The Hyperedge Event Model

Bomin Kim*, Aaron Schein†, Bruce A. Desmarais‡, and Hanna Wallach§

Abstract. We introduce the hyperedge event model (HEM)—a generative model for events that can be represented as directed edges with one sender and one or more receivers or one receiver and one or more senders. We integrate a dynamic version of the exponential random graph model (ERGM) of edge structure with a survival model for event timing to jointly understand who interacts with whom, and when. The HEM offers three innovations with respect to the literature—first, it extends a growing class of dynamic network models to model hyperedges. The current state-of-the-art approach to dealing with hyperedges is to inappropriately break them into separate edges/events. Second, our model involves a novel receiver selection distribution that is based on established edge formation models, but assures non-empty receiver lists. Third, the HEM integrates separate, but interacting, equations governing edge formation and event timing. We use the HEM to analyze emails sent among department managers in Montgomery County government in North Carolina. Our application demonstrates that the model is effective at predicting and explaining time-stamped network data involving edges with multiple receivers. We present an out-of-sample prediction experiment to illustrate how researchers can select between different specifications of the model.

MSC 2010 subject classifications: Primary 60K35, 60K35; secondary 60K35.

Keywords: dynamic network model, hyperedge, continuous time model, email data analysis.

1 Introduction

Processes that arise as time-stamped directed interactions are common in the social, natural, and phyiscal sciences. The data produced by such processes can be represented as dynamic directed networks—an object that has given rise to the development of several statistical model families. For example, stochastic actor-oriented models (SAOMs) (**??**) characterize that network evolutions occur as the senders decide to create or remove an edge from the existing network, one edge at a time. Event-based network models (**????**) provide a general framework for modeling the realization of edges that occur as instances in continuous time streams of events. This family of models is flexible enough to account for the many ways in which past network structures beget future ones—e.g., if node i directed a tie to node j recently, then node j will direct one to node i in the near future—and useful for understanding the traits and behaviours that are predictive of interactions.

A major limitation of existing dynamic network models is that they apply to edges

*Department of Statistics, Pennsylvania State University bzk147@psu.edu
†College of Information and Computer Sciences, UMass Amherst aschein@cs.umass.edu
‡Department of Political Science, Pennsylvania State University bdesmarais@psu.edu
§Microsoft Research NYC hanna@dirichlet.net

DOI: 0000

with one sender and one receiver. Dynamic network data often naturally arise as "hyperedges" (**????**) that include one sender and multiple receivers or one receiver and multiple senders. For example, in email networks (**?**), each email encodes a hyperedge from one sender to one or more receivers. Networks formed between neurons via axons and dendrites involve hyperedges with one sender and multiple receivers (axons) or one receiver and multiple senders (dendrites) (**?**). Networks formed through the cosponsorship of legislative bills (**?**) involve hyperedges with multiple senders (cosponsors) and one receiver (sponsor). Economic sanctions between countries (**?**) induce networks with hyperedges between multiple sending countries and one target country. Existing models require researchers to alter hyperedge data to fit with the pairwise edge structure of the model. For instance, **?** treat multicast interactions—one type of directed hyperedge which involves one sender and one or more receivers—via duplication (i.e., obtain pairwise interactions from the original multicast), to construct an approximate likelihood function in their inferential framework for model parameters. Similarly, **?** duplicate emails sent from one sender to one or more receivers and randomly jitter the sent times in order to avoid violating the assumption that two events cannot occur at the exact same time.

We develop a statistical dynamic network model, which we term the hyperedge event model (HEM), that integrates the two components that govern hyperedge event formation: (1) the formation of the vertices that are incident to the hyperedge, and (2) the timing of the hyperedge event. In what follows, we define the HEM's generative process for hyperedge event data (Section 2), derive the conditional posteriors for Bayesian inference, and present tests of our software implementation (Section 3). We then demonstrate our model's applicability in a case study (Section 4) where we analyze a corpus of internal county government emails and illustrate how to perform model selection, posterior predictive checks, and exploratory analysis using our model. We conclude in Section 5.

2 The hyperedge event model

The hyperedge event model (HEM) specifies a generative process for E unique hyperedge events that occur between V number of nodes. A single hyperedge event is indexed by $e \in [E]$—where $[E]$ denotes a categorical set with E levels $[E] = \{1, \ldots, E\}$—and consists of three components: the sender $s_e \in [V]$, an indicator vector of receivers $\boldsymbol{r}_e$—where the element 1 indicates "receiving" the hyperedge event e and 0 otherwise—and the timestamp $t_e \in (0, \infty)$. For simplicity, we assume that events are ordered by time such that $t_e \leq t_{e+1}$. While the model can be applied to two types of hyperedge events—events involving (1) one sender and one or more receivers, and (2) one or more senders and one receiver—here we only present the generative process for those involving one sender and one or more receivers (i.e., multicast). One notable feature of our generative process is that we draw auxiliary variables that serve as candidate data. Data is generated from the HEM through a sampling process applied to the auxiliary variables. The auxiliary variables drawn for event e include, for each possible sender $i \in [V]$, a time increment from event $e-1$ at which sender i would create event e, and an $V-1$ length vector indicating which nodes would be the receivers of event e if it were directed by sender i.

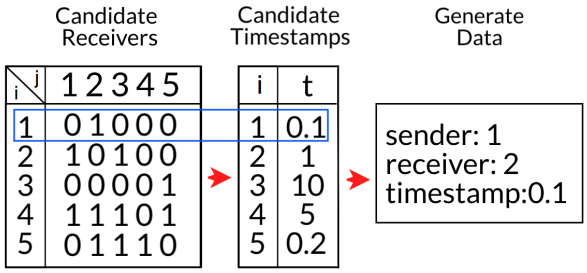


Figure 1: An illustrative example of the generative process of the HEM.

The data generated for event e under the HEM corresponds to the sender that would create event e the soonest—at the smallest time increment from event $e-1$. The receivers of event e generated under the HEM correspond to those receivers toward which the sender with the minimum time increment would have directed event e. We explain these steps in more detail below. Figure 1 presents an illustrative example on how the e^{th} event is generated, assuming $t_{e-1} = 0$ and $V = 5$. For hyperedge events that involve one receiver and one or more senders, we treat s_e to be an indicator vector of senders $\boldsymbol{s}_e$ and r_e to be the single receiver, and then follow the alternative generative process provided in Appendix A, which we derive as a simple reversal of the process used for multicasts (i.e., one sender and multiple receivers). Throughout the paper, we use the notation y_{iej} to imply "sender i directs an event e to receiver j," where y can be any parameter or covariate introduced in the HEM. When it is relevant to sender or receiver only, we use y_{ie} and y_{ej} to imply "sender i of an event e" and "receiver j of event e," respectively.

2.1 Candidate receivers

The HEM assumes that the sender of each hyperedge event is the sender that would initiate their respective event with the greatest urgency (i.e., the earliest timestamp). Our model thus assumes that for every event e, every possible sender i generates a candidate receiver set that would be the receiver set of event e if sender i were the sender. While **?** limits the possible receiver sets as combinations of nodes with the same cardinality as the observed receiver set, we take more realistic approach and allow all possible combination of nodes (except the no receiver case) to be considered. For an event e, we first define a series of vectors $\boldsymbol{u}_{ie}$ for $i = 1, \ldots, V$, where $\boldsymbol{u}_{ie}$ denotes sender i's receiver vector of zeros and 1's—i.e., 1's indicate the nodes to which i intends to direct event e. We then assume that each receiver vector $\boldsymbol{u}_{ie}$ comes from a modification of the multivariate Bernoulli (MB) distribution (**?**)—a model that has been used to model graphs in which the state of each edge indicator is drawn independently from an edge-specific Bernoulli distribution. Motivated by the Gibbs measure (**?**), we define a probability measure "MB_G" in order to avoid drawing hyperedge events with no receiver. This probability measure amounts to a non-empty Gibbs measure, in which the all-zero vector is excluded from the support of the multivariate Bernoulli distribution. As a result, this measure helps us to (1) allow a sender to select multiple receivers

for a single event, (2) force the sender to select at least one receiver, and (3) ensure a tractable normalizing constant for the receiver selection distribution. To summarize, for each node $i \in [V]$ we draw a binary vector $\boldsymbol{u}_{ie} = (u_{ie1}, \ldots, u_{ieV})$ using $\boldsymbol{u}_{ie} \sim \mathrm{MB}_G(\boldsymbol{\lambda}_{ie})$ which we define as

