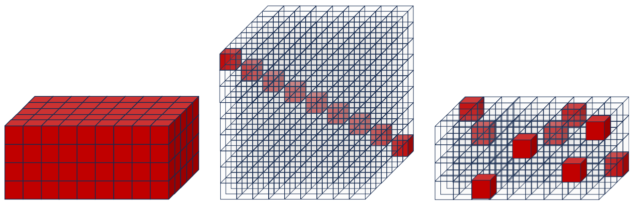
The ALℓ_0CORE Tensor Decomposition for Sparse Count Data

John Hood
University of Chicago

Aaron Schein
University of Chicago

Abstract

This paper introduces ALℓ_0CORE, a new form of probabilistic tensor decomposition. ALℓ_0CORE is a Tucker decomposition that constrains the number of non-zero elements (i.e., the ℓ_0-norm) of the core tensor to be at most Q. While the user dictates the total budget Q, the locations and values of the non-zero elements are latent variables allocated across the core tensor during inference. ALℓ_0CORE—i.e., allocated ℓ_0-constrained core—thus enjoys both the computational tractability of canonical polyadic (CP) decomposition and the qualitatively appealing latent structure of Tucker. In a suite of real-data experiments, we demonstrate that ALℓ_0CORE typically requires only tiny fractions (e.g., 1%) of the core to achieve the same results as Tucker at a correspondingly small fraction of the cost.



(a) Tucker (b) CP (c) ALℓ_0CORE

Figure 1: The core tensor $\mathbf{\Lambda}$ in three related tensor decompositions. Transparent versus red values denote zero versus non-zeros. ALℓ_0CORE relies on sparsity to achieve the representational richness of Tucker without suffering its "exponential blowup" in parameters.

1 INTRODUCTION

Tensor decomposition methods (Kolda and Bader, 2009; Cichocki et al., 2015) aim to find parsimonious representations of multi-mode data by seeking constrained (e.g., low-rank or sparse) reconstructions of observed tensors. When well-tailored, such methods are effective at distinguishing salient structure in data from idiosyncratic noise. Tensor decomposition methods have proven useful for a variety of predictive tasks like de-noising (Marot et al., 2008), imputation (Tomasi and Bro, 2005), and forecasting (Xiong et al., 2010), as well as for counterfactual prediction in the context of causal modeling (Amjad et al., 2019). Due to their parsimony, tensor decomposition methods also tend to be highly interpretable, and are routinely used for exploratory and descriptive data analysis.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

The two predominant forms of tensor decomposition, canonical polyadic (CP) and Tucker decomposition, each have benefits and drawbacks. CP decomposes an observed tensor $\mathbf{Y}$ of shape $D_1 \times \cdots \times D_M$ into M factor matrices $\Phi^{(1)}, \ldots, \Phi^{(M)}$, the mth having shape $D_m \times K$. CP can be understood as learning K latent *classes*, the kth of which represents a particular multi-mode pattern in the data via the factor vectors $\boldsymbol{\phi}_k^{(1)}, \ldots \boldsymbol{\phi}_k^{(M)}$. This decomposition is simple and computationally advantageous, scaling generically with K and the size of the observed tensor, $|\mathbf{Y}| = \prod_{m=1}^{M} D_m$, as $\mathcal{O}(|\mathbf{Y}| \cdot K)$.

The Tucker decomposition, on the other hand, seeks a richer representation. It allows for different numbers of factors in each mode, such that the mth factor matrix $\Phi^{(m)}$ takes shape $D_m \times K_m$. Each latent class then corresponds to one of the $\prod_{m=1}^{M} K_m$ unique combinations of per-mode factors, and is weighted by its corresponding entry in the $K_1 \times \cdots \times K_M$ *core tensor* $\mathbf{\Lambda}$. This richer representation, while qualitatively appealing, enacts a price: what is known as Tucker's "exponential blowup" is the fact that inference scales with the size of the core, $|\mathbf{\Lambda}| = \prod_{m=1}^{M} K_m$, as $\mathcal{O}(|\mathbf{Y}| \cdot |\mathbf{\Lambda}|)$.

This paper seeks a middle ground between CP and Tucker. We introduce a new family of probabilistic tensor decomposition models that enjoy the representational richness of Tucker without its attendant computational cost. Our approach is inspired by ideas in *count* tensor decomposition, and tailored accordingly.

Many tensors in practice are count tensors, or contingency tables, which arise naturally as summaries of tabular data. Such tensors tend to be very large but very sparse—i.e., $|\mathbf{Y}| \gg ||\mathbf{Y}||_0$—where the ℓ_0-norm counts the number of non-zeros. Many tensor decomposition methods are specifically tailored to this setting and adopt loss functions or likelihoods consistent with the assumption that the observed entries are Poisson-distributed. Under a Poisson assumption, CP generally scales only as $\mathcal{O}(||\mathbf{Y}||_0 \cdot K)$, and Tucker as $\mathcal{O}(||\mathbf{Y}||_0 \cdot |\mathbf{\Lambda}|)$, both of which represent a substantial computational improvement over their generic counterparts, particularly when $\mathbf{Y}$ is very sparse.

Several recent papers connect Tucker decomposition under a Poisson assumption to models of complex networks (Schein et al., 2016; De Bacco et al., 2017; Aguiar et al., 2023). Tucker has particular qualitative appeal in this setting, allowing for the representation of overlapping communities and cross-community interactions within different network layers and over different temporal regimes. Complex networks are often encoded as very large, sparse count tensors, and thus Poisson Tucker's dependence on only $||\mathbf{Y}||_0$ is critical.

Despite this, Tucker's qualitative appeal still remains incompatible with its computational cost. Modelers often seek to represent large numbers of latent communities, particularly in cases where observed networks contain large numbers of actors. However, specifying large numbers of communities drives the "exponential blowup" in Tucker's core tensor and is thus often infeasible. As a result, modelers typically only use Tucker decomposition with small core tensors. Tucker's conceptual appeal for the analysis of complex networks, among other applications involving large sparse tensors, is therefore not easily attainable, in practice.

This paper breaks ground on the analysis of large sparse tensors with the ALℓ_0CORE (pronounced "al-oh-core") decomposition. ALℓ_0CORE is a Tucker decomposition where the core tensor is constrained to have at most Q non-zero entries—i.e., $||\mathbf{\Lambda}||_0 \leq Q$. The budget Q can be fixed in advance, tuned as a hyperparameter, or inferred as a latent variable. The locations and values of the non-zero entries are treated as latent variables and allocated across the core during inference—the name "$\underline{\text{allo}}$cated $\underline{\ell_0}$-$\underline{\text{co}}$nstrained $\underline{\text{core}}$" is intended to emphasize this central idea. With this constraint, ALℓ_0CORE can explore the same exponentially large latent space as Tucker, but scale generically with only $||\mathbf{\Lambda}||_0$; see fig. 1. When tailored to count tensors, where it is particularly motivated, ALℓ_0CORE scales with $\mathcal{O}(||\mathbf{Y}||_0 \cdot ||\mathbf{\Lambda}||_0)$, and thus exploits the sparsity in both high-dimensional observed and latent spaces.

One might wonder why such a simple idea was not introduced sooner. Indeed, the literature on matrix and tensor (including Tucker) decompositions with both soft (e.g., ℓ_1) and hard (e.g., ℓ_0) sparsity constraints is vast. As we show, the computational advantage of the ALℓ_0CORE decomposition's sparsity constraint is unlocked when combined with a sampling-based approach to inference, which iteratively re-samples the coordinates of the non-zero locations from their complete conditionals. This approach both ensures that instantiations of the core tensor are sparse and permits model-fitting in sub-exponential time. Previous work that has sought to improve the computational efficiency of Tucker has typically looked to optimization, EM, or variational inference (VI)-based approaches, which are commonly believed to be faster than MCMC. Such approaches however, if naïvely adopted to infer the non-zeros in the core, would suffer the same "exponential blowup" as full Tucker. The computational advantages of the ALℓ_0CORE decomposition are naturally tied to MCMC, which we speculate is the reason it has thus far been overlooked.

An outline of the paper is as follows. We introduce notation and provide background on tensor decomposition in section 2. We then sketch a general recipe for ALℓ_0CORE in section 3, which can be combined with a variety of different probabilistic assumptions. In section 4, we give a complete example of such a model, by building Bayesian Poisson ALℓ_0CORE, which is tailored to sparse count tensors representing dynamic multilayer networks. We further provide a full Gibbs sampling algorithm for this model. In section 5, we review and draw connections to related work, highlighting other tensor decomposition methods that are either tailored to sparse data, like complex networks, or incorporate sparsity constraints. In section 6, we then report a suite of experiments on real data of international relations, where we show that ALℓ_0CORE can achieve substantially better predictive performance than full Tucker at only a tiny fraction of the cost, while yielding rich and interpretable latent structure.

2 PRELIMINARIES

Basic notation and terminology. we use lowercase symbols for scalars $\mathrm{y} \in \mathbb{R}$, lowercase bold symbols for vectors $\mathbf{y} \in \mathbb{R}^D$, uppercase symbols for matrices $\mathrm{Y} \in \mathbb{R}^{D_1 \times D_2}$, and uppercase bold symbols for tensors $\mathbf{Y} \in \mathbb{R}^{D_1 \times \cdots \times D_M}$. A tensor $\mathbf{Y}$ of shape $D_1 \times \cdots \times D_M$ has M *modes*, with the m^{th} mode having dimensionality D_m. The tensor contains scalar entries $\mathrm{y}_{\mathbf{d}}$ where $\mathbf{d} \equiv (\mathrm{d}_1, \ldots, \mathrm{d}_M)$ is a *multi-index* containing M coordinates that collectively identify a cell or location in the tensor. The m^{th} coordinate can take D_m values, denoted as $d_m \in [D_m] \equiv \{1, \ldots, D_m\}$.

Tensor decomposition. Tensor decompositions seek a *reconstruction* $\widehat{\mathbf{Y}}$—informally, $\mathbf{Y} \approx \widehat{\mathbf{Y}}$—that is a deterministic function of model parameters $\widehat{\mathbf{Y}} \equiv \widehat{\mathbf{Y}}(\Theta)$. Non-probabilistic methods seek to minimize a loss function $\widehat{\Theta} \leftarrow \arg\min_{\Theta} \ell(\mathbf{Y}, \widehat{\mathbf{Y}}(\Theta))$, while probabilistic methods define a likelihood $P(\mathbf{Y} \mid \widehat{\mathbf{Y}})$ and perform MLE, or introduce priors and perform posterior inference. Such likelihoods typically factorize—i.e.,

$$P(\mathbf{Y} \mid \widehat{\mathbf{Y}}) = \prod_{\mathbf{d}} P(\mathrm{y}_{\mathbf{d}} \mid \widehat{\mathrm{y}}_{\mathbf{d}}). \tag{1}$$

Poisson tensor decomposition. For sparse count tensors, it is common to assume a Poisson likelihood—i.e., $P(\mathrm{y}_{\mathbf{d}} \mid \widehat{\mathrm{y}}_{\mathbf{d}}) = \mathrm{Pois}\left(\mathrm{y}_{\mathbf{d}};\, \widehat{\mathrm{y}}_{\mathbf{d}}\right)$. Among other demonstrated benefits, this assumption leads to inference algorithms that scale only with the number of non-zeros $||\mathbf{Y}||_0$, which can be seen by writing the terms of the log-likelihood proportional to model parameters,

$$\log P(\mathbf{Y} \mid \widehat{\mathbf{Y}}) = \log \prod_{\mathbf{d}} \mathrm{Pois}\left(\mathrm{y}_{\mathbf{d}};\, \widehat{\mathrm{y}}_{\mathbf{d}}\right)$$
$$\propto_{\widehat{\mathbf{Y}}} \sum_{\mathbf{d}} \mathrm{y}_{\mathbf{d}} \log \widehat{\mathrm{y}}_{\mathbf{d}} - \sum_{\mathbf{d}} \widehat{\mathrm{y}}_{\mathbf{d}},$$

and observing that summands in the first sum need only be calculated at the non-zeros $\mathrm{y}_{\mathbf{d}} > 0$, while the second sum does not depend on the data and can typically be computed efficiently for standard parameterizations of $\widehat{\mathrm{y}}_{\mathbf{d}}$. Poisson tensor decomposition can be understood alternatively as minimizing the generalized KL loss or $\mathcal{I}$-divergence (Chi and Kolda, 2012). Its computational advantage applies both to explicitly probabilistic approaches, even those reliant on full Bayesian inference, as well as to non-probabilistic ones. We note that all parameters must be *non-negative*, to satisfy the definition of the Poisson, which promotes a "parts-based" representation (Lee and Seung, 1999) that is highly interpretable.

Tucker decomposition (Tucker, 1966). Under the Tucker decomposition, the reconstruction $\widehat{\mathbf{Y}}$ is the following multilinear function of model parameters:

$$\widehat{\mathbf{Y}} = \sum_{\mathrm{k}_1=1}^{K_1} \cdots \sum_{\mathrm{k}_M=1}^{K_M} \lambda_{\mathrm{k}_1,\dots,\mathrm{k}_M} \left(\boldsymbol{\phi}^{(1)}_{\mathrm{k}_1} \circ \cdots \circ \boldsymbol{\phi}^{(M)}_{\mathrm{k}_M}\right) \tag{2}$$

Unpacking this equation, each term $\boldsymbol{\phi}^{(m)}_{\mathrm{k}_m} \in \mathbb{R}^{D_m}$ represents the $(\mathrm{k}_m)^{\text{th}}$ *latent factor* in the m^{th} mode, which is stored as a column vector in the factor matrix $\Phi^{(m)} \in \mathbb{R}^{D_m \times K_m}$. The notation $(\circ)$ denotes the outer product, and thus each term $(\boldsymbol{\phi}^{(1)}_{\mathrm{k}_1} \circ \cdots \circ \boldsymbol{\phi}^{(M)}_{\mathrm{k}_M})$ is a tensor of the same shape as $\mathbf{Y}$—i.e., $D_1 \times \cdots \times D_M$. Tucker can thus be viewed as a weighted sum of tensors, each weighted by $\lambda_{\mathrm{k}_1,\dots,\mathrm{k}_M}$, an entry in the $K_1 \times \cdots \times K_M$ *core tensor* $\boldsymbol{\Lambda}$. We will often adopt multi-index notation to denote locations in the core, such that an entry is $\lambda_{\mathbf{k}}$, where $\mathbf{k} \equiv (\mathrm{k}_1, \dots, \mathrm{k}_M)$.

CP decomposition (Harshman, 1970). More widely used than Tucker is the canonical polyadic (CP) decomposition (also known as CANDECOMP or PARAFAC). It is a special case of Tucker, where the latent dimensions are all equal to a single value $K = K_1 = \cdots = K_M$ and the core tensor has only K non-zero values $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_K)$ which are arranged along its super-diagonal $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$—i.e.,

$$\lambda_{\mathrm{k}_1,\dots,\mathrm{k}_M} = \begin{cases} \lambda_k & \text{if } k = \mathrm{k}_1 = \cdots = \mathrm{k}_M \\ 0 & \text{otherwise} \end{cases}$$

In such a setting, eq. (2) can be simplified to the form usually given to describe CP—i.e.,

$$\widehat{\mathbf{Y}} = \sum_{k=1}^{K} \lambda_k \left(\boldsymbol{\phi}^{(1)}_k \circ \cdots \circ \boldsymbol{\phi}^{(M)}_k\right) \tag{3}$$

Tucker versus CP. Both decompositions in eqs. (2) and (3) can be viewed as weighted sums over a set of *latent classes.* While CP infers K classes, each defined by a unique set of M factors $\boldsymbol{\phi}^{(1)}_k, \dots, \boldsymbol{\phi}^{(M)}_k$, Tucker infers $\prod_{m=1}^{M} K_m$ classes, each defined by a *unique combination* of factors $\boldsymbol{\phi}^{(1)}_{\mathrm{k}_1}, \dots, \boldsymbol{\phi}^{(M)}_{\mathrm{k}_M}$. Tucker thus involves a much greater degree of parameter sharing, which can make it more statistically efficient, and does not require all modes have the same latent dimensionality, which can be qualitatively appealing. However, this all comes at a computational cost. Both decompositions scale with the number of latent classes, which for Tucker is exponential in the number of modes.

3 ALℓ_0CORE DECOMPOSITION

This section presents a general recipe for ALℓ_0CORE, which combines a sparsity constraint with a sampling-based approach to inference to simultaneously achieve the qualitative appeal and statistical efficiency of Tucker without suffering its computational cost.

