data by the IBP and LG-tIBP. First row: True image. Second, third, fourth and sixth rows: reconstructed image by IBP, Sprite, LG-tIBP, and M-tIBP, respectively. Fifth and seventh rows: features detected by LG-tIBP and M-tIBP, respectively, superimposed on the true image. Each color is a feature; colors are consistent between columns. Each pair of adjacent columns are two images from the DNK, SMB, TFC and WLK datasets, respectively.

slightly larger RMSE than LG-tIBP on this dataset, the features seems more intuitive.

In the Super Mario dataset, while LG-tIBP extracted the bush and brick clearly, M-tIBP managed to extract the text “100”, denoting points earned by the player (green). Sprite performs poorly, possibly due to the large, sparsely observed feature set. M-tIBP identified the blue sky as two parts: one is the red feature and the other is the green feature. Because bricks often appear in the center of the screen, the model learns to “occlude” that location with a patch of sky.

While LG-tIBP and M-tIBP can learn features and transformations, M-tIBP is, on the whole, more accurate and the reconstructions are clearer. Sprite can generally reconstruct data as well as M-tIBP, but the extracted features are less clear. One possible reason is that Sprite assumes all features are present in each image. Moreover, in practice it is difficult to know a priori the number of features in a dataset. These two factors mean Sprite is unlikely to scale to heterogeneous datasets such as SMB.

7. Discussion and Future Work

We have presented two nonparametric latent feature models for real-valued images, and presented a novel and efficient inference scheme. In this section, we discuss further applications of this inference paradigm, and discuss possible extensions to our models.

Exploitation of Pattern Matching Algorithms

This inference scheme uses scoring functions from classical image analysis as the proposal distribution in a Metropolis-Hastings algorithm and combines the robustness and computational appeal of a well-established pattern recognition tool.
8. Acknowledgments

The authors would like to thank Joseph Austerweil, Frank Wood, Finale Doshi-Velez and Michalis K. Titsias for publishing implementations. This research was supported by NSF grant #1018625. Jordan Boyd-Graber is also supported by the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072. Any opinions, findings, conclusions, or recommendations expressed are the authors’ and do not necessarily reflect those of the sponsors.

References