$$\Pr(\boldsymbol{u}_{ie} | \, \boldsymbol{b}, \boldsymbol{x}_{ie}) = \frac{1}{C(\boldsymbol{\lambda}_{ie})} \exp\Big(\log\big(\mathrm{I}(\|\boldsymbol{u}_{ie}\|_1 > 0)\big) + \sum_{j \neq i} \lambda_{iej} u_{iej}\Big), \tag{2.1}$$

where $C(\boldsymbol{\lambda}_{ie})$ is the normalizing constant with $\boldsymbol{\lambda}_{ie} = (\lambda_{ie1}, \ldots, \lambda_{ieV})$, $\|\cdot\|_1$ is the ℓ_1-norm, and the log-indicator term $\log\big(\mathrm{I}(\|\boldsymbol{u}_{ie}\|_1 > 0)\big)$ ensures that empty receiver sets are excluded from the distribution's support. These modeling assumptions facilitate efficient posterior inference since we can derive a closed form expression for the normalizing constant—i.e., $C(\boldsymbol{\lambda}_{ie}) = \prod_{j \neq i} \big(\exp(\lambda_{iej}) + 1\big) - 1$—and thus do not need to perform brute-force summation over the support of $\boldsymbol{u}_{ie} \in \{0, 1\}^V$. We provide detailed derivation steps for the normalizing constant $C(\boldsymbol{\lambda}_{ie})$ in Appendix B.

Here, we assume that λ_{iej} is the linear combination of statistics relevant to the receiver selection process for every possible sender–receiver pair (i, j) where $i \neq j$. In other words, we define

$$\lambda_{iej} = \boldsymbol{b}^\top \boldsymbol{x}_{iej}, \tag{2.2}$$

where $\boldsymbol{b}$ is a P-dimensional vector of coefficients with a Normal prior $\boldsymbol{b} \sim N(\boldsymbol{\mu}_b, \Sigma_b)$, and $\boldsymbol{x}_{iej}$ is a set of receiver selection features. As we show below, λ_{iej} contributes to the probability that i directs event e towards receiver j. The features $\boldsymbol{x}_{iej}$ can capture common network processes like popularity, reciprocity, and transitivity, as well as the effects of attributes of the sender and receivers (e.g., their gender), or attributes of sender–receiver pairs (e.g., whether the sender is a supervisor of the receiver's). Moreover, we allow the covariates could depend on the observed interaction data (e.g., how many times i and j interacted in the last 7 days), thus in such case the observed events are not conditionally independent given the model parameters.

2.2 Candidate timestamps

In modeling "when," we do not directly model the timestamp t_e. Instead, we assume that each sender's "time increment"—i.e., waiting time to next event since t_{e-1}—is drawn from a specific exponential family distribution. We assume that every sender $i \in [V]$ generates his/her own time increment from event $e - 1$ to event e, which we define as τ_{ie}. We further specify that the distribution of τ_{ie} has sender-specific mean μ_{ie} and common variance σ_τ^2. Following the generalized linear model (GLM) framework (**?**), we assume that sender i's time increment $\tau_{ie} > 0$ satisfy

$$\begin{aligned} \mathrm{E}(\tau_{ie}) &= \mu_{ie}, \\ \mathrm{Var}(\tau_{ie}) &= \sigma_\tau^2, \end{aligned} \tag{2.3}$$

where the timing rate μ_{ie} for sender i and event e is determined by the linear combination of statistics relevant to timing selection process, similar to Section 2.1. Specifically, we define

$$\mu_{ie} = g^{-1}(\boldsymbol{\eta}^\top \boldsymbol{z}_{ie}), \tag{2.4}$$

where $\boldsymbol{\eta}$ is a Q-dimensional vector of coefficients with a Normal prior $\boldsymbol{\eta} \sim N(\boldsymbol{\mu}_\eta, \Sigma_\eta)$, $\boldsymbol{z}_{ie}$ is a set of event timing features or covariates that could affect timestamps of events (including those depending on the observed interactions), and $g(\cdot)$ is the appropriate link function such as identity, log, or inverse. Possible choices of distribution include exponential, Weibull, gamma, and log-normal distributions, which are commonly used in time-to-event modeling (**??**). Based on the specific distribution, we may need another latent variable to account for the variance of time increments, such as the shape parameter for Weibull and gamma, or the variance parameter for log-normal, which is commonly used across $i \in [V]$ and $e \in [E]$. We use $f_\tau(\cdot; \mu, \sigma_\tau^2)$ and $F_\tau(\cdot; \mu, \sigma_\tau^2)$ to denote the probability density function (p.d.f) and cumulative density function (c.d.f), respectively, with mean μ and variance σ_τ^2.

2.3 Senders, receivers, and timestamps

After the HEM draws the candidate timestamp vector $\boldsymbol{\tau}_e = \{\tau_{1e}, \ldots, \tau_{Ve}\}$ at which the event would be created given the candidate sender and receiver combinations, finally our model assumes that the observed sender, receivers, and timestamp of hyperedge event e are generated by selecting the sender–receiver-set pair with the smallest time increment:

$$\begin{aligned} s_e &= \text{argmin}_i(\tau_{ie}), \\ \boldsymbol{r}_e &= \boldsymbol{u}_{s_e e}, \\ t_e &= t_{e-1} + \tau_{s_e e}. \end{aligned} \tag{2.5}$$

Therefore, the HEM assumes a sender-driven process—i.e., the receivers and timestamp of an event are jointly determined by the sender's urgency to direct the event to those selected receivers—which is similar to **?** in that each actor gets his/her own rate function to determine "who will make the next change in network." Note that our generative process allows for tied events. In the case of tied events—i.e., multiple senders draw exactly the same candidate timestamps—we assume that all events are generated and occur simultaneously. Algorithm 1 summarizes the entire generative process for hyperedge events with one sender and one or more receivers.

2.4 Basis in existing models

Though the HEM, based on our knowledge of the literature, is unique in modeling hyperedge events in continuous time, we build our model using features inspired by many of the existing models for dynamic network data. First, the features that govern which receivers are selected by a given sender are defined using subgraph configuration counts (e.g., the number of triangles formed through sending the tie, the number of previous hyperedges that would be reciprocated by the tie). The HEM's use of subgraph configuration formation follows a similar process underlying network change in the the continuous-time event model proposed by **?** as well as the relational event model (**?**), and is also related to the subgraph completion processes used to model discrete-time network change in the temporal ERGM (**?**). Second, the use of candidate receivers as a mechanism through which continuous network change is modeled is related to the use of

Algorithm 1 Generative process: one sender and one or more receivers

Input: number of events and nodes (E, V), covariates $(\boldsymbol{x}, \boldsymbol{z})$, and coefficients $(\boldsymbol{b}, \boldsymbol{\eta})$

for $e = 1$ to E **do**
  for $i = 1$ to V **do**
    for $j = 1$ to V $(j \neq i)$ **do**
      set $\lambda_{iej} = \boldsymbol{b}^\top \boldsymbol{x}_{iej}$
    end for
    draw $\boldsymbol{u}_{ie} \sim \mathrm{MB}_G(\boldsymbol{\lambda}_{ie})$
    set $\mu_{ie} = g^{-1}(\boldsymbol{\eta}^\top \boldsymbol{z}_{ie})$
    draw $\tau_{ie} \sim f_\tau(\mu_{ie}, \sigma_\tau^2)$
  end for
  if $n \geq 2$ tied events **then**
    set $s_e, \ldots, s_{e+n-1} = \mathrm{argmin}_i(\tau_{ie})$,
    set $\boldsymbol{r}_e = \boldsymbol{u}_{s_e e}, \ldots, \boldsymbol{r}_{e+n-1} = \boldsymbol{u}_{s_{e+n-1} e}$
    set $t_e, \ldots, t_{e+n-1} = t_{e-1} + \min_i \tau_{ie}$
    jump to $e = e + n$
  else
    set $s_e = \mathrm{argmin}_i(\tau_{ie})$
    set $\boldsymbol{r}_e = \boldsymbol{u}_{s_e e}$
    set $t_e = t_{e-1} + \min_i \tau_{ie}$
  end if
end for

latent changes between snapshot network observations in the stochastic actor oriented model (SAOM) by **?**. Third, also like the SAOM, the HEM includes separate sets of parameters that are used to model event timing and edge generation. Though unique in modeling hyperedges, the HEM is grounded in existing continuous-time models of networks, ERGM-style models, and the SAOM.