ALℓ_0CORE is a Tucker decomposition (eq. (2)) whose core tensor is constrained to have at most Q non-zeros,

$$||\boldsymbol{\Lambda}||_0 \leq Q \tag{4}$$

where $Q \in \mathbb{N}$ is a positive integer-valued *budget* which can be fixed in advance, tuned as a hyperparameter, or inferred as a latent variable.

Unlike in CP decomposition, which also has a sparse core tensor, the non-zero entries in ALℓ_0CORE need not necessarily lie along the core tensor's super-diagonal, and the core tensor need not necessarily be a hypercube with $K_1 = \cdots = K_M$.

Instead, with each $q \in [Q]$ we associate a *non-zero value* $\lambda_q > 0$ and a *location* in the core tensor, denoted by the multi-index $\mathbf{k}_q \equiv (\mathrm{k}_{q,1}, \dots, \mathrm{k}_{q,M})$ where $\mathrm{k}_{q,m} \in [K_m]$. Each q allocates its value λ_q to location $\mathbf{k}_q$ such that the value of the core tensor at an arbitrary location $\mathbf{k} \equiv (\mathrm{k}_1, \dots, \mathrm{k}_M)$ is

$$\lambda_{\mathbf{k}} = \sum_{q=1}^{Q} \mathbb{1}(\mathbf{k}_q = \mathbf{k})\, \lambda_q \tag{5}$$

The name ALℓ_0CORE is a contraction of "<u>allo</u>cated <u>ℓ_0</u>-<u>co</u>nstrained <u>core</u>" that is meant to convey this idea.

The ALℓ_0CORE decomposition can thus be written as

$$\widehat{\mathbf{Y}} = \sum_{q=1}^{Q} \lambda_q \left(\boldsymbol{\phi}^{(1)}_{\mathrm{k}_{q,1}} \circ \dots \circ \boldsymbol{\phi}^{(M)}_{\mathrm{k}_{q,M}} \right) \tag{6}$$

Like Tucker and unlike CP, this decomposition posits a different latent dimensionality K_m for every mode, and can represent any of the $\prod_{m=1}^{M} K_m$ possible latent classes. However, unlike Tucker and like CP, the number of represented classes is capped at a small number $Q \ll \prod_{m=1}^{M} K_m$ and thus the parameter space does not suffer Tucker's "exponential blowup" as we increase the number of modes M or the latent dimensionalities K_m.

While eq. (6) can clearly be computed without "exponential blowup" in cost, inference of the (unknown) non-zero locations $\mathbf{k}_q$ may still be exponential, since each multi-index can take any of $\prod_{m=1}^{M} K_m$ values. However, in what follows, we introduce additional probabilistic structure that makes the model amenable to a sampling-based inference scheme which both preserves the sparsity in instantiations of $\boldsymbol{\Lambda}$ and explores new values of $\mathbf{k}_q$ in $\mathcal{O}(Q \sum_{m=1}^{M} K_m)$ time.

Latent allocation. The key motif that facilitates efficient parameter inference in ALℓ_0CORE is the explicit treatment of the latent non-zero locations $\mathbf{k}_q$ as conditionally independent categorical random variables. A very general form for their prior is given by

$$\mathbf{k}_q \overset{\text{iid}}{\sim} \text{Categorical}(\boldsymbol{\Pi}) \qquad \text{for } q \in [Q] \tag{7}$$

where $\boldsymbol{\Pi}$ is some prior probability tensor of the same shape as $\boldsymbol{\Lambda}$ that sums to 1 over all cells $\sum_{\mathbf{k}} \pi_{\mathbf{k}} = 1$. Equation (7) simply states that the prior probability of the q^{th} unknown location being $\mathbf{k}_q$ is $P(\mathbf{k}_q \mid \boldsymbol{\Pi}) = \pi_{\mathbf{k}_q}$.

Although $\boldsymbol{\Pi}$ may seem to suffer the same "exponential blowup" that ALℓ_0CORE seeks to alleviate, we can restrict it to have low-rank structure—e.g.,

$$\boldsymbol{\Pi} = \boldsymbol{\pi}^{(1)} \circ \dots \circ \boldsymbol{\pi}^{(M)} \tag{8}$$

where $\boldsymbol{\pi}^{(m)} \in \Delta^{K_m - 1}$ is a simplex vector. This corresponds to a rank-1 CP decomposition of $\boldsymbol{\Pi}$, such that $P(\mathbf{k}_q \mid \boldsymbol{\Pi}) = \prod_{m=1}^{M} \pi^{(m)}_{\mathrm{k}_{q,m}}$, and thus avoids any exponential dependence on the number of modes. We note that more structured ways of modeling $\boldsymbol{\Pi}$ are compatible with ALℓ_0CORE but leave their development for the future and adopt this one in all that follows.

With this prior, inference is then based on re-sampling each index $\mathrm{k}_{q,m} \in [K_m]$ from its *complete conditional*

$$\text{for } q \in [Q] \text{ and } m \in [M]: \quad (\mathrm{k}_{q,m} \mid -) \sim P(\mathrm{k}_{q,m} \mid \mathbf{k}_{q,\neg m}, \mathbf{Y}, -) \tag{9}$$

which is its distribution conditional on data $\mathbf{Y}$, the other indices in the multi-index, denoted here as $\mathbf{k}_{q,\neg m}$, and all other latent variables and parameters in the model, represented by the $(-)$. We could also simply use $P(\mathrm{k}_{q,m} = k \mid -)$ to refer to this conditional. However, we explicitly include $\mathbf{Y}$ in eq. (9) to highlight that this is a posterior distribution, and explicitly include $\mathbf{k}_{q,\neg m}$ to emphasize that we re-sample each *sub-index* of $\mathbf{k}_q$ while holding all other sub-indices fixed.

For any likelihood, the complete conditional in eq. (9) takes the following general form

$$P(\mathrm{k}_{q,m} = k \mid -) = \frac{\pi^{(m)}_{k}\, P\big(\mathbf{Y} \mid \widehat{\mathbf{Y}}(\mathrm{k}_{q,m} = k)\big)}{\sum_{k'=1}^{K_m} \pi^{(m)}_{k'}\, P\Big(\mathbf{Y} \mid \widehat{\mathbf{Y}}(\mathrm{k}_{q,m} = k')\Big)}$$

where the notation $\widehat{\mathbf{Y}}(\mathrm{k}_{q,m} = k)$ denotes the function in eq. (6) computed after setting $\mathrm{k}_{q,m} = k$.

This probability involves only K_m summands in the denominator. One loop through eq. (9) thus takes $\mathcal{O}(Q \sum_{m=1}^{M} K_m)$ time for a given likelihood and shape of the observed tensor. By comparison, a naïve approach that re-samples each multi-index jointly from $P(\mathbf{k}_q \mid -)$ would involve a sum over all core entries, and thus cost $\mathcal{O}(Q \prod_{m=1}^{M} K_m)$.

Repeatedly re-sampling the sub-indices from their complete conditionals constitutes a Gibbs sampler—i.e., a Markov chain whose stationary distribution is the exact posterior $P(\mathbf{k}_1, \dots \mathbf{k}_Q \mid \mathbf{Y})$. This can be incorporated into a variety of inference schemes for the other latent variables, λ_q and $\Phi^{(m)}$, such as Monte Carlo EM, stochastic variational inference, or full MCMC, the details of which are model-dependent.

We emphasize here that regardless of the approach taken for other variables, the computational benefit of ALℓ_0CORE is tied to a sampling-based approach for $\mathbf{k}_q$. By sampling $\mathbf{k}_q$ instead of computing expectations as one would do in EM or variational inference, we maintain a sparse instantiation of the core tensor which allows eq. (6) to be computed with only Q summands. Moreover, by adopting a sampling approach, ALℓ_0CORE explores the state-space of $\mathbf{k}_q$ without "exponential blowup" in time or memory complexity.

4 BAYESIAN POISSON ALℓ_0CORE

We now provide a complete example of an ALℓ_0CORE-based model and corresponding inference algorithm. The model we develop here is tailored to large sparse count tensors and motivated by applications to the analysis of dynamic multilayer networks.

This model assumes each entry of a count tensor $\mathbf{Y} \in \mathbb{N}_0^{D_1 \times \cdots \times D_M}$ is conditionally Poisson distributed,

$$\mathrm{y}_{\mathbf{d}} \overset{\text{ind.}}{\sim} \text{Pois}\big(\widehat{\mathrm{y}}_{\mathbf{d}}\big), \quad \widehat{\mathrm{y}}_{\mathbf{d}} \equiv \sum_{q=1}^{Q} \lambda_q \prod_{m=1}^{M} \phi^{(m)}_{\mathrm{d}_m, \mathrm{k}_{q,m}} \tag{10}$$

where the form of $\widehat{\mathrm{y}}_{\mathbf{d}}$ unpacks eq. (6) for a specific $\mathbf{d}$.

Following eqs. (7) and (8), we assume a rank-1 decomposition of the prior probability tensor $\mathbf{\Pi}$, so that the M coordinates of the Q non-zero locations in the core are conditionally independent:

$$\mathrm{k}_{q,m} \overset{\text{ind.}}{\sim} \text{Categorical}(\boldsymbol{\pi}^{(m)}) \tag{11}$$

We then posit conjugate Dirichlet priors,

$$\boldsymbol{\pi}^{(m)} \overset{\text{ind.}}{\sim} \text{Dirichlet}(\tfrac{\alpha_0^{(m)}}{D_m}, \dots, \tfrac{\alpha_0^{(m)}}{D_m}), \tag{12}$$

where $\alpha_0^{(m)}$ is a hyperparameter. For both the non-zero core values and the entries of the factor matrices, we posit independent gamma priors, which are conditionally conjugate to the Poisson likelihood:

$$\lambda_q \overset{\text{iid}}{\sim} \text{Gam}(a_0,\, b_0), \tag{13}$$

$$\phi^{(m)}_{d,k} \overset{\text{iid}}{\sim} \text{Gam}(e_0,\, f_0) \quad d \in [D_m], k \in [K_m] \tag{14}$$

Full MCMC inference. With a standard auxiliary variable scheme, this model is amenable to full Bayesian inference via Gibbs sampling for all parameters. It is also possible to adopt a hybrid approach that interleaves Gibbs sampling for the non-zero core locations (eq. (9)) with point estimation or variational inference of the model parameters; we leave exploration of such schemes for future work.

As with all "allocative" Poisson models (Schein, 2019; Yıldırım et al., 2020), the likelihood in eq. (10) can be equivalently expressed so that the observed count $\mathrm{y}_{\mathbf{d}} = \sum_{q=1}^{Q} \mathrm{y}_{\mathbf{d},q}$ is assumed equal to a sum of "latent sources" (Cemgil, 2009), each of which is a conditionally independent Poisson random variable

$$\mathrm{y}_{\mathbf{d},q} \overset{\text{ind.}}{\sim} \text{Pois}\left(\widehat{\mathrm{y}}_{\mathbf{d},q}\right), \quad \widehat{\mathrm{y}}_{\mathbf{d},q} \equiv \lambda_q \prod_{m=1}^{M} \phi^{(m)}_{\mathrm{d}_m, \mathrm{k}_{q,m}} \tag{15}$$

While the gamma priors in eqs. (13) and (14) are not directly conjugate to the likelihood of $\mathrm{y}_{\mathbf{d}}$ due to the sum in its rate, they are conjugate to eq. (15). An MCMC approach therefore begins by sampling the latent sources as auxiliary variables from their complete conditional, which is multinomial:

$$\left(\left(\mathrm{y}_{\mathbf{d},q}\right)_{q=1}^{Q} \mid -\right) \sim \text{Multi}\left(\mathrm{y}_{\mathbf{d}},\, \left(\tfrac{\widehat{\mathrm{y}}_{\mathbf{d},q}}{\sum_{q'=1}^{Q} \widehat{\mathrm{y}}_{\mathbf{d},q'}}\right)_{q=1}^{Q}\right)$$

This is commonly referred to as the "thinning step" in the Poisson factorization literature and represents the main computational bottleneck. Notice that it need only be run for $\mathrm{y}_{\mathbf{d}} > 0$ and thus scales well with only the observed non-zeros $||\mathbf{Y}||_0$. Bayesian Poisson Tucker decomposition (Schein et al., 2016) features an analogous step—however, there, $\mathrm{y}_{\mathbf{d}}$ is thinned across all $\prod_{m=1}^{M} K_m$ latent classes of the Tucker decomposition, and thus scales as $\mathcal{O}(||\mathbf{Y}||_0 \cdot |\mathbf{\Lambda}|)$, whereas here the thinning step is only $\mathcal{O}(||\mathbf{Y}||_0 \cdot ||\mathbf{\Lambda}||_0)$.

Conditioned on values of all latent sources $\mathrm{y}_{\mathbf{d},q}$, the complete conditionals for all other parameters are available in closed form via conjugacy and can be sampled from efficiently—we relegate the statement and derivation of their specific forms to the appendix.

5 RELATED WORK

The basic idea of promoting sparse representations in tensor or matrix decompositions has been widespread in the literature for decades. The majority of such work introduces *soft* sparsity constraints that encourage sparsity but do not guarantee any specific degree of it. This is most commonly achieved by penalizing the ℓ_1-norm of model parameters, which can be incorporated into loss functions while preserving convexity. Such work goes back to at least the non-negative sparse basis and coding methods of Hoyer (2002, 2004). Penalizing the ℓ_0-norm of model parameters introduces a non-convex constraint, and much less work has embraced hard sparsity constraints as a result. Morup et al. (2008), for instance, motivate ℓ_0 constraints, but then approximate them with convex ℓ_1 constraints.

Much of the existing work on ℓ_0-constrained decompositions are for non-negative *matrix* factorization (Peharz and Pernkopf, 2012; Bolte et al., 2014; Cohen and Gillis, 2019; de Leeuw, 2020; Wu et al., 2022; Nadisic et al., 2022). An exception is that of Kiers and Giordani (2020) who develop an ℓ_0-constrained CP decomposition method, which consists of truncating small values in the factor matrices to zero during inference.

A number of papers present approaches involving the phrase "sparse Tucker". This typically refers to sparsity in the observed tensor, not the core. However, a few recent papers introduce sparsity to the core tensor (Park et al., 2021; Fang et al., 2021; Zhang and Ng, 2022). While these approaches introduce interesting new motifs, such as spike-and-slab priors, they do not exploit that sparsity to improve the computational efficiency of Tucker, with some having quadratic or even cubic dependence on the size of the core tensor.

There is a rich tradition of CP decomposition models for large sparse count tensors or contingency tables under Poisson or multinomial likelihoods (Welling and Weber, 2001; Dunson and Xing, 2009; Chi and Kolda, 2012; Zhou et al., 2015; Schein et al., 2015, inter alia).

There is much less work on Tucker decomposition for count tensors. Bhattacharya and Dunson (2012) introduced Tucker under a multinomial likelihood and showed its relation to latent class models. Johndrow et al. (2017) further developed a "collapsed" Tucker model which allocates the M sub-indices of each latent class to $M' \leq M$ groups, thus interpolating between CP ($M'=1$) and full Tucker ($M'=M$); this work shares many themes with ours, but is distinct.

Finally, multiple recent convergent lines of work advocate for the use of Poisson Tucker decomposition in analyzing complex multilayer networks (Schein et al., 2016; De Bacco et al., 2017; Aguiar et al., 2023; Stoehr et al., 2023), as its structure can be seen to unify a number of community block models in the statistical networks literature. While it can be applied more generally, ALℓ_0CORE draws inspiration from this line of work and is intended to allow Tucker's promise to be practically achievable in the analysis of large networks.

6 COMPLEX NETWORK DATA

In this section we apply the ALℓ_0CORE model in section 4 to two real datasets of dynamic multilayer networks of nation-state actors. We evaluate ALℓ_0CORE's predictive performance as a function of wall-clock time as we vary Q, and compare it to CP and full Tucker decomposition, using both our own implementation of Gibbs sampling, as well as a recent state-of-the-art implementation of EM (Aguiar et al., 2023). We also provide an illustrative exploration of the interpretable latent structure inferred by ALℓ_0CORE.

6.1 International Relations Event Datasets

Data of the form "country i took action a to country j at time step t" are routinely analyzed in political science (Schrodt et al., 1995). Such data can be viewed as a dynamic multilayer network of V country actors with A layers (i.e., action types) that evolves over T time steps, and can be represented as a 4-mode count tensor $\mathbf{Y} \in \mathbb{N}_0^{V \times V \times A \times T}$, where each entry $y_{i,j,a,t}$ records the number of times i took action a to j during time step t. We adopt the notation $y_{i,j,a,t} \equiv y^{(t)}_{i \xrightarrow{a} j}$.

A line of papers advocate for Tucker decompositions to analyze international relations event data (Hoff, 2015, 2016; Schein et al., 2016; Minhas et al., 2016; Stoehr et al., 2023), and we follow in this line by analyzing two of such datasets with ALℓ_0CORE. The 4-mode core tensor $\boldsymbol{\Lambda}$, in this setting, can be interpreted as capturing community–community interactions, where an entry $\lambda_{c,d,k,r}$ is the rate at which the c^{th} *community* of sender countries takes actions in *topic k* towards the d^{th} *community* of receiver countries when in *temporal regime r*. We adopt the notation $\lambda_{c,d,k,r} \equiv \lambda^{(r)}_{c \xrightarrow{k} d}$.

TERRIER. We construct the first tensor using the Temporally Extended, Regular, Reproducible International Event Records (TERRIER) dataset, recently released by Halterman et al. (2017). This dataset codes for $V=206$ country actors and adopts the CAMEO coding scheme for action types (Gerner et al., 2002), wherein $A = 20$ high-level action types range in severity from MAKE PUBLIC STATEMENT ($a = 1$) to ENGAGE IN MASS VIOLENCE ($a = 20$). The observation window ranges from January 2000 to December 2006, which we bin monthly to obtain $T = 84$ time steps. The resulting tensor $\mathbf{Y} \in \mathbb{N}_0^{206\times206\times20\times84}$ is 99.5% sparse.

ICEWS. We construct the second tensor using the Integrated Crisis Early Warning System (ICEWS) dataset (Boschee et al., 2015). ICEWS also uses the CAMEO coding scheme ($A = 20$) but represents slightly more $V = 249$ country actors, and has a longer observation window from 1995 to 2013, which we bin yearly to obtain $T = 19$. The resulting count tensor $\mathbf{Y} \in \mathbb{N}_0^{249\times249\times20\times19}$ is 98.5% sparse.

6.2 Experimental Design and Settings

Design. To evaluate predictive performance, we randomly generate multiple train–test splits of the ICEWS and TERRIER tensors. Each split holds out a set of *fibers* of the tensor—i.e., vectors indexed by coordinates to all modes but one. In this case, we select a set of time fibers—i.e., we randomly select combinations of sender, receiver, and action type (i, j, a) and hold out the vector $\mathbf{y}_{i \xrightarrow{a} j} \in \mathbb{N}_0^T$ for each one. In each split, we hold out a random 1% of all $V \cdot V \cdot A$ such fibers, and generate 8 such splits for each tensor.

ALℓ_0CORE settings. We implemented the Gibbs sampling algorithm for the ALℓ_0CORE model in section 4 using the programming language Julia. In all experiments, we discard the first 1,000 samples as burn-in and then run MCMC for another 4,000 iterations, saving every 20th sample. This returns a set of $S = 200$ posterior samples of the model parameters, which can be used to compute a set of reconstructions $\big(\widehat{\mathbf{Y}}^{(s)}\big)_{s=1}^{S}$.

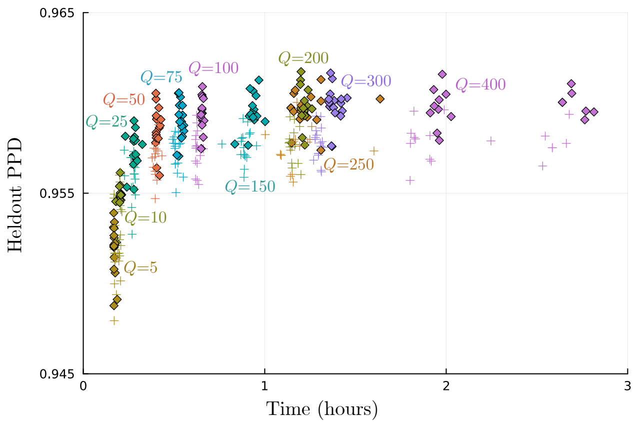


Figure 2: PPD by wall-clock time. Each point is an ALℓ_0CORE model run for 5,000 iterations, with a certain Q, denoted by color, and either the large or small core shape, denoted by $\diamond$ or $+$, respectively. Performance plateaus early, suggesting ALℓ_0CORE can match full Tucker at a small fraction of cost.

Throughout these experiments, we vary Q to see its effect on both predictive performance and runtime. Since the computational cost is determined by Q (rather than the size of the core), we always set $K_1 = \cdots = K_M = Q$, for simplicity, and further initialize the core tensor to be super-diagonal $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_Q)$. We refer to this family as *canonical* ALℓ_0CORE, since it contains CP as a special case. We further set the hyperparameters to standard default values a_0=b_0=e_0=1.0, f_0=10, and $\alpha_0^{(m)}$=0.1.

Baselines. We compare ALℓ_0CORE to three baselines: 1) CP using Gibbs sampling, 2) full Tucker using Gibbs sampling, and 3) full Tucker using EM. All three models adopt a Poisson likelihood, as given in the left-hand side of eq. (10). The Gibbs sampling baselines further adopt analogous prior distributions. We implement them in Julia using much of the same code as for ALℓ_0CORE, and run them using the same settings. For the third baseline, we use a recent state-of-the-art implementation of EM for 3-mode Tucker known as NNTuck (Aguiar et al., 2023), and adapt its Python code for 4-mode Tucker .

Evaluation criteria. For each tensor and train-test split, we fit each model to the training set, and evaluate the pointwise predictive density (PPD) it assigns to heldout data—this is defined as

$$\text{PPD}(\mathcal{H}) = \exp\left(\frac{1}{|\mathcal{H}|}\sum_{\mathbf{d}\in\mathcal{H}} \log\left[\frac{1}{S}\sum_{s=1}^{S} \text{Pois}(y_{\mathbf{d}}; \widehat{y}_{\mathbf{d}}^{(s)})\right]\right),$$

where $\mathcal{H}$ is the set of multi-indices in the test set. We also examine PPD on the positive-only part of the test set—i.e., PPD$(\mathcal{H}_{>0})$ where $\mathcal{H}_{>0} \equiv \{\mathbf{d} \in \mathcal{H} : y_{\mathbf{d}} > 0\}$.

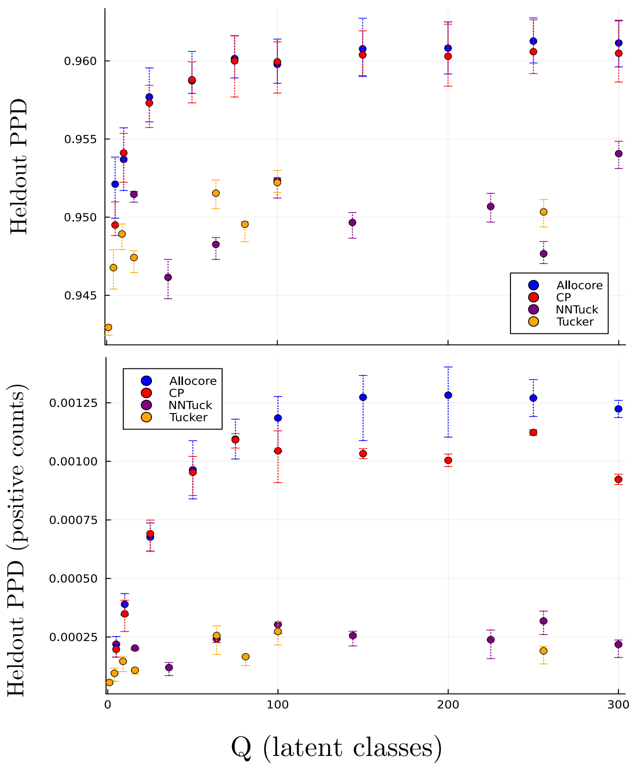


Figure 3: PPD across models on the full heldout set (top) and the positive-only heldout (bottom) across Q. Error bars span the interquartile range across masks.

6.3 Predictive Results

We first examine the predictive performance of ALℓ_0CORE as Q increases. Note that as $Q \to |\boldsymbol{\Lambda}|$, AL$\ell_0$CORE becomes full Tucker decomposition. We can thus ask: at what fraction of core density $Q/|\boldsymbol{\Lambda}|$ (and thus fraction of computational cost), does ALℓ_0CORE achieve all the benefits of Tucker?

We consider both large and small core shapes. The small core shape we use is the one used by Schein et al. (2016) for the same data: $20 \times 20 \times 6 \times 3$. The large core shape is $50 \times 50 \times 6 \times 10$. For each core shape, we then consider values of Q ranging from 5 to 400. Note that $Q = 400$ still represents only a small fraction of full Tucker for both the small (5.56% dense) and large (0.27% dense) cores.

Figure 2 plots heldout PPD as a function of wall-clock time for all unique combinations of budget Q, core shape, and train-test splits of TERRIER. Performance plateaus very early for both core shapes, around $Q = 150$, which takes approximately 1 hour for 5,000 MCMC iterations. By comparison, one iteration of EM for full Tucker (NNTuck), using the small core shape, takes around 125 seconds. Aguiar et al. (2023) recommend running NNTuck for 1,000 iterations, which would take around 1.5 days for a single run (they further recommend 20 random restarts). It is worth emphasizing that NNTuck represents the state-of-the-art for Poisson Tucker decomposition and that its severe run-time in this setting reflects the fundamental problem with all Tucker decomposition.

We further compare ALℓ_0CORE to all of the CP and Tucker decomposition baselines. To fairly compare across these models, we set Q in CP and ALℓ_0CORE to be equal and further set the shape of the core tensor for the full Tucker methods (Gibbs sampling and NNTuck) such that the core has roughly Q total entries. We consider $Q \in \{5, 10, 25, 50, 75, 100, 150, 200, 250, 300\}$ and fit every model with every Q to every train-test split.

Figure 3 reports the results. ALℓ_0CORE performs substantially better than either Tucker approach. Moreover, as Q increases, CP's performance degrades, suggesting possible overfitting, while ALℓ_0CORE appears robust. We note this tells only part of the story, as the full Tucker approaches took substantially longer to run, even while performing substantially worse.

We report in the appendix a number of other predictive results, similar to those reported here, as well as a toy synthetic data experiment seeking to better understand ALℓ_0CORE's shrinkage properties.

6.4 Qualitative Results

Finally, we fit ALℓ_0CORE to the fully-observed ICEWS dataset and explore its inferred latent structure. For this, we binned time steps binned by month (rather than year) to explore more fine-grained temporal structure, which yields $T = 228$ time steps, and a resulting count tensor $\mathbf{Y}$ of shape $249 \times 249 \times 20 \times 228$.

In this setting, the core tensor has shape $C \times D \times K \times R$. Following Schein et al. (2016), we interpret C and D to be the number of *communities* of sender and receiver countries, respectively, K to be the number of *action topics*, and R to be the number of *temporal regimes*. We set these to $C = D = 50$, $K = 20$, and $R = 300$. The number of possible latent classes is C^2DK =15,000,000. By comparison Schein et al. (2016) used a $20 \times 20 \times 6 \times 3$ core, resulting in 7,200, while Schein et al. (2015) used a CP decomposition with only 50 latent classes. ALℓ_0CORE is able to explore such a large latent space due to its sparsity constraint, which we set here to $Q = 400$.

We ran 10,000 MCMC iterations and used the last posterior sample for exploratory analysis. We first inspected how many classes were inferred to be non-zero. Recall that Q is merely an upper bound on the non-zero classes. We find that only 215 (of the possible 400) core elements are non-zero. Moreover, we find that the inferred non-zero values $\lambda_{\mathbf{k}}$ for those 215 classes are highly unequal, with only 44 classes accounting for 50% of the total mass i.e., $\sum_{\mathbf{k}} \lambda_{\mathbf{k}}$, and 150 accounting for 95%. It is intriguing that ALℓ_0CORE shrinks to this particular range of 50–150 classes, which is the range typically used to apply CP to the same data.

We provide visualizations of the top 100 inferred classes, ranked by their value $\lambda_{\mathbf{k}}$, in section B.3 and a more detailed visualization of six of those classes in fig. 4. Each class in this setting corresponds to a location (c, d, k, r) in the core for which the inferred value $\lambda_{c,d,k,r} \equiv \lambda^{(r)}_{c \xrightarrow{k} d}$ is non-zero. We can interpret a given class by inspecting the four factor vectors it indexes—i.e., the c^{th} and d^{th} community factors, $\boldsymbol{\phi}^{(1)}_c \in \mathbf{R}^V_+$ and $\boldsymbol{\phi}^{(2)}_d \in \mathbf{R}^V_+$, over sender and receiver countries, respectively, the k^{th} topic factor $\boldsymbol{\phi}^{(3)}_k \in \mathbf{R}^A_+$ over actions, and the r^{th} regime factor $\boldsymbol{\phi}^{(4)}_r \in \mathbf{R}^T_+$ over time steps. Since these are non-negative, simply inspecting the largest elements in each factor tends to yield highly intuitive interpretations.

In exploring the inferred classes, we found one particularly interesting pattern. For multiple major wars that occurred during the observation window 1995–2013, we find two pairs of sender and receiver communities, (c, d) and (c', d'), which correspond to the two sets of belligerents on either side of the war. Furthermore, for all these wars, we find a shared cadence involving three inferred classes. In the first class, community c takes actions in topic $k = 19$ to community d'—i.e., $c \xrightarrow{19} d'$—for some regime r. This topic $k = 19$, places most of its weight on the action FIGHT. In the second class, the same c takes actions in topic $k = 9$ to the same d'—i.e., $c \xrightarrow{9} d'$—with this topic placing its weight on cooperative actions, such as INTEND TO COOPERATE. In the third class, sender community c' (which corresponds roughly to the same belligerents in receiver community d') takes actions in topic $k = 14$ towards community d—i.e., $c' \xrightarrow{14} d$. We find this same cadence of three classes—i.e., $c \xrightarrow{19} d'$, $c \xrightarrow{9} d'$, $c' \xrightarrow{14} d$—for five pairs of (c, d) and (c', d'), which correspond to belligerents on either side of 1) the 2001 US-led invasion of Afghanistan, 2) the 2003 US-led invasion of Iraq, 3) the 2006 Israel–Hezbollah war, 4) the Israeli-Palestinian conflict, and 5) the Yugoslav wars. We visualize this cadence for two of these wars in figure 4. We emphasize that surfacing this kind of pattern is facilitated by ALℓ_0CORE's ability to share factors across classes (something CP cannot do) but to do so *selectively* without suffering Tucker's "exponential blowup".