The models discussed above are those, like HEM, in which edge formation is driven by subgraph structure. There is another class of network models, originating with the latent space model introduced by **?**, in which tie formation is modeled by embedding nodes in a space of latent attributes. Under this modeling framework, each node is attributed with a coordinate in a metric space, and the likelihood of a tie between two nodes is inversely related to the distance between the two nodes in the metric space. The latent space model has seen several extensions, the latest of which tis the latent factor model proposed by **?**. A number of dynamic extensions of the latent space framework have been developed (e.g., **???**). Another popular form of latent variable model for networks is the stochastic block model (SBM) (**??**). In the SBM, each vertex is associated with one or more blocks, and each block is attributed with a vector of parameters that determines the rate at which members of the respective block form ties with members of each block. The blocks reduce vertex-to-vertex tie probabilities to block-to-block tie probabilities. Like the latent space model, the SBM has been extended to model discrete-time longitudinal networks (**?**). The HEM, as we have presented it,

does not involve latent node attributes, but in future work the HEM could be extended to incorporate latent node features—resulting in a latent space model and/or SBM for dynamic hyperedges.

3 Posterior inference

In this section we describe how we invert the generative process to obtain the posterior distribution over the latent variables—candidate receivers $\{\boldsymbol{u}_e\}_{e=1}^E$, coefficients for receiver selection features $\boldsymbol{b}$, and coefficients for event timing features $\boldsymbol{\eta}$—conditioned on the observed data $\{(s_e, \boldsymbol{r}_e, t_e)\}_{e=1}^E$, covariates $\{(\boldsymbol{x}_e, \boldsymbol{z}_e)\}_{e=1}^E$, and hyperparameters $(\boldsymbol{\mu}_b, \Sigma_b, \boldsymbol{\mu}_\eta, \Sigma_\eta)$. We draw the samples using Markov chain Monte Carlo (MCMC) methods, repeatedly resampling the value of each latent variable from its conditional posterior via a Metropolis-within-Gibbs sampling algorithm. In the next subsections, we provide each latent variable's conditional posterior along with pseudocode of MCMC in Algorithm 2 and some discussion on the computational complexity of our algorithm. We also evaluate the correctness of both our mathematical derivations and software implemenation using the prior–posterior simulator test of **?** in Appendix C.

3.1 Conditional posteriors

Candidate receivers

In our model, direct computation of the posterior densities for the latent variables $\boldsymbol{b}$ and $\boldsymbol{\eta}$—i.e., $\Pr(\boldsymbol{b}|\, \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{r}, \boldsymbol{t})$ and $\Pr(\boldsymbol{\eta}|\, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{r}, \boldsymbol{t})$—is not possible. However, it is possible to augment the data by candidate receivers $\boldsymbol{u}$ such that we can obtain their conditional posterior by conditioning on samples of $\boldsymbol{u}$. This approach—i.e., "data augmentation"—is commonly used throughout Bayesian statistics (**??**). Since u_{iej} is a binary random variable, it may be sampled from a Bernoulli distribution with probability $p_{iej} = \frac{\exp(\lambda_{iej})}{\exp(\lambda_{iej}) + \mathrm{I}(\|\boldsymbol{u}_{ie\setminus j}\|_1 > 0)}$, since

$$\begin{aligned} \Pr(u_{iej} = 1|\, \boldsymbol{u}_{ie\setminus j}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{r}, \boldsymbol{t}) &\propto \exp(\lambda_{iej}) \\ \Pr(u_{iej} = 0|\, \boldsymbol{u}_{ie\setminus j}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{r}, \boldsymbol{t}) &\propto \mathrm{I}(\|\boldsymbol{u}_{ie\setminus j}\|_1 > 0), \end{aligned} \tag{3.1}$$

where the subscript "$\setminus j$" denotes a quantity excluding data from position j and $\mathrm{I}(\cdot)$ is the indicator function that prevents empty receiver sets.

Coefficients for receiver selection features

Unlike the candidate receivers above, the conditional posterior density of $\boldsymbol{b}$ does not have a closed form; however $\boldsymbol{b}$ may instead be re-sampled using the Metropolis–Hastings (MH) algorithm. Assuming a generic Normal prior $\boldsymbol{b} \sim N(\boldsymbol{\mu}_b, \Sigma_b)$, the conditional posterior

distribution of $\boldsymbol{b}$ is proportional to

$$\Pr(\boldsymbol{b}|\,\boldsymbol{u},\boldsymbol{x},\boldsymbol{s},\boldsymbol{r},\boldsymbol{t}) \propto \phi(\boldsymbol{b};\boldsymbol{\mu}_b,\Sigma_b)\times\prod_{e=1}^{E}\prod_{i=1}^{V}\frac{1}{C(\boldsymbol{\lambda}_{ie})}\exp\Big(\log\big(\mathrm{I}(\|\boldsymbol{u}_{ie}\|_1>0)\big)+\sum_{j\neq i}\lambda_{iej}u_{iej}\Big), \tag{3.2}$$

where $\phi(\boldsymbol{b};\boldsymbol{\mu}_b,\Sigma_b)$ denotes the multivariate normal density of $\boldsymbol{b}$ with mean $\boldsymbol{\mu}_b$ and covariance matrix Σ_b. We re-sample the new value of $\boldsymbol{b}$ using the proposal distribution $\boldsymbol{b}^{new}\sim N(\boldsymbol{b},\delta_b\times I_P)$, where δ_b is the proposal variance parameter for $\boldsymbol{b}$, and make the accept/reject decision based on the evaluation of conditional posterior above.

Coefficients for event timing features

Likewise, we use the MH algorithm to update the latent variable $\boldsymbol{\eta}$. Assuming a Normal prior $\boldsymbol{\eta}\sim N(\boldsymbol{\mu}_\eta=\mathbf{0},\Sigma_\eta=2\times I_Q)$, the conditional posterior distribution of $\boldsymbol{\eta}$ for an untied event case is proportional to

$$\Pr(\boldsymbol{\eta}|\,\boldsymbol{u},\boldsymbol{z},\boldsymbol{s},\boldsymbol{r},\boldsymbol{t}) \propto \phi(\boldsymbol{\eta};\boldsymbol{\mu}_\eta,\Sigma_\eta)\times\prod_{e=1}^{E}\Big(f_\tau(\tau_e;\mu_{s_ee},\sigma_\tau^2)\times\prod_{i\neq s_e}\big(1-F_\tau(\tau_e;\mu_{ie},\sigma_\tau^2)\big)\Big), \tag{3.3}$$

where $f_\tau(\tau_e;\mu_{s_ee},\sigma_\tau^2)$ is the probability that the e^{th} observed time increment comes from the specified distribution $f_\tau(\cdot)$ with the observed sender's mean μ_{s_ee}, and $\prod_{i\neq s_e}\big(1-F_\tau(\tau_e;\mu_{ie},\sigma_\tau^2)\big)$ is the probability that the rest of (unobserved) senders for event e all draw time increments greater than τ_e. Again, we re-sample the new value of $\boldsymbol{\eta}$ using the proposal distribution $\boldsymbol{\eta}^{new}\sim N(\boldsymbol{\eta},\delta_\eta\times I_Q)$, where δ_η is the proposal variance parameter for $\boldsymbol{\eta}$. Moreover, under the existence of tied events, the conditional posterior distribution of $\boldsymbol{\eta}$ is written as proportional to

$$\begin{aligned}&\Pr(\boldsymbol{\eta}|\,\boldsymbol{u},\boldsymbol{z},\boldsymbol{s},\boldsymbol{r},\boldsymbol{t}) \propto \phi(\boldsymbol{\eta};\boldsymbol{\mu}_\eta,\Sigma_\eta)\\ &\times\prod_{m=1}^{M}\Big(\prod_{e:t_e=t_m^*}f_\tau(t_m^*-t_{m-1}^*;\mu_{s_ee},\sigma_\tau^2)\times\prod_{i\notin\{s_e\}_{e:t_e=t_m^*}}\big(1-F_\tau(t_m^*-t_{m-1}^*;\mu_{ie},\sigma_\tau^2)\big)\Big),\end{aligned} \tag{3.4}$$

where $t_1^*,\ldots,t_M^*$ are the unique timepoints across E events ($M\leq E$). If $M=E$ (i.e., no tied events), equation (3.4) reduces to equation (3.3). Note that when we have the latent variable to quantify the variance in time increments σ_τ^2 (based on the choice of timestamp distribution in Section 2.2), we also use equation (3.3) (or equation (3.4) in case there exist tied events) for the additional MH update—e.g., $\Pr(k|\,\boldsymbol{\eta},\boldsymbol{u},\boldsymbol{z},\boldsymbol{s},\boldsymbol{r},\boldsymbol{t})$ for Weibull, $\Pr(\theta|\,\boldsymbol{\eta},\boldsymbol{u},\boldsymbol{z},\boldsymbol{s},\boldsymbol{r},\boldsymbol{t})$ for gamma, and $\Pr(\sigma_\tau^2|\,\boldsymbol{\eta},\boldsymbol{u},\boldsymbol{z},\boldsymbol{s},\boldsymbol{r},\boldsymbol{t})$ for log-normal.

3.2 Computational complexity

The sampler uses a blocked Gibbs sampler for the u_{iej}. Given the constraint that $\sum_j u_{iej}>0$, this sampler may be rather inefficient in the case where there are many one-to-one interactions (as is the case in the application, where 83% of the interactions

Algorithm 2 MCMC algorithm

Input: number of outer and inner iterations (O, I_1, I_2) and initial values of $(\boldsymbol{u}, \boldsymbol{b}, \boldsymbol{\eta})$

for $o = 1$ to O **do**
  for $e = 1$ to E **do**
    for $i = 1$ to V **do**
      for $j = 1$ to V $(j \neq i)$ **do**
        update u_{iej} using Gibbs update —equation (3.1)
      end for
    end for
  end for
  for $n = 1$ to I_1 **do**
    update $\boldsymbol{b}$ using MH algorithm—equation (3.2)
  end for
  for $n = 1$ to I_2 **do**
    update $\boldsymbol{\eta}$ using MH algorithm—equation (3.3) or (3.4)
  end for
  if extra parameter for σ_τ^2 **then**
    update the variance parameter using MH algorithm—equation (3.3) or (3.4)
  end if
end for
summarize the results with the last chain of $\boldsymbol{b}$ and $\boldsymbol{\eta}$

are dyadic). In order to go from the state $u_{ie} = (1, 0, 0, 0, 0, 0)$ to $u_{ie} = (0, 1, 0, 0, 0, 0)$ one needs to go through a state where two receivers are activated, which has low probability in this case. It would be good to comment on this, and in general on the mixing of the MCMC sampler.

The authors should provide some indication of the computational complexity per iterations of their sampler. The application to email interaction data is rather small (18 nodes and 680 emails), so it would be good to know how many nodes/events the proposed approach can handle. Your quantitative assessments are based on 5 nodes in the simulation and 18 nodes in the application. These are quite small networks compared to those one would expect in realworld settings (such as those you list in the introduction). To what extent are your computational methods able to scale to much larger networks? An application to a bigger dataset would be useful. Moreover, more information on computational time should be provided.

Given that the model is relatively complex, some sensitivity analyses should be carried out to check how much posterior inference is affected by the hyperparameters' settings, and, possibly, suggest some default values.

There are many MH routines in the literature. Which type of MH do you consider? What is the proposal distribution? What about the acceptance rate? Is there any smart proposal in this case which helps in increasing the acceptance rate.

You rely on data augmentation MCMC, which has been shown to mix quite poorly (also in recent theoretical papers). Indeed, as expected, you end up thinning the chains every 40 samples in the application. However, there is no comment on this in the paper. I think it should be highlighted and not just hidden in the thinning.

4 Application to email data

We now present a case study applying our method to Montgomery county government email data. Our data come from the North Carolina county government email dataset collected by **?** that includes internal email corpora covering the inboxes and outboxes of managerial-level employees of North Carolina county governments. Out of over twenty counties, we chose Montgomery County to (1) test our model using data with a large proportion of hyperedges (16.76%), all of which are emails sent from one sender to two or more receivers, and (2) limit the scope of this initial application. The Montgomery County email network contains 680 emails, sent and received by 18 department managers over a period of 3 months (March–May) in 2012. For this case study, we formulate our model specification through definitions of the receiver selection features $\boldsymbol{x}$ and event timing features $\boldsymbol{z}$. We then report a suite of experiments—out-of-sample prediction for model selection and posterior predictive checks—that illustrate how alternative formulations of the HEM can be compared, and evaluate how well our model recovers the distribution of the observed data. Finally, we demonstrate an exploratory analysis of Montgomery County email data using the model estimates to discover substantively meaningful patterns in organizational communication networks.

4.1 Covariates

Receiver selection features

A primary purpose of any network model is to use the posterior distributions to learn which features predict and/or explain edge formation (e.g., is edge formation reciprocal, are edges more likely to be formed among nodes with the same gender). This email application specifically gives rise to the following question: "To what extent are nodal, dyadic or triadic network effects relevant to predicting future emails?" As an illustrative example, we form the receiver selection features $\boldsymbol{x}$ for Montgomery County email data using nodal, dyadic, and triadic covariates. First, by default we include a normally distributed intercept term to account for the average (or baseline) number of receivers. Next, as we want to test whether gender plays a role in receiver selection process, we include three nodal covariates—the gender information of sender and receiver, and their homophily indicator (i.e., an indicator of whether the sender and receiver are of the same gender). Additionally, we include four interval-based nodal network covariates—outdegree of sender (i.e., the number of emails sent), indegree of receiver (i.e., the number of emails received), hyperedge size of sender (i.e., the number of total receivers directed from the sender), and the interaction between (i.e., scalar product of) outdegree and hyperedge size—to study the effect of nodal behaviors on future interactions. For dyadic and triadic network effects, we employ the network statistics in

? and summarize past interaction behaviors based on the time interval prior to and including t_{e-1}. Specifically, our time interval tracks 7 days prior to the last email was sent $l_e = (t_{e-1} - 7 \text{ days}, t_{e-1}]$. For $i \in [V], j \in [V]$, and $e \in [E]$, we define 14 covariates for $\boldsymbol{x}_{iej}$:

1. intercept: $x_{iej1} = 1$;
2. gender_sender(i): $x_{iej2} = I(\text{gender of sender } i = \text{female})$;
3. gender_receiver(j): $x_{iej3} = I(\text{gender of receiver } j = \text{female})$;
4. gender_homophily(i, j): $x_{iej4} = I(x_{iej2} = x_{iej3})$;
5. outdegree(i): $x_{iej5} = \sum_{e':t_{e'} \in l_e} I(s_{e'} = i)$;
6. indegree(r): $x_{iej6} = \sum_{d':t_{e'} \in l_e} I(r_{e'j} = 1)$;
7. hyperedge_size(i): $x_{iej7} = \sum_{e':t_{e'} \in l_e} \sum_{j=1}^{V} I(s_{e'} = i)\, I(r_{e'j} = 1)$;
8. interaction(i): $x_{iej8} = x_{iej5} \times x_{iej7}$;
9. send(i, j): $x_{iej9} = \sum_{e':t_{e'} \in l_e} I(s_{e'} = i)\, I(r_{e'j} = 1)$;
10. receive(i, j): $x_{iej10} = \text{send}(j, i)$;
11. two_send(i, j): $x_{iej11} = \sum_{h \neq i,j} \text{send}(i, h)\, \text{send}(h, j)$;
12. two_receive(i, j): $x_{iej12} = \sum_{h \neq i,j} \text{send}(h, i)\, \text{send}(j, h)$;
13. sibling(i, j): $x_{iej13} = \sum_{h \neq i,j} \text{send}(h, i)\, \text{send}(h, j)$;
14. cosibling(i, j): $x_{iej14} = \sum_{h \neq i,j} \text{send}(i, h)\, \text{send}(j, h)$;

where $I(\cdot)$ is an indicator function. The network statistics (5–14) are designed so that their coefficients have a straightforward interpretation. The function "outdegree(i)" and "indegree(j)" measure the gregariousness and popularity effects of the node by counting the number of emails sent from i and received by j, respectively, within the last 7 days. The gregariousness effect refers to the tendency for nodes that created many events in the past to continue to do so in the future. The popularity effect refers to the tendency for nodes that were selected as receivers of many events in the past to continue to do so in the future. Moreover, in order to capture the individual tendency of senders to select two or more receivers, we include the statistic "hyperedge_size(i)"—the number of events directed from sender i within last 7 days where events with n number of receivers are counted as n separate events—as a variant of outdegree statistic, accounting for hyperedges. We also include the interaction term, "interaction(i)", between outdegree and hyperedge size. This interaction allows us to model a possible tradeoff between the hyperedge size and the total number of events created by i. Dyadic statistics "send(i, j)" and "receive(i, j)" are defined as above such that these covariates measure the number of events directed from i to j and j to i, respectively, within the last 7 days. In the

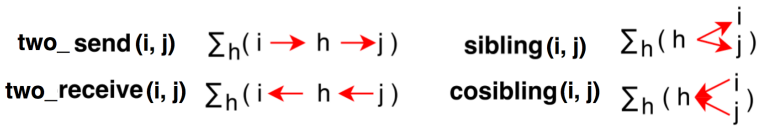


Figure 2: Visualization of triadic statistics: two_send, two_receive, sibling, and cosibling.

example of triadic statistics, the covariate "two_send(i, j)" counts the events involving some node h distinct from i and j such that events from i to h and h to j are both observed within the last 7 days. This statistic captures the tendency for events to close transitive triads (i.e., triads in which i directs to j and h, and j directs to h). We include other triadic covariates that behave similarly and exhibit analogous interpretations, which are illustrated in Figure 2.