Inferred non-zero core elements

rank	$\lambda^{(r)}_{c \xrightarrow{k} d}$	c	d	k	r
1	11.03	2	2	10	252
2	10.36	31	31	9	81
3	10.36	9	9	12	159
4	10.13	33	33	9	233
5	9.58	37	37	12	87
6	9.36	12	12	2	162
7	8.9	43	43	10	43
8	8.57	41	41	2	41
9	8.35	15	15	10	115
10	8.13	11	11	16	61
	…		…		
20	6.65	49	49	9	49
21	6.61	14	14	2	14
22	6.55	23	23	9	273
23	6.35	46	46	9	196
24	6.25	22	22	2	222
25	**6.24**	**40**	**40**	**19**	**40**
26	**6.23**	**19**	**19**	**14**	**269**
27	6.14	43	43	10	143
28	**6.14**	**6**	**6**	**19**	**106**
29	5.92	32	32	9	132
30	5.85	47	47	12	62
31	5.77	24	24	14	124
32	5.76	9	9	12	59
33	5.67	13	13	10	252
34	5.62	25	25	9	25
35	5.57	27	27	9	177
36	**5.53**	**17**	**17**	**14**	**67**
37	5.42	30	30	9	30
38	5.36	5	5	10	255
39	5.34	3	3	16	3
40	5.24	10	10	10	210
	…		…		
65	**3.78**	**40**	**40**	**9**	**90**
	…		…		
93	**2.51**	**6**	**6**	**9**	**156**
	…		…		
211	0.004	19	38	15	88
212	0.003	19	38	15	295
213	0.001	19	38	10	295
214	0.001	33	38	10	295
215	0.0004	19	38	10	88

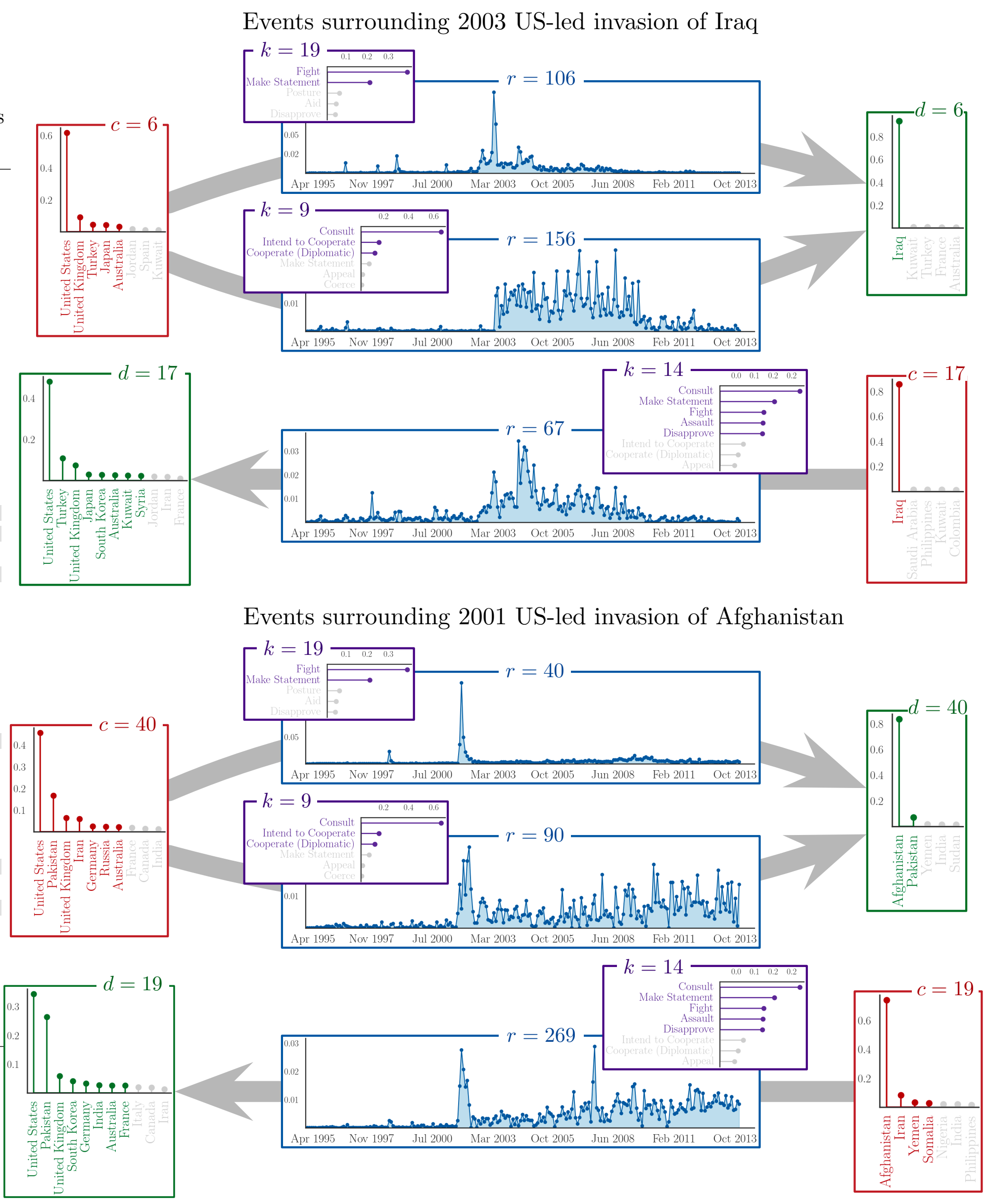


Figure 4: Example of the latent class structure inferred by ALℓ_0CORE with $Q = 400$ on the ICEWS dataset. *Left:* The inferred non-zero locations in the core tensor, with rows sorted according to the inferred values $\lambda^{r}_{c \xrightarrow{k} d}$. *Right:* Two sets of three inferred classes, each corresponding to a major war, and each following the same cadence—i.e., $c \xrightarrow{19} d'$, $c \xrightarrow{9} d'$, $c' \xrightarrow{14} d$—described in section 6.4. The blue stem plots depict all elements in a given time-step factor, while red, green, and purple stem plots depict only the largest elements in a given sender, receiver, and action factor, respectively; in these, stems are greyed out when normalized values fall below 0.02.

7 CONCLUSION

The Tucker decomposition's qualitative appeal has always been difficult to practically achieve due to the "exponential blowup" associated with increasing the number of latent factors. For many problems involving large sparse count tensors, such as the analysis of complex multilayer networks, where modelers often seek to represent many communities and their interactions, the disparity between Tucker's conceptual appeal and its practicality is stark. The ALℓ_0CORE decomposition remedies this disparity by leveraging sparsity to decouple the size of the core tensor from the computational cost of inference. Although particularly motivated for the analysis of sparse count data, the motif of allocated sparsity could be incorporated into many other models and methods, development of which are promising avenues of future work.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2140001.

References

Aguiar, I., Taylor, D., and Ugander, J. (2023). A tensor factorization model of multilayer network interdependence.

Amjad, M., Misra, V., Shah, D., and Shen, D. (2019). mRSC: Multi-dimensional Robust Synthetic Control. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(2):1–27.

Bhattacharya, A. and Dunson, D. B. (2012). Simplex Factor Models for Multivariate Unordered Categorical Data. *Journal of the American Statistical Association*, 107(497):362–377.

Bolte, J., Sabach, S., and Teboulle, M. (2014). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494.

Boschee, E., Lautenschlager, J., O'Brien, S., Shellman, S., Starz, J., and Ward, M. (2015). ICEWS Coded Event Data.

Cemgil, A. T. (2009). Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009:785152.

Chi, E. C. and Kolda, T. G. (2012). On Tensors, Sparsity, and Nonnegative Factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299.

Cichocki, A., Mandic, D., De Lathauwer, L., Zhou, G., Zhao, Q., Caiafa, C., and Phan, H. A. (2015). Tensor Decompositions for Signal Processing Applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163.

Cohen, J. E. and Gillis, N. (2019). Nonnegative Low-rank Sparse Component Analysis. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8226–8230.

De Bacco, C., Power, E. A., Larremore, D. B., and Moore, C. (2017). Community detection, link prediction, and layer interdependence in multilayer networks. *Physical Review E*, 95(4):042317.

de Leeuw, D. (2020). *A Novel ℓ_0-Sparse Nonnegative Matrix Factorization Algorithm (ℓ_0-SNMF) for Community Detection. An Approach Using Iterative Majorization.* PhD thesis, Erasmus University Rotterdam, School of Economics.

Dunson, D. B. and Xing, C. (2009). Nonparametric Bayes Modeling of Multivariate Categorical Data. *Journal of the American Statistical Association*, 104(487):1042–1051.

Fang, S., Kirby, R. M., and Zhe, S. (2021). Bayesian streaming sparse Tucker decomposition. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 558–567.

Gerner, D., Jabr, R., and Schrodt, P. (2002). Conflict and Mediation Event Observations (CAMEO): A New Event Data Framework for the Analysis of Foreign Policy Interactions. *International Studies Association, New Orleans*.

Halterman, A., Irvine, J., Landis, M., Jalla, P., Liang, Y., Grant, C., and Solaimani, M. (2017). Adaptive scalable pipelines for political event data generation. In *2017 IEEE International Conference on Big Data*, pages 2879–2883.

Harshman, R. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-model factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

Hoff, P. D. (2015). Multilinear tensor regression for longitudinal relational data. *The Annals of Applied Statistics*, 9(3):1169–1193.

Hoff, P. D. (2016). Equivariant and Scale-Free Tucker Decomposition Models. *Bayesian Analysis*, 11(3).

Hoyer, P. (2002). Non-negative sparse coding. *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565.

Hoyer, P. O. (2004). Non-negative Matrix Factorization with Sparseness Constraints. *The Journal of Machine Learning Research*, 5:1457–1469.

Johndrow, J. E., Bhattacharya, A., and Dunson, D. B. (2017). Tensor decompositions and sparse log-linear models. *The Annals of Statistics*, 45(1):1–38.

Kiers, H. A. L. and Giordani, P. (2020). CANDECOMP/PARAFAC with zero constraints at arbitrary positions in a loading matrix. *Chemometrics and Intelligent Laboratory Systems*, 207:104145.

Kolda, T. G. and Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Marot, J., Fossati, C., and Bourennane, S. (2008). About advances in tensor data denoising methods. *EURASIP Journal on Advances in Signal Processing*, 2008:1–12.

Minhas, S., Hoff, P. D., and Ward, M. D. (2016). A new approach to analyzing coevolving longitudinal networks in international relations. *Journal of Peace Research*, 53(3):491–505.

Morup, M., Madsen, K. H., and Hansen, L. K. (2008). Approximate L_0 constrained non-negative matrix and tensor factorization. In *2008 IEEE International Symposium on Circuits and Systems*, pages 1328–1331.

Nadisic, N., Cohen, J., Vandaele, A., and Gillis, N. (2022). Matrix-wise ℓ_0-constrained sparse nonnegative least squares. *Machine Learning*, 111:1–43.

Park, M., Jang, J.-G., and Sael, L. (2021). VEST: Very Sparse Tucker Factorization of Large-Scale Tensors. In *2021 IEEE International Conference on Big Data and Smart Computing*, pages 172–179.

Peharz, R. and Pernkopf, F. (2012). Sparse nonnegative matrix factorization with ℓ_0-constraints. *Neurocomputing*, 80(1):38–46.

Schein, A. (2019). Allocative Poisson factorization for computational social science.

Schein, A., Paisley, J., Blei, D. M., and Wallach, H. (2015). Bayesian Poisson Tensor Factorization for Inferring Multilateral Relations from Sparse Dyadic Event Counts. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Schein, A., Zhou, M., Blei, D. M., and Wallach, H. (2016). Bayesian Poisson Tucker decomposition for learning the structure of international relations. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2810–2819.

Schrodt, P., Neack, L., Haney, P., and Hey, J. (1995). Event data in foreign policy analysis. In *Foreign Policy Analysis: Continuity and Change in Its Second Generation*, pages 145–166. Prentice Hall.

Stoehr, N., Radford, B. J., Cotterell, R., and Schein, A. (2023). The Ordered Matrix Dirichlet for State-Space Models. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 1888–1903.

Tomasi, G. and Bro, R. (2005). PARAFAC and missing values. *Chemometrics and Intelligent Laboratory Systems*, 75(2):163–180.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Welling, M. and Weber, M. (2001). Positive tensor factorization. *Pattern Recognition Letters*, 22(12):1255–1261.

Wu, F., Cai, J., Wen, C., and Tan, H. (2022). Co-sparse Non-negative Matrix Factorization. *Frontiers in Neuroscience*, 15.

Xiong, L., Chen, X., Huang, T.-K., Schneider, J., and Carbonell, J. G. (2010). Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 211–222. SIAM.

Yıldırım, S., Kurutmaz, M. B., Barsbey, M., Şimşekli, U., and Cemgil, A. T. (2020). Bayesian allocation model: marginal likelihood-based model selection for count tensors. *IEEE Journal of Selected Topics in Signal Processing*, 15(3):560–573.

Zhang, X. and Ng, M. K. (2022). Sparse nonnegative tucker decomposition and completion under noisy observations.

Zhou, J., Bhattacharya, A., Herring, A., and Dunson, D. (2015). Bayesian factorizations of big sparse tensors. *Journal of the American Statistical Association*, 110(512):1562–1576.

Checklist

1. For all models and algorithms presented, check if you include:
   - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**
   - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**
   - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **No**
2. For any theoretical claim, check if you include:
   - (a) Statements of the full set of assumptions of all theoretical results. **Not Applicable**
   - (b) Complete proofs of all theoretical results. **Not Applicable**
   - (c) Clear explanations of any assumptions. **Yes**
3. For all figures and tables that present empirical results, check if you include:
   - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Include the Al$\ell_0$core source code.**
   - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes** combination in section 6.2 and Supplementary Material.
   - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**
   - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**, cloud provider.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   - (a) Citations of the creator If your work uses existing assets. **Yes**
   - (b) The license information of the assets, if applicable. **Not Applicable**
   - (c) New assets either in the supplemental material or as a URL, if applicable. [**Yes**/No/Not Applicable] Provide ALℓ_0CORE source code in supplemental material.
   - (d) Information about consent from data providers/curators. **No**
   - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   - (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
   - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

A DERIVATIONS AND CODE

A.1 Complete Conditional Derivations

We detail and derive the complete conditional updates for posterior inference under Bayesian Poisson ALℓ_0CORE, leveraging Poisson additivity to streamline the following derivations and use ($\propto_x$) to denote *proportional to in x* (equal up to a constant c_0, where c_0 does not depend on x).

Poisson additivity. For $i \in [n]$, if $x_i \overset{\text{ind.}}{\sim} \text{Pois}(c\theta_i)$, then marginally, $x_\bullet \equiv \sum_{i=1}^n x_i \sim \text{Pois}(c\sum_{i=1}^n \theta_i)$.

Complete conditional for $\phi^{(m)}_{d,k}$. For each $m \in [M]$, $\mathrm{k}_m \in [K_m]$, $d_m \in [D_m]$, we leverage the Poisson additivity of the $\mathrm{y}_{\mathbf{d},q}$ as follows.

$$\mathrm{y}^{(m)}_{d_m \mathrm{k}_m} = \sum_{q:\mathrm{k}_{q,m}=\mathrm{k}_m} \sum_{d' \in \mathcal{D}, d'_m = d_m} \mathrm{y}_{d'_q}\,; \tag{16}$$

$$\mathrm{y}^{(m)}_{d_m \mathrm{k}_m} \sim \text{Pois}(\phi^{(m)}_{d_m,\mathrm{k}_m} \underbrace{\sum_{q:\mathrm{k}_{q,m}=\mathrm{k}_m} \lambda_q \sum_{d' \in \mathcal{D}, d'_m = d_m} \prod_{m' \neq m} \phi^{(m')}_{d_{m'} \mathrm{k}_{q,m'}}}_{c^{(m)}_{d_m,\mathrm{k}_m}})\,, \tag{17}$$

$$\mathrm{y}^{(m)}_{d_m \mathrm{k}_m} \sim \text{Pois}(\phi^{(m)}_{d_m,\mathrm{k}_m} c^{(m)}_{d_m,\mathrm{k}_m})\,; \tag{18}$$

where for fixed m, $\left(\mathrm{y}_{d_m,\mathrm{k}_m}\right)_{\mathrm{k}_m \in [K_m], d_m \in [D_m]}$ are mutually independent.
In its complete conditional, factor matrix entry $\phi^{(m)}_{d,k}$ is Gamma-distributed:

$$\begin{aligned}
P(\phi^{(m)}_{d,k} \mid -) &= P(\phi^{(m)}_{d,k} \mid (\Phi^{(m')})_{m' \neq m}, (\lambda_q, (\mathrm{y}_{\mathbf{d},q}), (\mathrm{k}_{q,m'})_{m'=1}^M)_{q=1}^Q) && (19)\\
&= P(\phi^{(m)}_{d,k} \mid \mathrm{y}^{(m)}_{d,k}, c^{(m)}_{d,k}) && (20)\\
&\propto_{\phi^{(m)}_{d,k}} P(\mathrm{y}^{(m)}_{d,k} \mid c^{(m)}_{d,k}, \phi^{(m)}_{d,k}) P(c^{(m)}_{d,k} \mid \phi^{(m)}_{d,k}) P(\phi^{(m)}_{d,k}) && (21)\\
&\propto_{\phi^{(m)}_{d,k}} P(\mathrm{y}^{(m)}_{d,k} \mid c^{(m)}_{d,k}, \phi^{(m)}_{d,k}) P(c^{(m)}_{d,k}) P(\phi^{(m)}_{d,k}) && (22)\\
&\propto_{\phi^{(m)}_{d,k}} P(\mathrm{y}^{(m)}_{d,k} \mid c^{(m)}_{d,k}, \phi^{(m)}_{d,k}) P(\phi^{(m)}_{d,k}) && (23)\\
&\propto_{\phi^{(m)}_{d,k}} \text{Pois}(\mathrm{y}^{(m)}_{d,k}; c^{(m)}_{d,k} \cdot \phi^{(m)}_{d,k}) \Gamma(\phi^{(m)}_{d,k}; c_0, d_0) && (24)\\
&\propto_{\phi^{(m)}_{d,k}} \Gamma(\phi^{(m)}_{d,k}; c_0 + \mathrm{y}^{(m)}_{d,k}, d_0 + c^{(m)}_{d,k}) \qquad \square && (25)
\end{aligned}$$

Complete conditional for $\left(\mathrm{y}_{\mathbf{d},q}\right)_{q=1}^Q$. To derive the conditionals for the latent counts $\left(\mathrm{y}_{\mathbf{d},q}\right)_{q=1}^Q$, we first observe that their sum $\mathrm{y}_{\mathbf{d}} = \sum_{q=1}^Q \mathrm{y}_{\mathbf{d},q}$ is observed. Due to the relationship between the Poisson and multinomial distributions, we have that

$$\left(\left(\mathrm{y}_{\mathbf{d},q}\right)_{q=1}^Q \mid -\right) \sim \text{Multi}(\mathrm{y}_{\mathbf{d}}, \Pi_{\mathbf{d}})\,, \tag{26}$$

$$\Pi_{\mathbf{d}} \equiv \left(\frac{\lambda_q \prod_{m=1}^M \phi^{(m)}_{d_m \mathrm{k}_{q,m}}}{\sum_{q'=1}^Q \lambda_{q'} \prod_{m=1}^M \phi^{(m)}_{d_m \mathrm{k}_{q',m}}}\right)_{q=1}^Q \qquad \square \tag{27}$$

where $\Pi_{\mathbf{d}} \in \mathbb{R}^Q, \sum_{q=1}^Q \Pi_{\mathbf{d},q} = 1\,, \Pi_{\mathbf{d},q} \geq 0$.

Complete conditional for $\boldsymbol{\pi}^{(m)}$. By Dirichlet-multinomial conjugacy, for each $\boldsymbol{\pi}^{(m)}$,

$$(\boldsymbol{\pi}^{(m)} \mid -) \sim \text{Dirichlet}(\boldsymbol{\alpha}^{(m)})\,, \quad \boldsymbol{\alpha}^{(m)} \in \mathbb{R}^{K_m}, \tag{28}$$

$$\text{where } \boldsymbol{\alpha}^{(m)}_k = \alpha_0 + \sum_{q=1}^Q 1\{\mathrm{k}_{q,m} = k\} \qquad \square \tag{29}$$

Complete conditional for $\boldsymbol{\lambda_q}$. Define y_q as $\mathrm{y}_q \equiv \sum_{\mathbf{d}} \mathrm{y}_{\mathbf{d},q}$, a sum over the latent counts $(\mathrm{y}_{\mathbf{d},q})$.

$$\begin{aligned}
P(\lambda_q \mid -) &= P(\lambda_q \mid (\Phi^{(m)})_{m=1}^M, \mathrm{y}_q, (\mathrm{k}_{q,m})_{m=1}^M) && (30)\\
&\propto_{\lambda_q} P((\Phi^{(m)})_{m=1}^M, \mathrm{y}_q, (\mathrm{k}_{q,m})_{m=1}^M \mid \lambda_q) P(\lambda_q) && (31)\\
&= P(\lambda_q) P(\mathrm{y}_q \mid (\Phi^{(m)}, (\mathrm{k}_{q,m})_{m=1}^M, \lambda_q) P((\Phi^{(m)})_{m=1}^M \mid (\mathrm{k}_{q,m})_{m=1}^M, \lambda_q) P((\mathrm{k}_{q,m})_{m=1}^M \mid \lambda_q) && (32)\\
&= P(\lambda_q) P(\mathrm{y}_q \mid (\Phi^{(m)}, \mathrm{k}_{q,m})_{m=1}^M, \lambda_q) P((\Phi^{(m)})_{m=1}^M) P((\mathrm{k}_{q,m})_{m=1}^M) && (33)\\
&\propto_{\lambda_q} P(\lambda_q) P(\mathrm{y}_q \mid (\Phi^{(m)}, \mathrm{k}_{q,m})_{m=1}^M, \lambda_q) && (34)\\
&\propto_{\lambda_q} \Gamma(\lambda_q; a_0, b_0) \mathrm{Pois}(\mathrm{y}_q; \lambda_q \prod_{m=1}^M \sum_{d_m=1}^{D_m} \phi^{(m)}_{d_m \mathrm{k}_{q,m}}) && (35)\\
&\propto_{\lambda_q} \lambda_q^{a_0-1} e^{-b_0\lambda_q} e^{-\lambda_q \prod_{m=1}^M \sum_{d_m=1}^{D_m} \phi^{(m)}_{d_m \mathrm{k}_{q,m}}} \lambda_q^{\mathrm{y}_q} && (36)\\
&\propto_{\lambda_q} \lambda_q^{a_0+\mathrm{y}_q-1} e^{-\lambda_q [b_0 + \prod_{m=1}^M \sum_{d_m=1}^{D_m} \phi^{(m)}_{d_m \mathrm{k}_{q,m}}]} && (37)
\end{aligned}$$

implying that (38)

$$(\lambda_q \mid -) \sim \Gamma\left(a_0 + \mathrm{y}_q, b_0 + \prod_{m=1}^M \sum_{d_m=1}^{D_m} \phi^{(m)}_{d_m \mathrm{k}_{q,m}}\right) \qquad \square \qquad (39)$$

Complete conditional for $\mathbf{k}_{\boldsymbol{q,m}}$. We define

$$\mathrm{y}_{d_m \bullet, q} = \sum_{d' \in \mathcal{D}, d'_m = d} \mathrm{y}_{\mathbf{d}',q} \sim \mathrm{Pois}(\phi^{(m)}_{d_m \mathrm{k}_{q,m}} \underbrace{\lambda_q \sum_{d' \in \mathcal{D}, d'_m = d} \prod_{m' \neq m} \phi^{(m')}_{d'_{m'} \mathrm{k}_{q,m'}}}_{c_{d_m,q}}) \qquad (40)$$

Then

$$\begin{aligned}
P(\mathrm{k}_{q,m} = k \mid -) &\propto_{\mathrm{k}_{q,m}} P(\mathrm{k}_{q,m} = k) P((\mathrm{y}_{d_m \bullet, q})_{d_m=1}^{D_m} \mid \mathrm{k}_{q,m} = k, \lambda_q, (\mathrm{k}_{q,m'})_{m' \neq m}, (\Phi^{(m)})_{m=1}^M) && (41)\\
&\propto_{\mathrm{k}_{q,m}} \boldsymbol{\pi}^{(m)}_k \prod_{d_m=1}^{D_m} \mathrm{Pois}(\mathrm{y}_{d_m \bullet, q}; \phi^{(m)}_{d_m,k} \lambda_q \sum_{d' \in \mathcal{D}, d'_m = d} \prod_{m' \neq m} \phi^{(m')}_{d'_{m'} \mathrm{k}_{q,m'}}) && (42)\\
&\propto_{\mathrm{k}_{q,m}} \boldsymbol{\pi}^{(m)}_k \prod_{d_m=1}^{D_m} \mathrm{Pois}(\mathrm{y}_{d_m \bullet, q}; \phi^{(m)}_{d_m,k} c_{d_m,q}) && (43)\\
&= \frac{\boldsymbol{\pi}^{(m)}_k \prod_{d_m=1}^{D_m} \mathrm{Pois}(\mathrm{y}_{d_m \bullet, q}; \phi^{(m)}_{d_m,k} c_{d_m,q})}{\sum_{k'=1}^{K_m} \boldsymbol{\pi}^{(m)}_{k'} \prod_{d_m=1}^{D_m} \mathrm{Pois}(\mathrm{y}_{d_m \bullet, q}; \phi^{(m)}_{d_m,k'} c_{d_m,q})} \qquad \square && (44)
\end{aligned}$$

A.2 Code for Al$\ell_0$core

https://github.com/jhood3/allocore

B ADDITIONAL RESULTS

B.1 Synthetic Experiment

The generative model defined for Bayesian Poisson ALℓ_0CORE implicitly defines a prior on the latent dimensionality K_m of each mode M. As an exploratory exercise, we generate synthetic data to evaluate if ALℓ_0CORE can detect ground-truth latent dimensions $(K^*_m)_{m=1}^M$ and $||\Lambda^*||_0$ in the simplest of settings. Our synthetic design generates $\mathbf{Y}$ according to Bayesian Poisson ALℓ_0CORE,

$$\mathrm{y}_{\mathbf{d}} = \sum_{q=1}^Q \mathrm{y}_{\mathbf{d},q}, \quad \mathrm{y}_{\mathbf{d},q} \overset{\text{ind.}}{\sim} \mathrm{Pois}\left(\widehat{\mathrm{y}}_{\mathbf{d},q}\right), \quad \widehat{\mathrm{y}}_{\mathbf{d},q} \equiv \lambda_q \prod_{m=1}^M \phi^{(m)}_{\mathrm{d}_m, \mathrm{k}_{q,m}} \qquad (45)$$

Factor matrices $\Phi^{(1)}, \Phi^{(2)}$ are sampled column-wise as $\boldsymbol{\phi}^{(1)}_{:,\mathrm{k}_1}, \boldsymbol{\phi}^{(2)}_{:,\mathrm{k}_2} \sim \text{Dirichlet}(\mathbf{0.01})$ and scaled such that each column sums to 5 (each entry is multiplied by 5). We require heterogeneity over the columns of $\Phi^{(3)}$, fixing $\boldsymbol{\phi}^{(3)}_{:,1} = (2.5, 0.5, 1, 0.5, 0.5)$, $\boldsymbol{\phi}^{(3)}_{:,2} = (0.5, 2.5, 1, 0.5, 0.5)$. Setting $K_m = Q = 50$, we fit ALℓ0CORE to $\mathbf{Y}$. Figure 5 shows the trajectory of sampled $\{(K^{(s)}_m)^3_{m=1}, Q^{(s)}\}^S_{s=1}$ during training. ALℓ0CORE infers core tensors with effective shape approximate to that of $\boldsymbol{\Lambda}^*$. Histograms over samples estimate the marginal posterior distribution for each K_m and Q. While ALℓ0CORE overestimates K^*_3, the posterior samples concentrate around K^*_1, K^*_2, and Q^*, indicating recovery of ground truth parameters. We leave for future work theoretical investigation these properties of ALℓ0CORE.

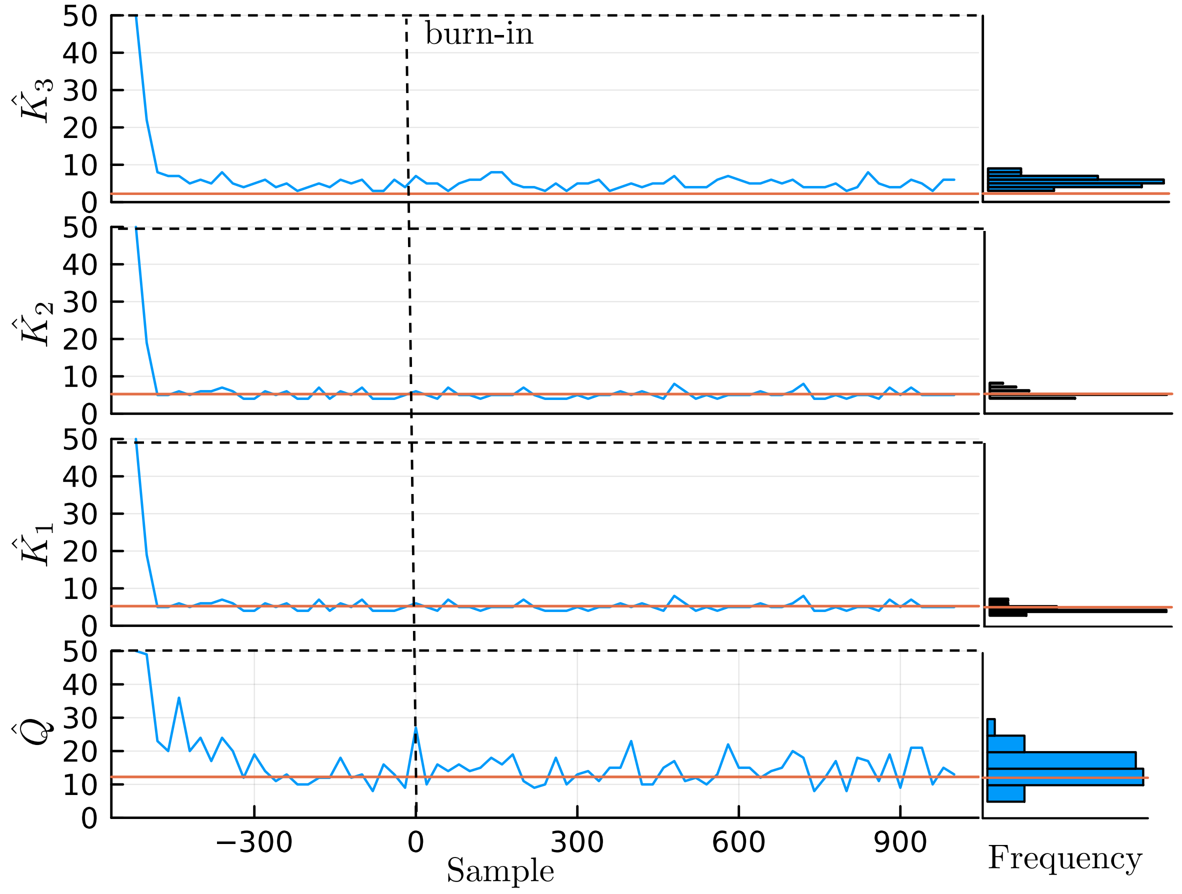


Figure 5: Posterior estimates of effective dimensionalities K^*_1, K^*_2 and K^*_3, and number of classes Q^* in a synthetic setting. *Left:* Trace plot from the Gibbs sampler. *Right:* Histograms of posterior samples. The red line denotes the ground truth value.

B.2 Quantitative Results

Supplementary quantitative results on the ICEWS dataset are given in fig. 6.

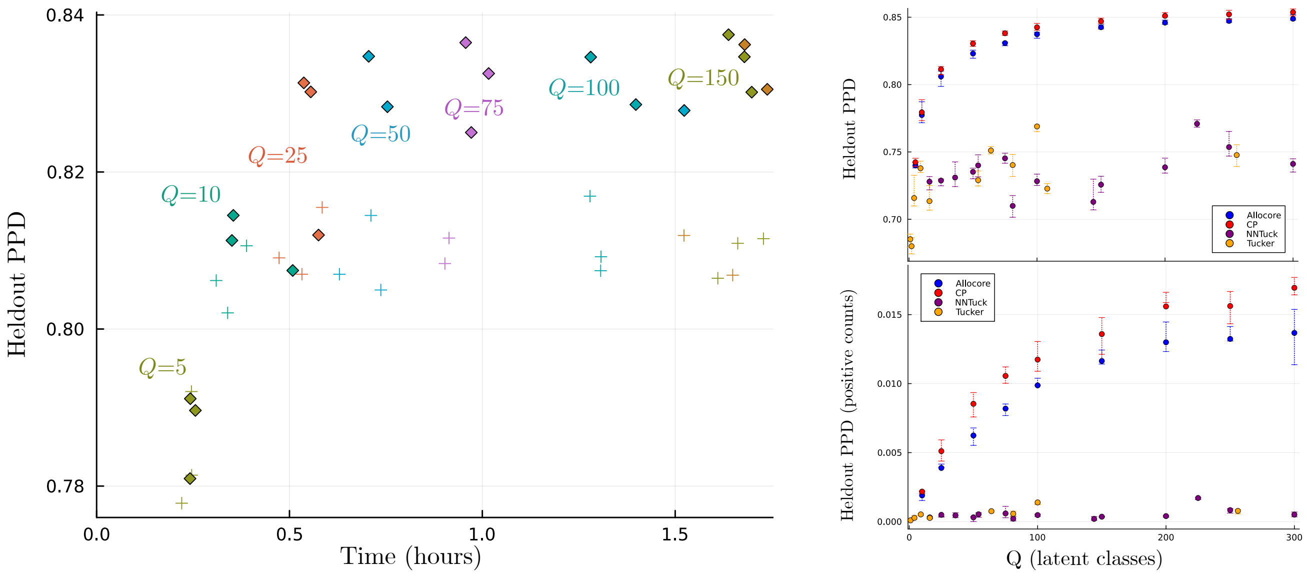


Figure 6: *Left:* PPD by wall-clock time (ICEWS data). Each point is an ALℓ_0CORE model run for 5,000 iterations, with a unique combination of Q, denoted by color, and core tensor size, denoted by $\diamond$- (small) versus +-shaped (large) markers. Performance plateaus early suggesting that sparse ALℓ_0CORE achieves the same performance as full Tucker at only a fraction of the cost. *Right:* Heldout PPD on the full heldout set (top) and the heldout set restricted to positive counts (bottom) across Q. ALℓ_0CORE is plotted in blue, CP in red, NNTuck in purple, and Tucker in yellow. Error bars span the interquartile range across masks.

B.3 Qualitative Results

We plot each learned latent class for an ALℓ_0CORE model fitted to ICEWS monthly data, a tensor of size $249 \times 249 \times 20 \times 228$, with $Q = 400$, $C = 50$, $D = 50$, $K = 20$, and $R = 300$, ordered by the non-zero core values $\lambda_{\mathbf{k}}$ in descending order.