Event timing features

For the event timing features $\boldsymbol{z}$, introduced in Section 2.2, we identify a set of covariates which may affect the time until the next event. Similar to the receiver selection features, we include nodal statistics which are time-invariant (such as gender or manager status) or time-dependent (such as the network statistics used for $\boldsymbol{x}$). In addition, we select some event-specific covariates based on the temporal aspect of the $(e-1)^{\text{th}}$ event—e.g., whether the previous email was sent (1) during the weekend and (2) before or past midday (AM/PM)—since we expect the email interactions within county government to be less active during the weekend and in the evening. To be specific, the timestamp statistics are defined as

1. intercept: $z_{ie1} = 1$;
2. gender(i): $z_{ie2} = I(\text{gender of sender } i = \text{female})$;
3. manager(i): $z_{ie3} = I(\text{sender } i \text{ is the County Manager})$;
4. outdegree(i): $z_{ie4} = \sum_{e':t_{e'} \in l_e} I(s_{e'} = i)$;
5. indegree(i): $z_{ie5} = \sum_{e':t_{e'} \in l_e} I(r_{e'i} = 1)$;
6. weekend(e): $z_{ie6} = I(t_{e-1} \text{ is during the weekend})$;
7. PM(e): $z_{ie7} = I(t_{e-1} \text{ in PM})$.

Note that our generative process for timestamps in Section 2.2 is sender-oriented where the sender determines when to send the email; thus we incorporate network statistics that depend only on i—specifically, the in and outdegrees of sender i.

4.2 Model selection

The HEM defines a flexible family of models, each of which is defined by a set of features (i.e., the receiver selection features $\boldsymbol{x}$, the selection of event timing features $\boldsymbol{z}$, and the distribution of time increments f). Many of these components will be specified based on user expertise (e.g., regarding which features would drive receiver selection), but some decisions may require a data-driven approach to model specification. For example, though theoretical considerations may inform the specification of features, subject-matter expertise is unlikely to inform the decision regarding the family of event timing distribution. Furthermore, since different distribution families (and model specifications more generally) may involve different size parameter spaces, any data-driven approach to model comparison must guard against over-fitting the data. In this section we present a general-purpose approach to evaluating the HEM specification using out-of-sample prediction. We illustrate this approach by comparing alternative distributional families for the event timing component of the model. Here, we specifically compare the predictive performance from two distributions—log-normal and exponential. We particularly choose the log-normal distribution based on some exploratory analysis (e.g., histogram and simple regressions) on raw time increments data, and select the exponential distribution as a baseline alternative that is a commonly specified distribution for time-to-event data, and is also used in the stochastic actor-oriented models (SAOMs) (**?**) as well as their extensions (**?**).

We evaluated the models' ability to predict out-of-sample events and timestamps on Montgomery County email data. We generated a train–test split of the data by randomly selecting 10% of senders, receivers, and timestamp variables to be held out. Our model then imputed these missing variables during inference, sampling them from their conditional posterior along with the other latent variables. Algorithm 3 outlines this procedure in detail. We compare the predictive performance of two versions of our model, each with a different timing distribution over the time increments, by repeating the predictive assessments$N = 500$ times where we make new predictions each time for the fixed held out data. We summarize the results of prediction experiments for missing senders, receivers, and timestamps in Figure 3. First, we compare the posterior probability of correct senders for each of the missing emails $\{e : s_e = \text{NA}\}$, which corresponds to $\pi_{s_e e}$ in Algorithm 3. We call this measure the "correct sender posterior probability." In Figure 3a, we display boxplots for the distribution of mean correct sender posterior probability—i.e., $\hat{\pi}_{s_e e} = \frac{1}{N}\sum_{n=1}^{N} \pi_{s_e e}^{(n)}$—across the missing emails. The results show that both log-normal and exponential distributions achieve better predictive performance for missing senders compared to what is expected under random guess (i.e., choose one out of V possible senders $= 1/18$), with the log-normal model showing better performance than the exponential model. Secondly, since the receiver vector is binary, we compute F_1 scores for missing receiver indicators (i.e., all e and j with r_{ej}=NA) by taking the harmonic mean of precision and recall:

$$\begin{aligned} F_1 &= 2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \text{ where} \\ \text{recall} = \frac{\text{TP}}{\text{TP+FN}} &\text{ and } \text{precision} = \frac{\text{TP}}{\text{TP+FP}}, \end{aligned} \tag{4.1}$$

Algorithm 3 Out-of-sample predictions

Input: data $\{(s_e, \boldsymbol{r}_e, t_e)\}_{e=1}^{E}$, number of new data to generate D, and initial values of $(\boldsymbol{b}, \boldsymbol{\eta}, \boldsymbol{u}, \sigma_\tau^2)$

Test splits:
draw test senders (out of E senders)
draw test receivers (out of $E \times (V-1)$ receiver indicators $\{\{r_{ej}\}_{j \in [V]_{\setminus s_e}}\}_{e=1}^{E}$)
draw test timestamps (out of E timestamps)
set the test data as "missing" (NA)

Imputation and inference:
for $d = 1$ **to** D **do**
  for $e = 1$ **to** E **do**
    if $s_e = \text{NA}$ **then**
      compute $\boldsymbol{\pi}_e = (\pi_{1e}, \ldots, \pi_{Ve})$, where
      $\pi_{ie} = \left(f_\tau(\tau_e; \mu_{ie}, \sigma_\tau^2) \times \prod_{i' \neq i} \left(1 - F_\tau(\tau_e; \mu_{i'e}, \sigma_\tau^2)\right) \right) \times$
      draw $s_e \sim \text{Categorical}(\boldsymbol{\pi}_e)$
    end if
    for $j \in [V]_{\setminus s_e}$ **do**
      if $r_{ej} = \text{NA}$ **then**
        draw $r_{ej} \sim \text{Bernoulli}(p_{ej})$, where $p_{ej} = \frac{\exp(\lambda_{iej})}{\exp(\lambda_{iej}) + \text{I}(\|\boldsymbol{u}_{ie\setminus j}\|_1 > 0)}$
      end if
    end for
    if $t_e = \text{NA}$ **then**
      draw τ_e from its conditional distribution using importance sampling, where
      $P(\tau_e | \cdot) \propto \left(f_\tau(\tau_e; \mu_{s_e e}, \sigma_\tau^2) \times \prod_{i \neq s_e} \left(1 - F_\tau(\tau_e; \mu_{ie}, \sigma_\tau^2)\right) \right) \times$
    end if
    run inference and update $(\boldsymbol{u}, \boldsymbol{b}, \boldsymbol{\eta})$ given the imputed and observed data
  end for
  store the estimates for test data
end for

with TP denoting true positive (i.e., $r_{ej}^{(\text{obs})} = r_{ej}^{(\text{pred})} = 1$), FN denoting false negative (i.e., $r_{ej}^{(\text{obs})} = 1$ but $r_{ej}^{(\text{pred})} = 0$), and FP denoting false positive (i.e., $r_{ej}^{(\text{obs})} = 0$ but $r_{ej}^{(\text{pred})} = 1$). Although the generative process for events (Section 2.1) is not directly affected by the choice of timestamp distribution, Figure 3b reveals slight difference between log-normal and exponential in their performance in predicting missing receiver indicators, where log-normal on average outperforms exponential. Finally, the prediction error for the d^{th} missing timestamp is estimated using the median of the absolute relative errors, often referred to as median absolute percentage error (MdAPE), across $N = 500$ predictions:

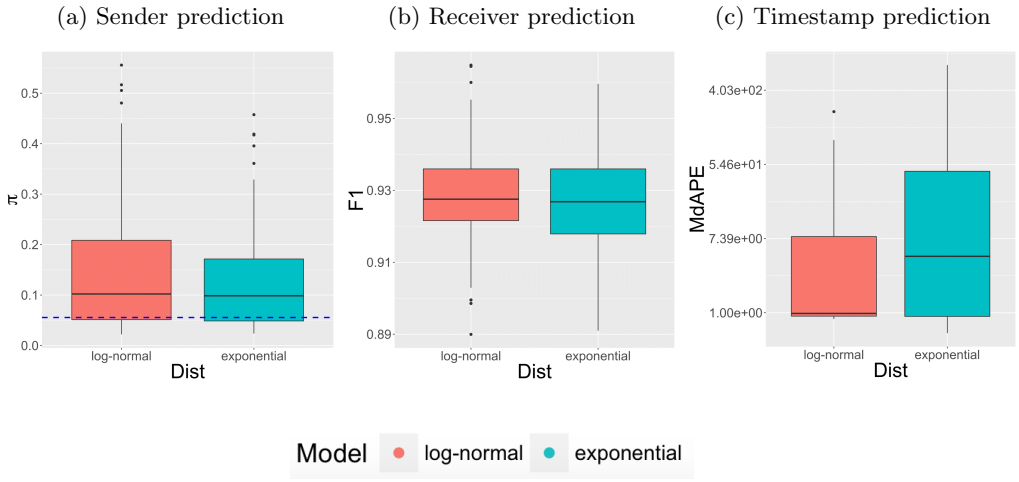


Figure 3: Comparison of predictive performance between log-normal and exponential distributions: (a) correct sender posterior probability from sender predictions, (b) F_1 scores from receiver predictions, and (c) median absolute relative error from timestamp predictions. Blue line in (a) represents the correct sender probability expected by random guess—i.e., $1/V = 1/18 \approx 0.056$.

$$\text{MdAPE}_e = \text{median}\left(\left\{\left|\frac{\tau_e^{(\text{obs})} - \tau_e^{(\text{pred})_1}}{\tau_e^{(\text{obs})}}\right|, \ldots, \left|\frac{\tau_e^{(\text{obs})} - \tau_e^{(\text{pred})_N}}{\tau_e^{(\text{obs})}}\right|\right\}\right). \tag{4.2}$$

Figure 3c presents boxplots for the median absolute percentage errors. These plots show that the log-normal distribution fits the time increments significantly better than the exponential distribution. We speculate that this difference can be simply explained by a lack of flexibility in the one-parameter exponential distribution. As illustrated above, we can use this out-of-sample prediction task for two uses—(1) to provide an effective answer to the question "how does the HEM perform at filling in the missing components of time-stamped network data?" and (2) to offer one standard way to determine the distribution of time increments in Section 2.2.

4.3 Posterior predictive checks

In this section, we perform posterior predictive checks (PPC) (**?**) to evaluate the appropriateness of our model specification for Montgomery County email data. We formally generated entirely new data by simulating $N = 500$ synthetic email datasets $\{(s_e, \boldsymbol{r}_e, t_e)\}_{e=1}^{E}$ from the generative process in Section 2, conditional upon a set of inferred latent variables from inference in Section 4.4. For the test of goodness-of-fit in terms of network dynamics, we use multiple statistics that summarize meaningful aspects of the data: (1) outdegree distribution and (2) indegree distribution, which is the number of emails sent by and received by each node respectively, (3) receiver size distri-

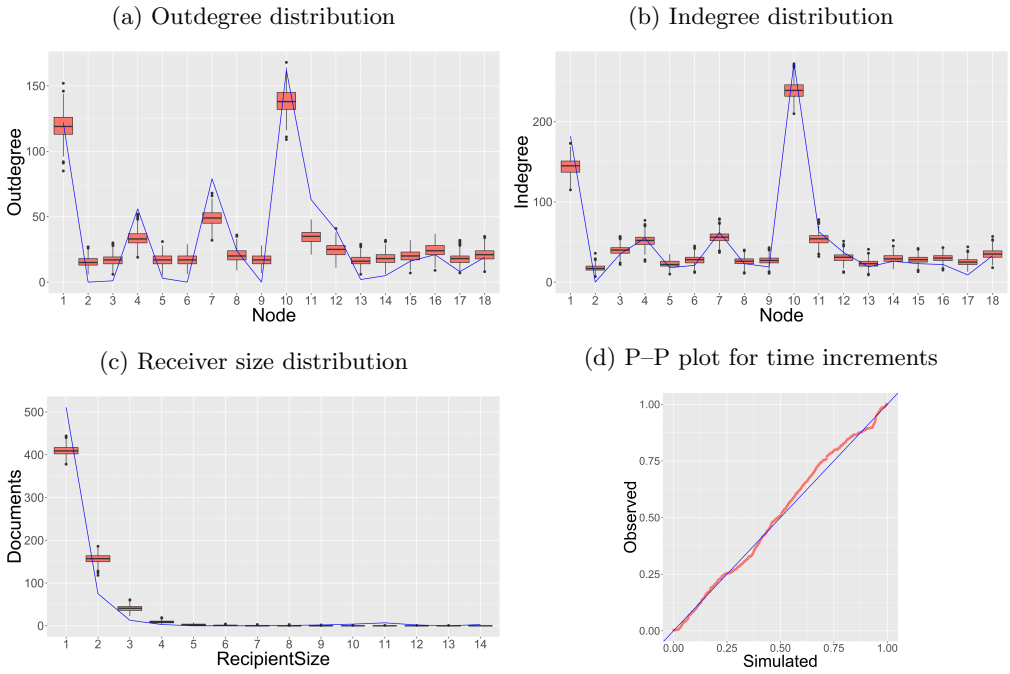


Figure 4: PPC results from log-normal distribution. Blue lines denote the observed statistics in (a)–(c) and denotes the diagonal line in (d).

bution (i.e., the number of receivers on each emails), and (4) a probability–probability (P–P) plot for time increments.

Figure 4 illustrates the results of posterior predictive checks using the log-normal model, which fits the timestamps better than the exponential model (Section 4.2). The upper two plots show node-specific posterior predictive degree distributions across $N = 500$ synthetic samples, where the left one is for the outdegree statistic and the right plot is for the indegree statistic. For both plots, the x-axis represents the nodes ($a = 1, \ldots, 18$), and the y-axis represents the number of emails sent or received by the node. When compared with the observed outdegree and indegree statistics (red lines), our model appears to fit the overall distribution of sending and receiving activities across the nodes. For example, node 1 and 10 have a significantly higher level of both sending and receiving activities relative to the rest and this is captured in the model-simulated data. The outdegree distribution of some low-activity nodes are not precisely recovered; however, the indegree distribution looks much better. Since we use more information in the receiver selection process (i.e., network effects) while we rely solely on minimum time increments when choosing the observed sender, these results are expected. The lower left plot is the distribution of receiver sizes, where the x-axis spans over the size of receivers 1 to 14 (which is the maximum size of observed receivers) and the y-axis denotes the number of emails with x-number of receivers. The result shows that our model is

underestimating emails with one receiver while overestimating emails with two, three, and four receivers. One explanation behind what we observe is that the model is trying to recover broadcast emails, which are the emails with ≥ 10 number of receivers, so that the intercept estimate b_1 is slightly moved toward right. It would be an interesting problem in future research to consider how the hyperedge size distribution can be further modified to capture this distribution more accurately. The plot on the lower right is the P–P plot for time increments, which depicts the two cumulative distribution functions—one for simulated time increments and another for observed time increments—against each other in order to assess how closely two data sets agree. Here, the closeness to the diagonal line connecting $(0, 0)$ and $(1, 1)$ gives a measure of difference between the simulated and observed time increments, and our P–P plot shows that we have great performance in reproducing the observed timing distribution. Our findings from the predictive experiments in Section 4.2 are further revealed in the PPC from exponential distribution, where the PPC plots comparing log-normal and exponential distributions are presented in Appendix D.