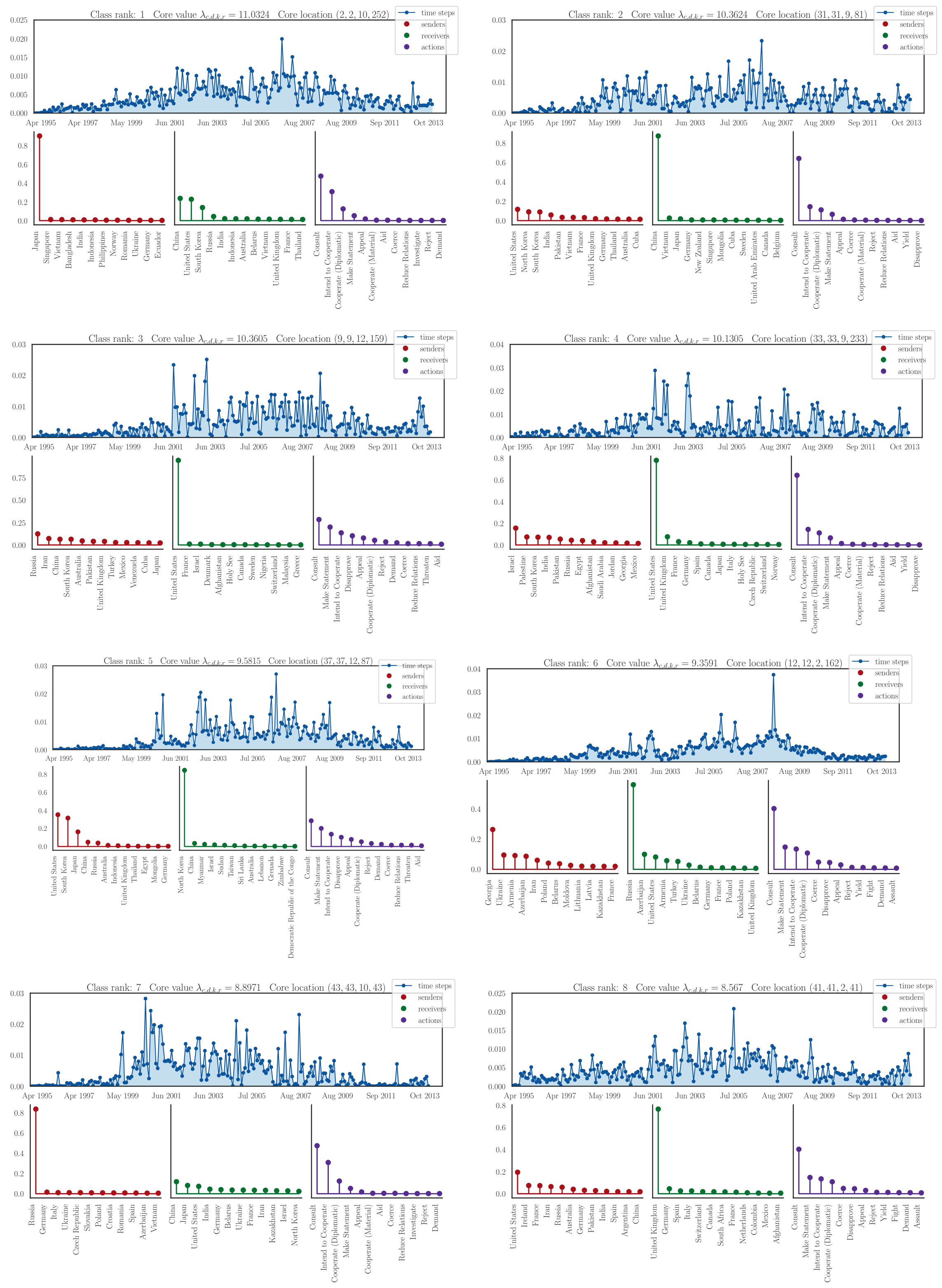


Figure 7: Inferred latent classes 1–8.

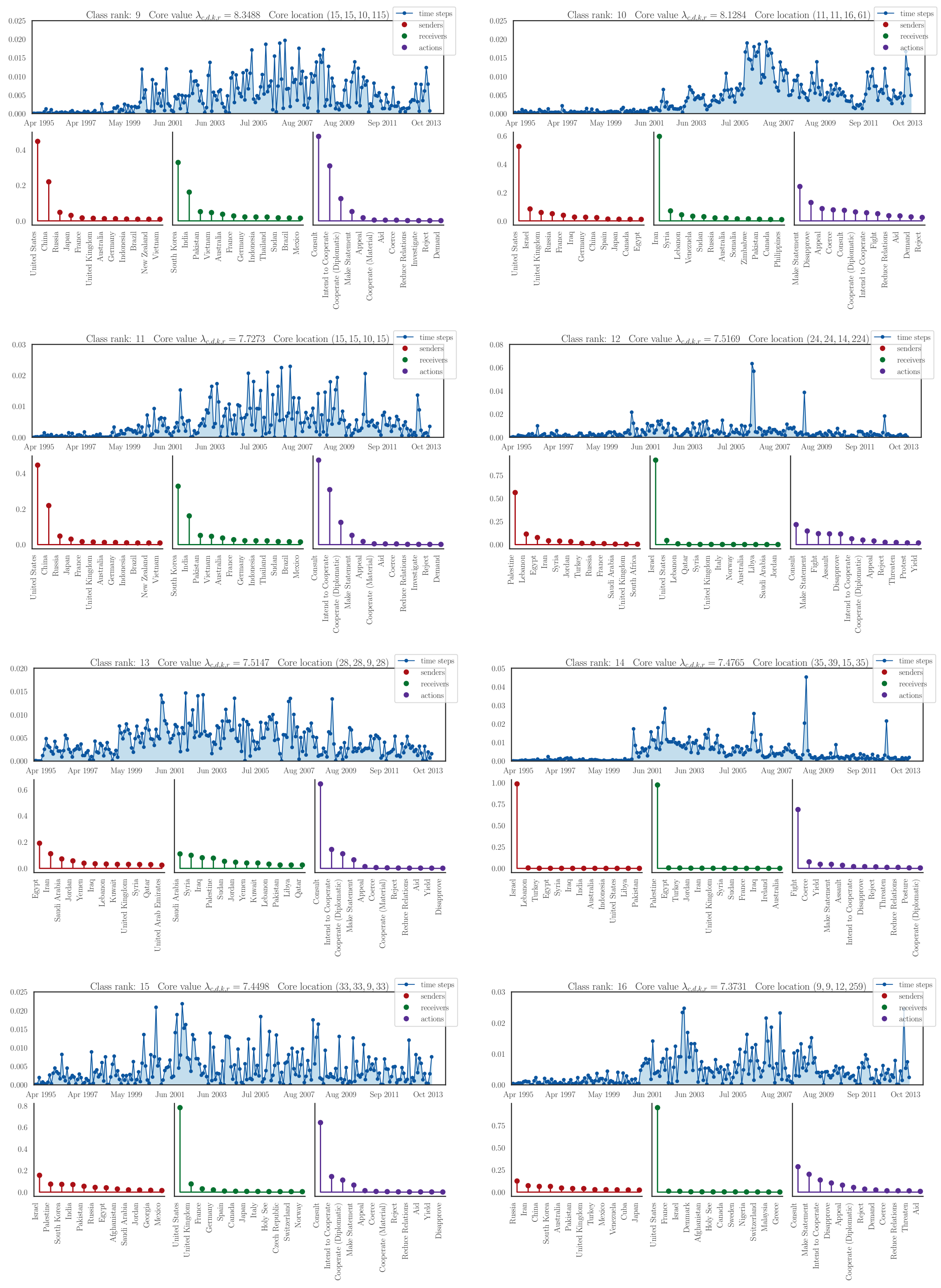


Figure 8: Inferred latent classes 9–16.

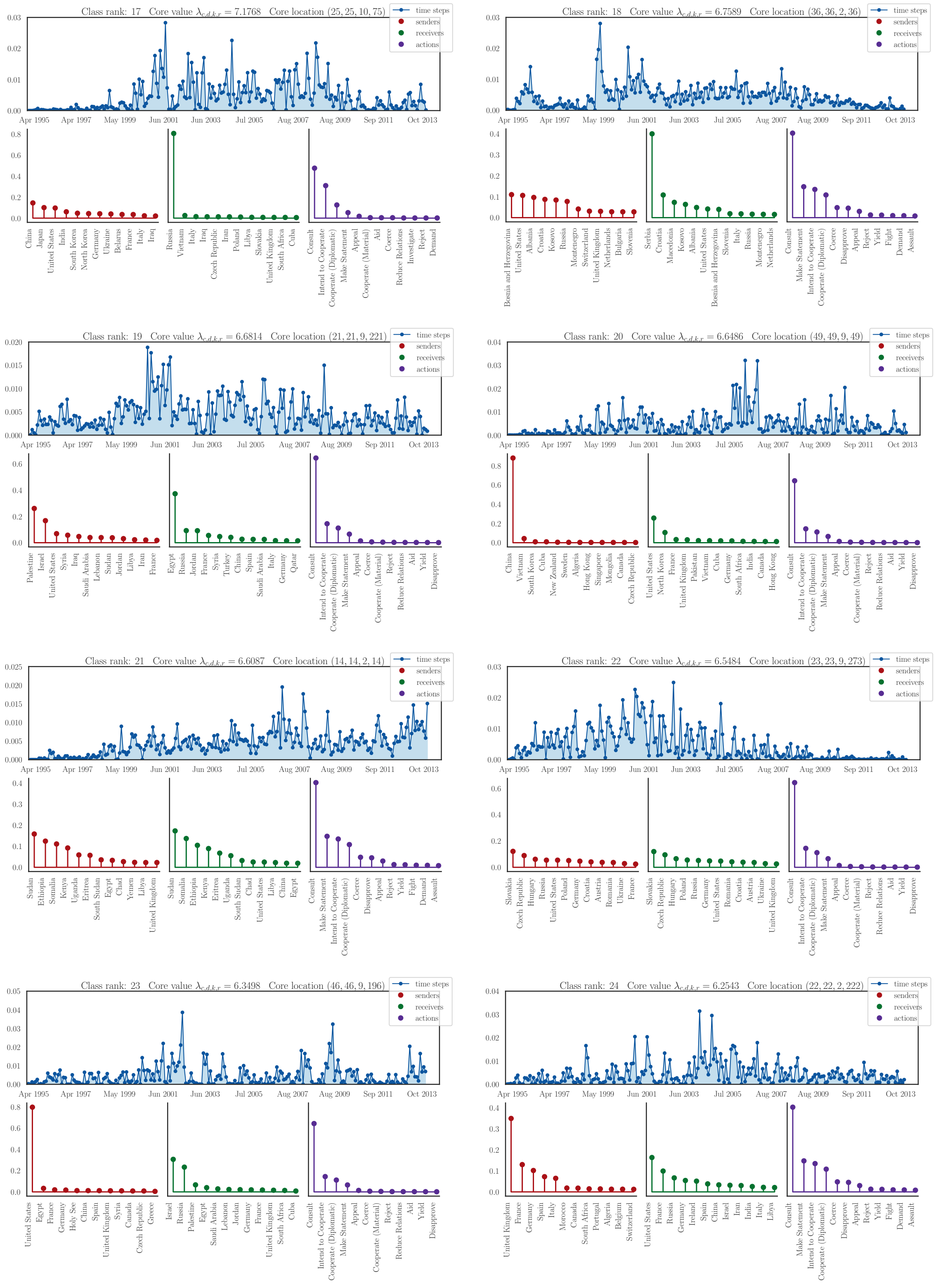


Figure 9: Inferred latent classes 17–24.

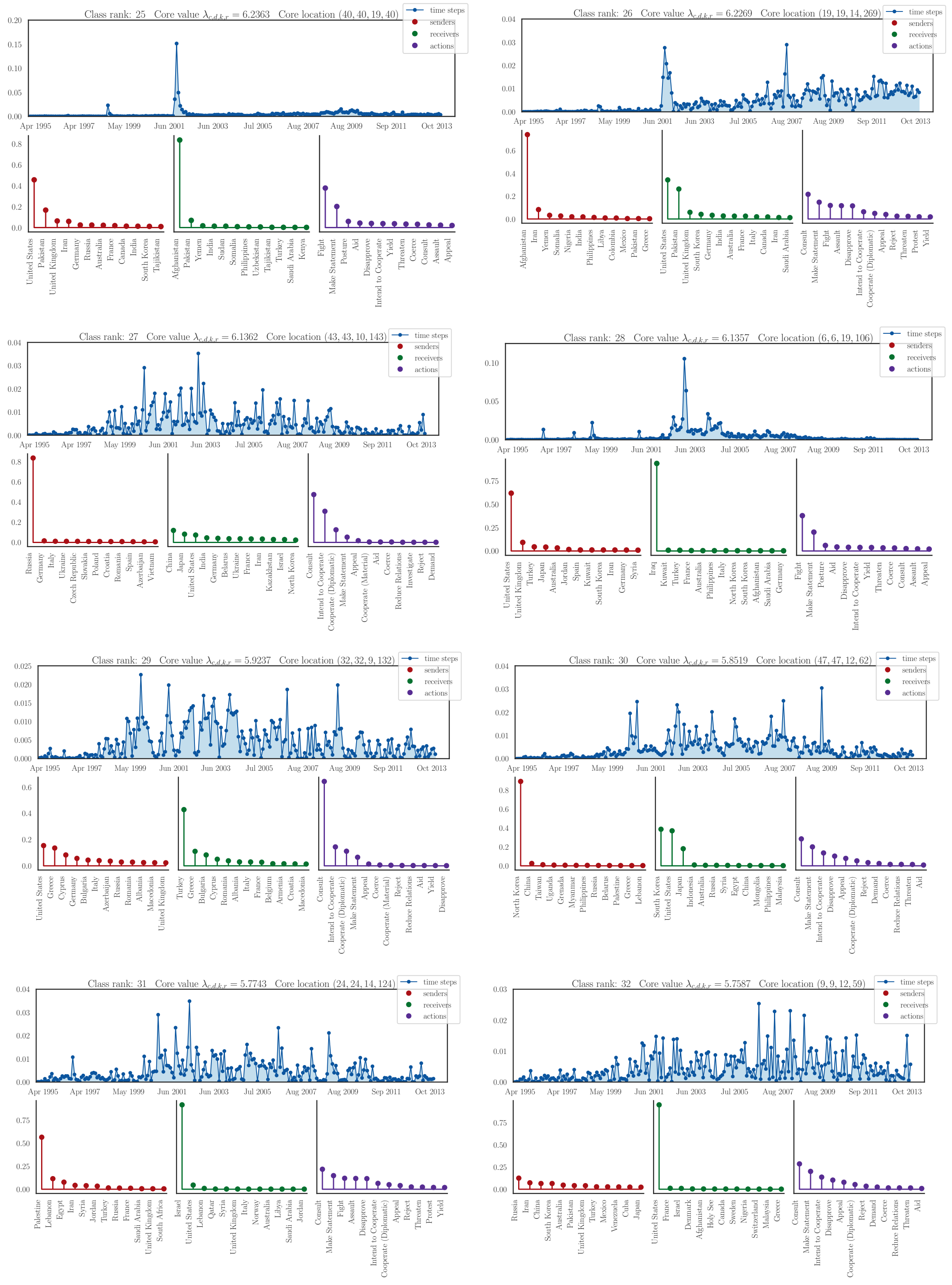


Figure 10: Inferred latent classes 25–32.

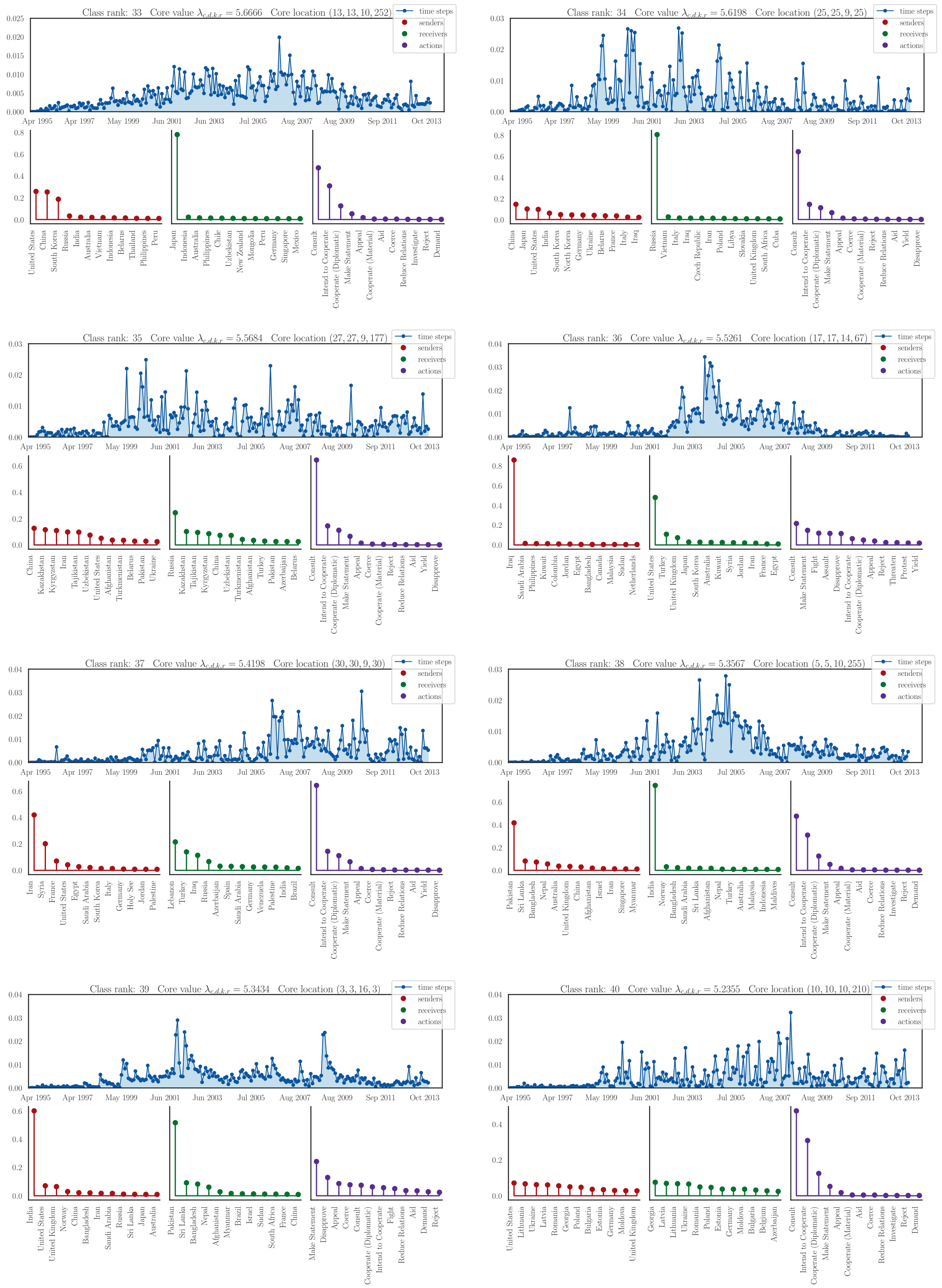


Figure 11: Inferred latent classes 33–40.

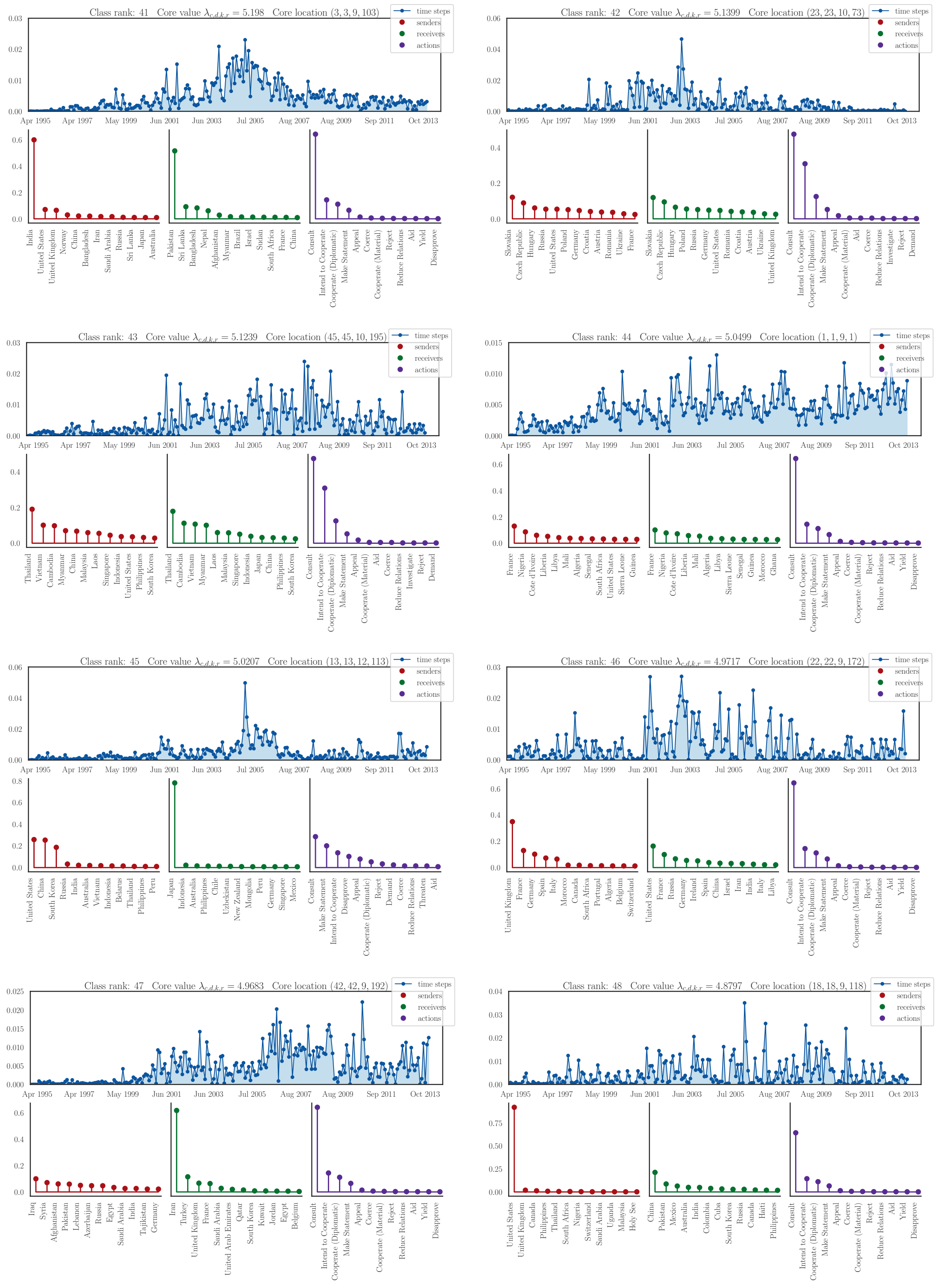


Figure 12: Inferred latent classes 41–48.

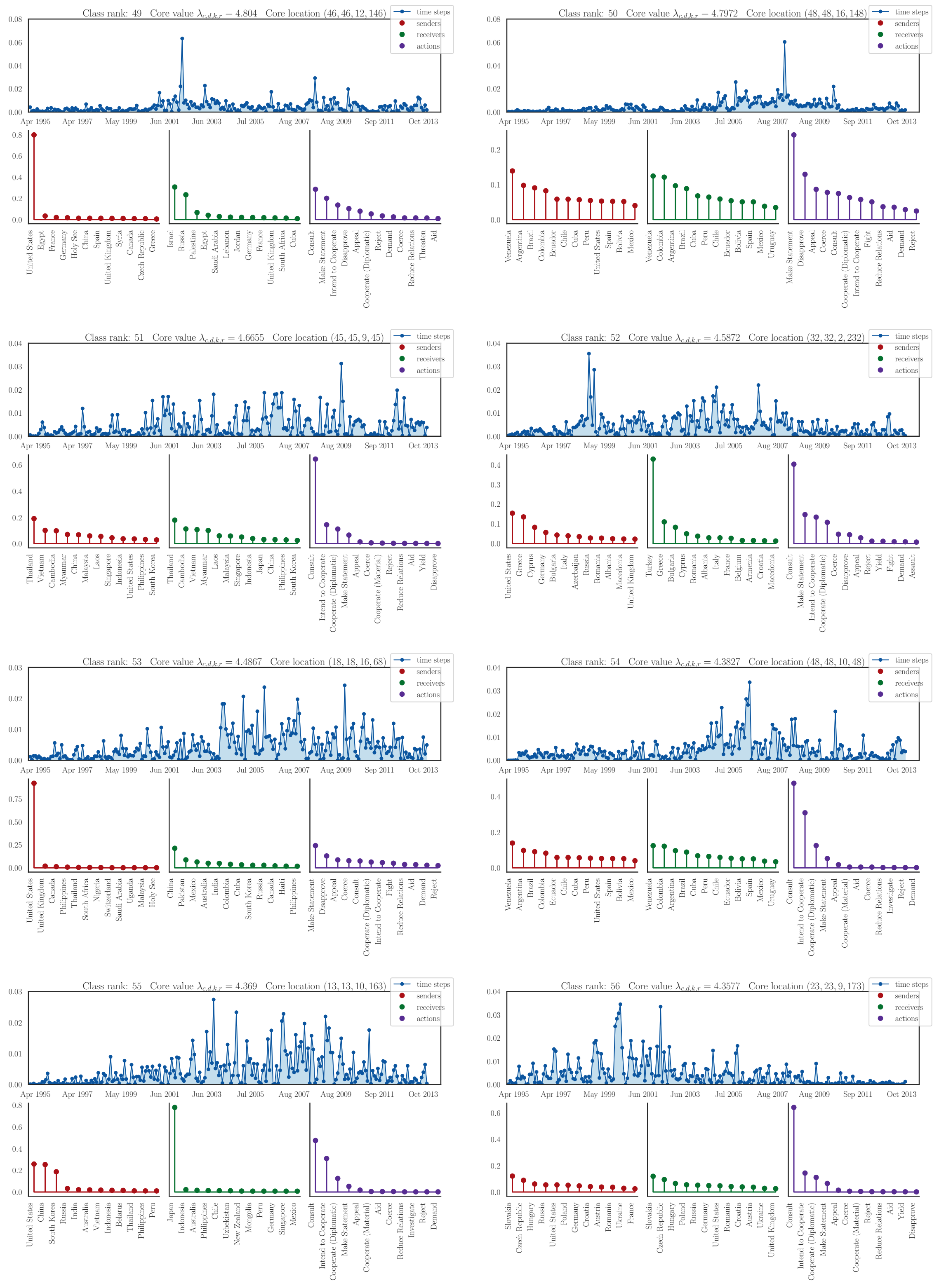


Figure 13: Inferred latent classes 49–56.

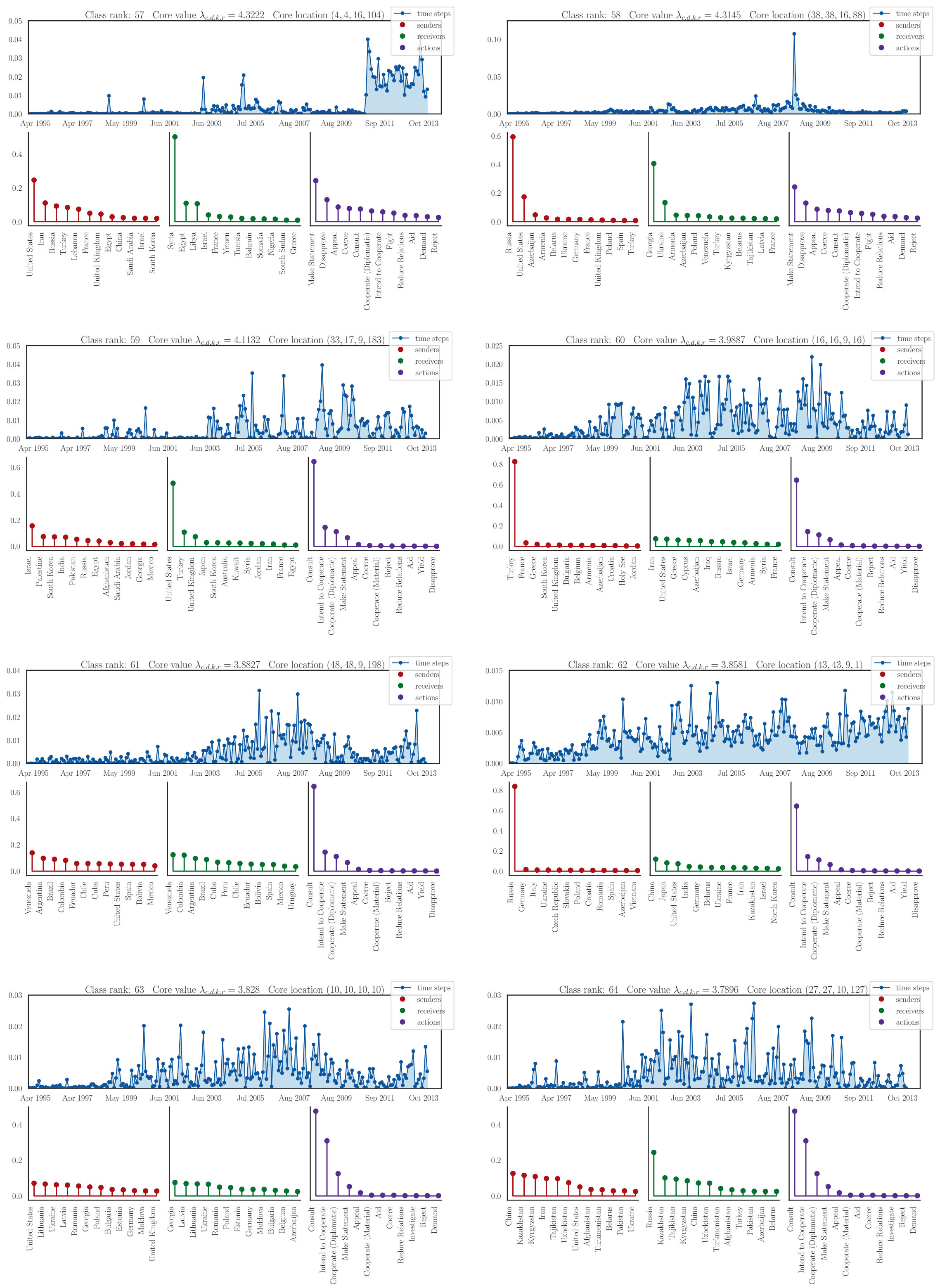


Figure 14: Inferred latent classes 57–64.

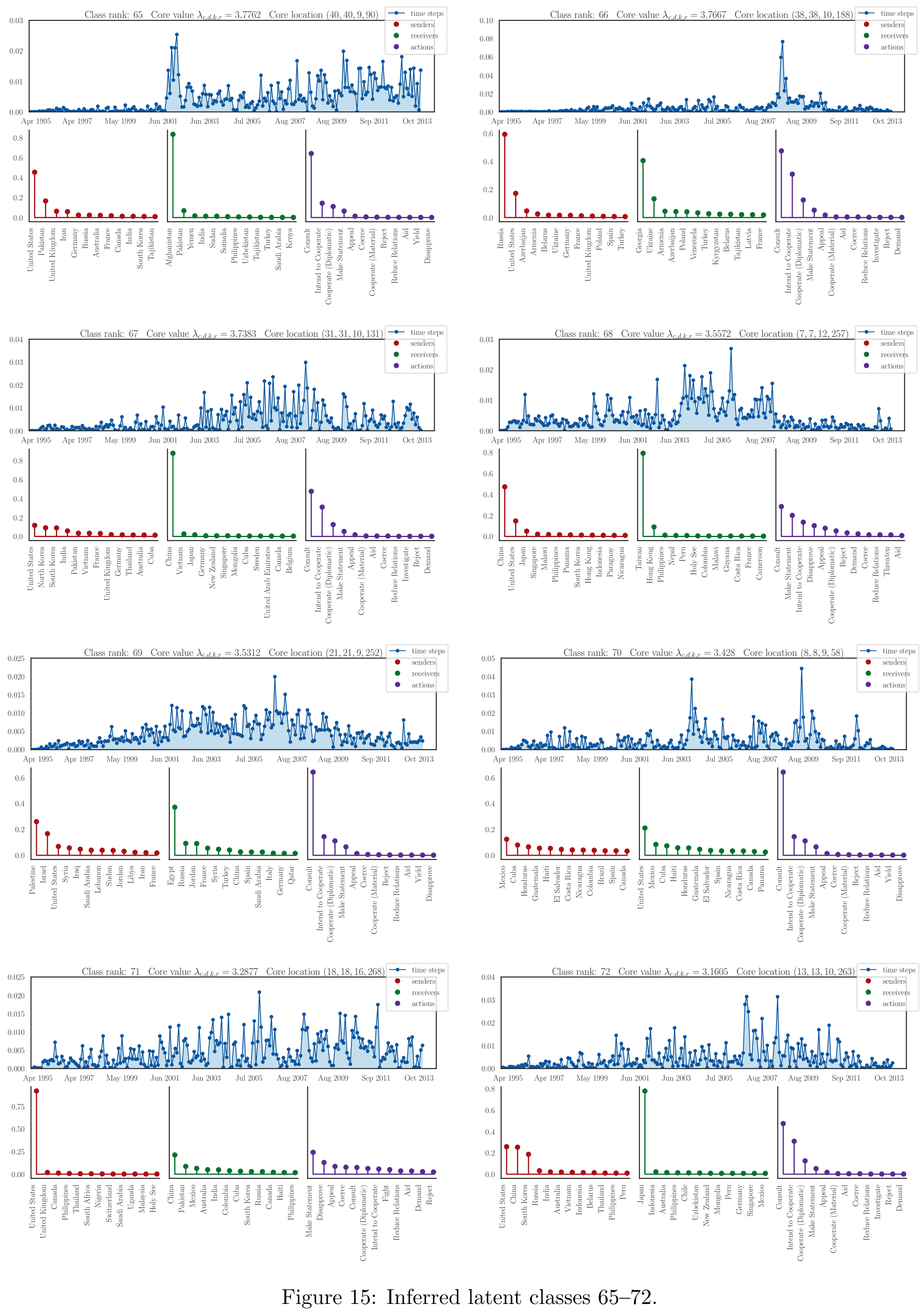


Figure 15: Inferred latent classes 65–72.