4.4 Exploratory analysis

Based on the prediction experiments in Section 4.2, we interpret the results from the HEM using the log-normal distribution, with emphasis on understanding the effects of receiver selection and event timing features defined in Section 4.1. We assume weakly informative priors for latent variables such as $\boldsymbol{b} \sim N(\boldsymbol{\mu}_b = \mathbf{0}, \Sigma_b = 2 \times I_P)$, $\boldsymbol{\eta} \sim N(\boldsymbol{\mu}_\eta = \mathbf{0}, \Sigma_\eta = 2 \times I_Q)$, and $\sigma_\tau^2 \sim$ inverse-Gamma$(a = 2, b = 1)$, and then run the MCMC (Algorithm 2) with $O = 55,000$ outer iterations and a burn-in of 15,000, where we thin by keeping every 40^{th} sample. While the inner iterations for σ_τ^2 is fixed as 1, we specify the inner iterations $I_1 = 20$ for $\boldsymbol{b}$ and $I_2 = 10$ for $\boldsymbol{\eta}$ to adjust for slower convergence rates. Convergence diagnostics including the traceplots and Geweke diagnostics (**?**) are provided in Appendix E.

Coefficients for receiver selection features

Figure 5 shows the boxplots summarizing posterior samples of $\boldsymbol{b}$, where Figure 5a displays the coefficients for nodal covariates and 5b displays the coefficients for dyadic and triadic covariates. Since we use the logit functional form

$$\text{logit}(\lambda_{iej}) = \log\left(\frac{\lambda_{iej}}{1-\lambda_{iej}}\right) = b_1 + b_2 x_{iej2} + \ldots + b_{14} x_{iej14},$$

and can interpret the $\boldsymbol{b}$ estimates in terms of odds ratios $\frac{\lambda_{iej}}{1-\lambda_{iej}} = \exp(b_1 + b_2 x_{iej2} + \ldots + b_{14} x_{iej14})$. First of all, we find the effects of nodal covariates "gender_sender(i)" and "gender_receiver(j)" are both nearly always negative in the posterior samples. The log odds that any other node will be added as a receiver of an email is approximately two times less if the sender is a woman. The posterior distribution of the statistic "outdegree(i)" is mostly negative, if node i sent n emails to anyone last week, then node i is approximately $\exp(-0.109 \times n) \approx (0.897)^n$ times less likely to send an email to j. However, this straightforward interpretation of the outdegree statistic only applies when

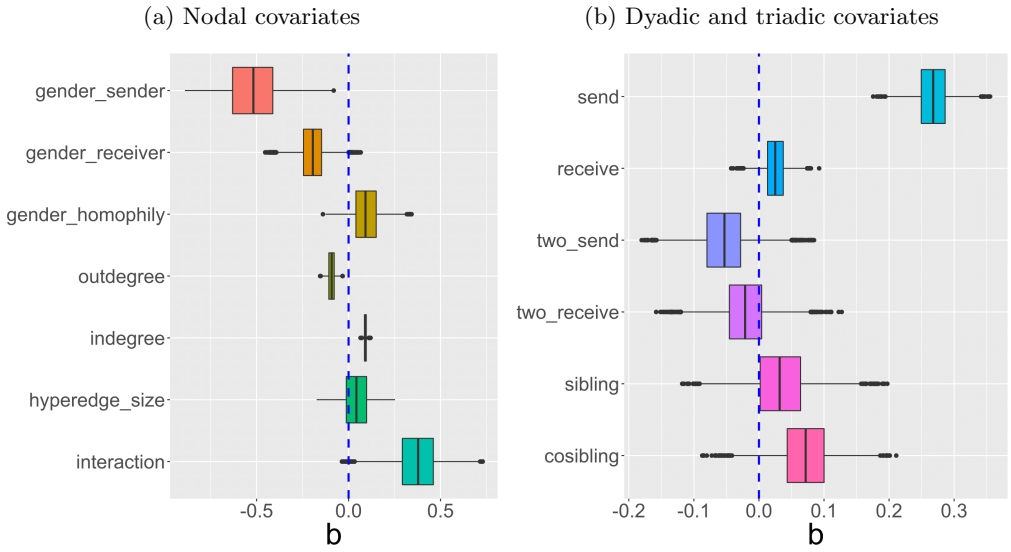


Figure 5: Posterior distribution of $\boldsymbol{b}$ estimates.

the hyperedge size is low. The scenario in which a node sends a lot of low-hyperedge-size emails may arise due to the use of email for a one-on-one conversation. The large positive estimates of the interaction between hyperedge size and outdegree indicate that those who have recently sent many emails with many receivers on each email are likely to continue sending broadcast emails. This scenario may arise from someone being responsible for distributing timely announcements. When we look at the effect of "indegree(j)," we see a clear popularity effect—those who have recently received a lot of emails are likely to continue receiving a lot of emails. If the receiver j received n emails over the last week, node i is $\exp(0.086 \times n) \approx (1.091)^n$ times more likely to send an email to j.

When we look at the effects of dyadic and triadic covariates, one thing that stands out is the large and positive posterior distribution of the statistic "send(i, j)" (i.e., number of times i sent emails to j over the last week) with the posterior mean $\hat{b}_9 = 0.274$, implying that if i sent n emails to j last week, then node i is approximately $\exp(0.274 \times n) \approx (1.315)^n$ times more likely to send an email to j. The posterior distributions for the reciprocity effect (i.e., "receive(i, j)"), and the four triadic effects, are all fairly evenly spread around zero, so our results do not justify conclusions regarding the nature of these effects in the Montgomery county email network.

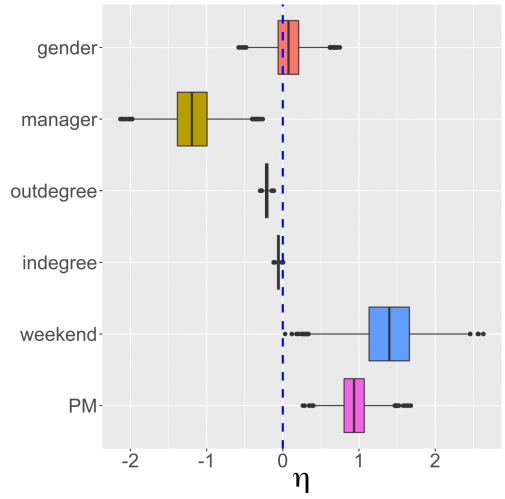


Figure 6: Posterior distribution of $\boldsymbol{\eta}$ estimates.

Coefficients for event timing features

For event timing features, Figure 6 shows the boxplots summarizing posterior samples of $\boldsymbol{\eta}$. Note that interpretations of the estimated coefficients for $\hat{\boldsymbol{\eta}}$ should be based on the specified time unit of the datset; we specify time units to be hours for the Montgomery county email data. Moreover, since we assume the log-normal distribution for time increments, the coefficients are interpreted in terms of the change in the average log time.

$$\log(\tau_{ie}) \sim N(\mu_{ie}, \sigma_\tau^2), \text{ with}$$
$$\mu_{ie} = \eta_1 + \eta_2 z_{ie2} + \ldots + \eta_7 z_{ie7}.$$

The posterior estimates of two temporal effects "weekend(e)" and "PM(e)" imply that if the $(e-1)^{\text{th}}$ email was sent during the weekend or after midday, then the time from $(e-1)^{\text{th}}$ email to e^{th} email is expected to take longer than what is expected under their counterparts (i.e., weekdays and am). Specifically, these effects increase the expected value of $\log(\tau_{ie})$ by $\hat{\eta}_6 = 1.552$ and $\hat{\eta}_7 = 0.980$, respectively. On the contrary, the covariates "manager(i)", "outdegree(i)", and "indegree(i)" shorten the amount of time until the next email. For example, being a county manager (i.e., the lead county administrator) lowers the expected value of $\log(\tau_{ie})$ by $\hat{\eta}_3 = -1.070$. The posterior mean estimates for the "outdegree(i)" and "indegree(i)" statistics are $\hat{\eta}_4 = -0.206$ and $\hat{\eta}_5 = -0.060$, respectively. These effects indicate that those who are involved in either sending or receiving a lot of emails recently are likely to send emails with greater speed. The posterior distribution for the effect of the gender of the manager is evenly spread around zero. In addition, the posterior mean estimates for the variance parameter σ_τ^2 in the log-normal distribution is approximately $\hat{\sigma}_\tau^2 = 14.093$ with its 95% credible interval $(12.709, 15.555)$, indicating that there exists large variability in the time increments of emails.