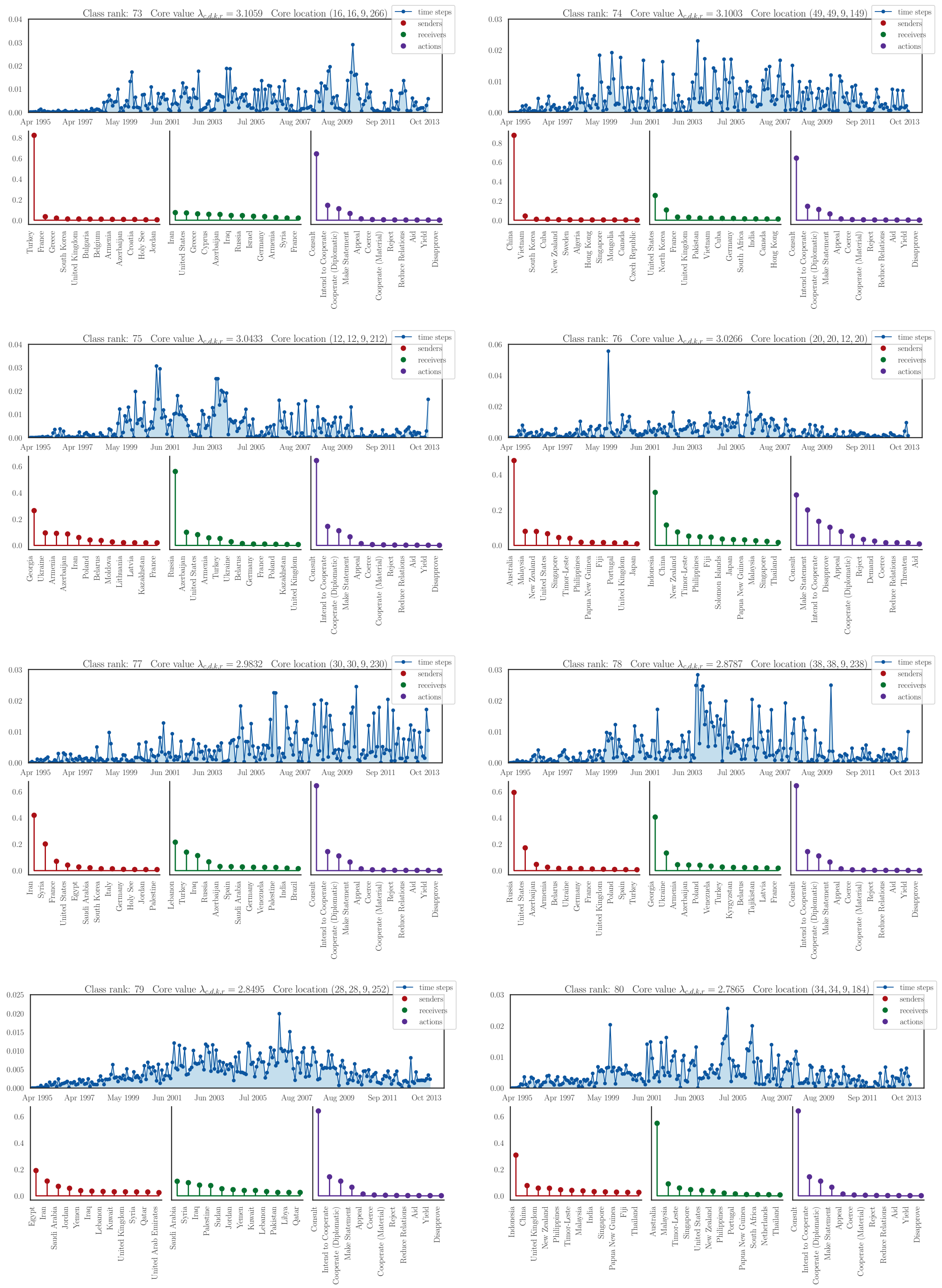


Figure 16: Inferred latent classes 73–80.

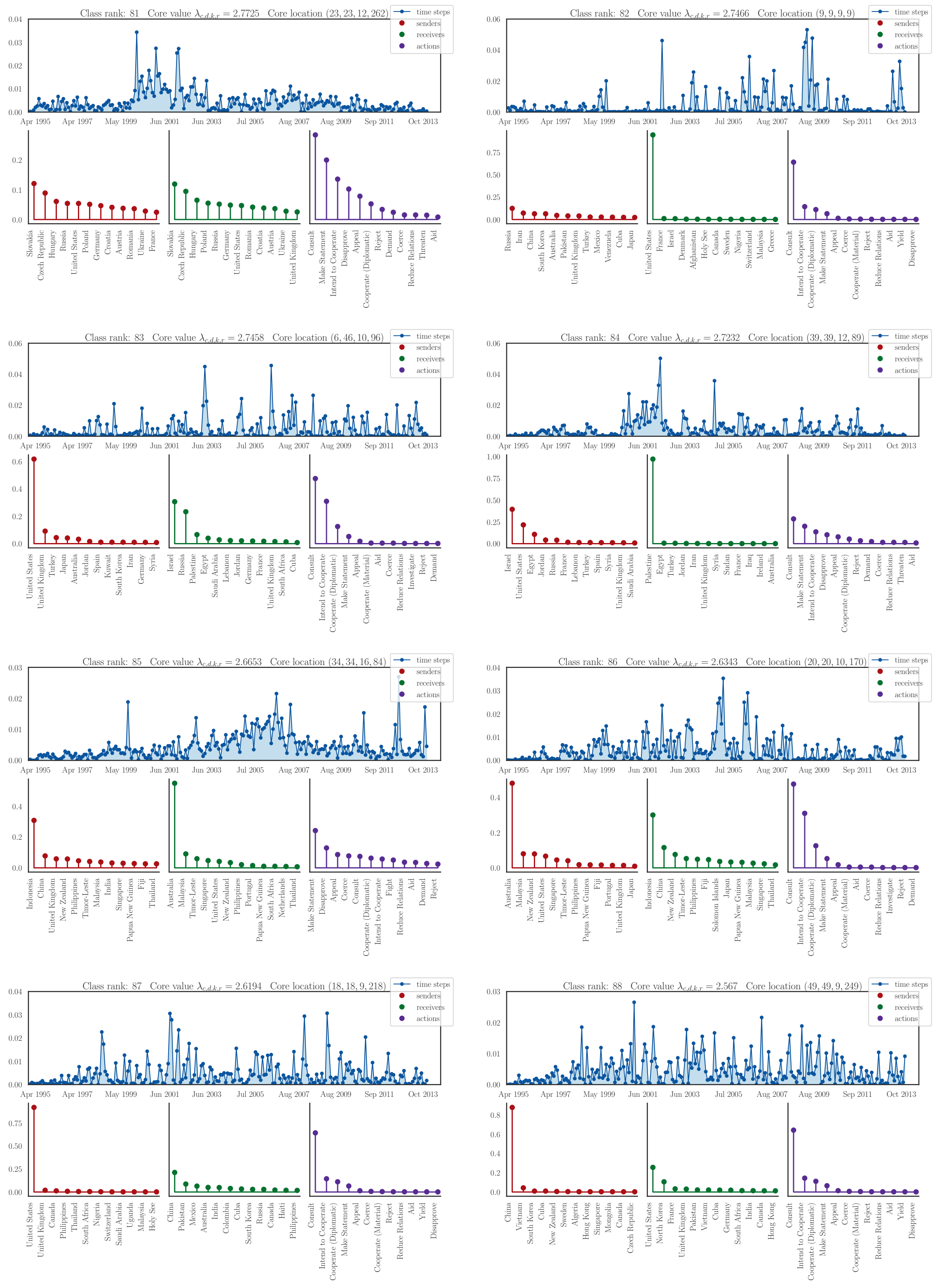


Figure 17: Inferred latent classes 81–88.

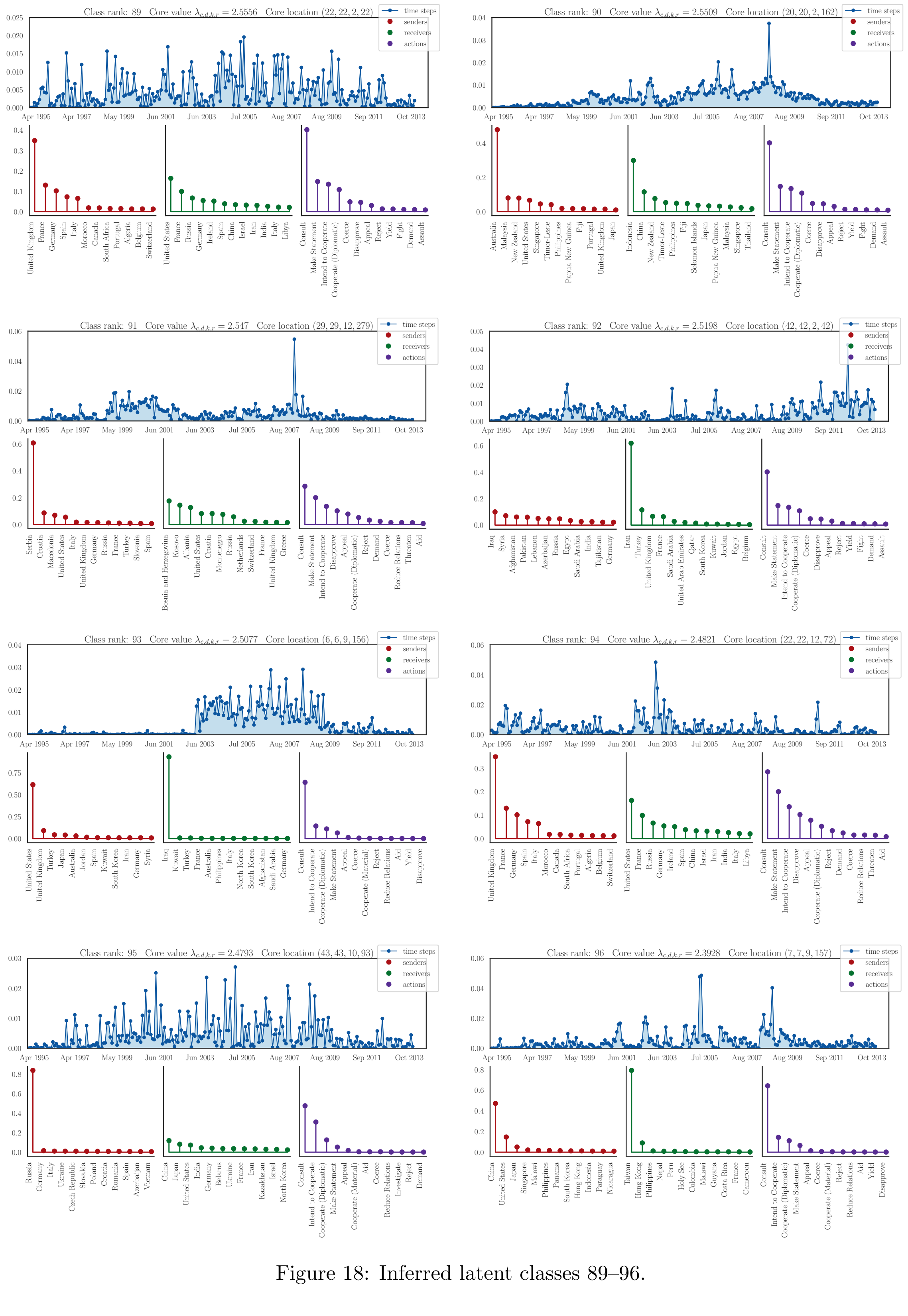


Figure 18: Inferred latent classes 89–96.

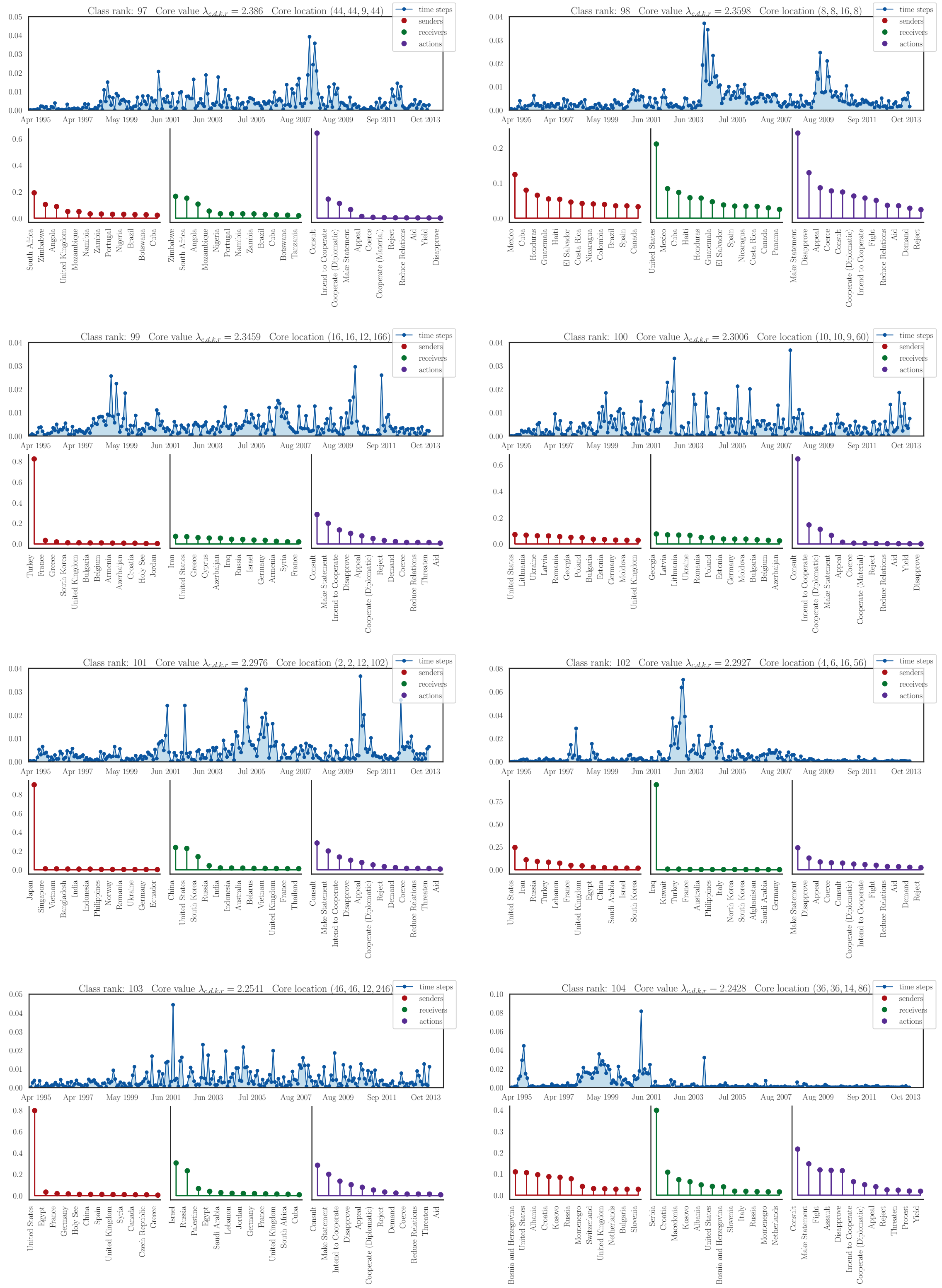


Figure 19: Inferred latent classes 97–104.