5 Conclusion

Motivated by a growing class of dynamic network models which deal with events recorded in continuous time, the hyperedge event model (HEM) can effectively learn the underlying dynamics in events and their corresponding timestamp formations, providing novel insights to the literature. The HEM explicitly models hyperedges through a receiver selection distribution that forces the sender to select at least one receiver; this obviates the need to preprocess hyperedge data—e.g., by "duplicating" hyperedges—to match the assumptions of traditional network models. Our model treating them as pure duplicates. In modeling the timestamps (more precisely time increments) of events, our generalized linear model (GLM) based formulation offers new innovations by eliminating the need to stick with one parameter distribution (e.g., exponential distribution). To our knowledge, the HEM is the only existing model that can be used to generate the sender, receivers, and timestamp of interactions in real time. To make better use of the proposed model, we provide an algorithm for predictive experiments that helps to learn which specification of HEM provides a better fit to the data.

We have demonstrated the effectiveness of our model by analyzing the Montgomery County government emails, where emails serve as a canonical example of directed hyperedge events with one sender and one or more receivers. The estimated effects for receiver selection features reveal that our model is able to understand the structural dynamics similar to those used in the exponential random graph model (ERGM). Our model also learns the effects of event timing features by integrating a survival model for event timing. Although we illustrate the entire framework and application in the context of one type of hyperedge, one sender and one or more receivers, our model can be easily extended to allow the opposite case, one or more sender and one receiver, by slight modification of the generative process (shown in Appendix A). This extension involves promising applications to socio-political networks such as international sanctions and co-sponsorship of bills, and biological networks such as those formed through neural dendrites.

Acknowledgments

This work was supported in part by the University of Massachusetts Amherst Center for Intelligent Information Retrieval and in part by National Science Foundation grants DGE-1144860, SES-1619644, and CISE-1320219. Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect those of the sponsors.

Appendix

Appendix A: Alternative generative process

Algorithm 4 Generative process: one receiver and one or more senders

Input: number of events and nodes (E, V), covariates $(\boldsymbol{x}, \boldsymbol{z})$, and coefficients $(\boldsymbol{b}, \boldsymbol{\eta})$

for $e = 1$ to E **do**
  for $j = 1$ to V **do**
    for $i = 1$ to V $(i \neq j)$ **do**
      set $\lambda_{iej} = \boldsymbol{b}^\top \boldsymbol{x}_{iej}$
    end for
    draw $\boldsymbol{u}_{je} \sim \text{MB}_G(\boldsymbol{\lambda}_{je})$
    set $\mu_{je} = g^{-1}(\boldsymbol{\eta}^\top \boldsymbol{z}_{je})$
    draw $\tau_{je} \sim f_\tau(\mu_{je}, \sigma^2_\tau)$
  end for
  if $n \geq 2$ tied events **then**
    set $r_e, \ldots, r_{e+n-1} = \text{argmin}_j(\tau_{je})$
    set $\boldsymbol{s}_e = \boldsymbol{u}_{r_e e}, \ldots, \boldsymbol{s}_{e+n-1} = \boldsymbol{u}_{r_{e+n-1} e}$
    set $t_e, \ldots, t_{e+n-1} = t_{e-1} + \min_j \tau_{je}$
    jump to $e = e + n$
  else
    set $r_e = \text{argmin}_j(\tau_{je})$
    set $\boldsymbol{s}_e = \boldsymbol{u}_{r_e e}$
    set $t_e = t_{e-1} + \min_j \tau_{je}$
  end if
end for

Appendix B: Normalizing constant of MB_G

Our probability measure "MB_G"—the multivariate Bernoulli distribution with non-empty Gibbs measure—defines the probability of sender i selecting the binary receiver vector $\boldsymbol{u}_{ie}$ as

$$\Pr(\boldsymbol{u}_{ie} | \, \boldsymbol{b}, \boldsymbol{x}_{ie}) = \frac{1}{C(\boldsymbol{\lambda}_{ie})} \exp\Big(\log\big(\text{I}(\|\boldsymbol{u}_{ie}\|_1 > 0)\big) + \sum_{j \neq i} \lambda_{iej} u_{iej}\Big),$$

where λ_{iej} is a linear combination of receiver selection features—i.e., $\lambda_{iej} = \boldsymbol{b}^\top \boldsymbol{x}_{iej}$—as defined in Secton 2.1.

To use this distribution efficiently, we derive a closed-form expression for $C(\boldsymbol{\lambda}_{ie})$ that does not require brute-force summation over the support of $\boldsymbol{u}_{ie}$ (i.e., $\forall \boldsymbol{u}_{ie} \in [0,1]^V$). We recognize that if $\boldsymbol{u}_{ie}$ were drawn via independent Bernoulli distributions in which $\Pr(u_{iej} = 1 | \, \boldsymbol{b}, \boldsymbol{x}_{ie})$ was given by $\text{logit}(\lambda_{iej})$, then

$$\Pr(\boldsymbol{u}_{ie} | \, \boldsymbol{b}, \boldsymbol{x}_{ie}) \propto \exp\Big(\sum_{j \neq i} \lambda_{iej} u_{iej}\Big).$$

This is straightforward to verify by looking at

$$\Pr(u_{iej} = 1 | \, \boldsymbol{u}_{ie\setminus j}, \boldsymbol{b}, \boldsymbol{x}_{ie}) = \frac{\exp(\lambda_{iej})}{\exp(\lambda_{iej}) + 1},$$

where the subscript "$\setminus j$" denotes a quantity excluding data from position j. Now we denote the logistic-Bernoulli normalizing constant as $C^l(\boldsymbol{\lambda}_{ie})$, which is defined as

$$C^l(\boldsymbol{\lambda}_{ie}) = \sum_{\boldsymbol{u}_{ie} \in [0,1]^V} \exp\Big(\sum_{j \neq i} \lambda_{iej} u_{iej}\Big).$$

Now, since

$$\exp\Big(\log\Big(\mathrm{I}(\|\boldsymbol{u}_{ie}\|_1 > 0)\Big) + \sum_{j \neq i} \lambda_{iej} u_{iej}\Big) = \exp\Big(\sum_{j \neq i} \lambda_{iej} u_{iej}\Big),$$

except when $\|\boldsymbol{u}_{ie}\|_1 = 0$, we note that

$$\begin{aligned} C(\boldsymbol{\lambda}_{ie}) &= C^l(\boldsymbol{\lambda}_{ie}) - \exp\Big(\sum_{\forall u_{iej} = 0} \lambda_{iej} u_{iej}\Big) \\ &= C^l(\boldsymbol{\lambda}_{ie}) - 1. \end{aligned}$$

We can therefore derive a closed form expression for $C(\boldsymbol{\lambda}_{ie})$ via a closed form expression for $C^l(\boldsymbol{\lambda}_{ie})$. This can be done by looking at the probability of the zero vector under the logistic-Bernoulli model:

$$\frac{1}{C^l(\boldsymbol{\lambda}_{ie})} \exp\Big(\sum_{\forall u_{iej} = 0} \lambda_{iej} u_{iej}\Big) = \prod_{j \neq i} \Big(1 - \frac{\exp(\lambda_{iej})}{\exp(\lambda_{iej}) + 1}\Big).$$

Then, we have

$$\frac{1}{C^l(\boldsymbol{\lambda}_{ie})} \quad = \prod_{j \neq i} \frac{1}{\exp(\lambda_{iej}) + 1}.$$

Finally, the closed form expression for the normalizing constant is

$$C(\boldsymbol{\lambda}_{ie}) = \prod_{j \neq i} \big(\exp(\lambda_{iej}) + 1\big) - 1.$$

Appendix C: Getting it Right (GiR) test

Software development is integral to the objective of applying our model to real world data. Code review is a valuable process in any research computing context, and the prevalence of software bugs in statistical software is well documented (e.g., **??**). With highly complex models such as the HEM, there are many ways in which software bugs can be introduced and go unnoticed. As such, we present a joint analysis of the integrity of our generative model, sampling equations, and software implementation.

? introduced the "Getting it Right" (GiR) test—a joint distribution test of posterior simulators which can detect errors in sampling equations as well as software bugs—and it has been used to test the implementation of Bayesian inference algorithms (**?**). The test involves comparing the distributions of variables simulated from two joint distribution samplers, which we call "forward" and "backward" samplers. The "forward" sampler draws joint samples of the latent and observable variables from the prior. The "backward" sampler begins by first drawing a joint sample of the latent and observed variables from the prior. It then alternates between re-sampling the latent variables, conditioned on the observable variables, from the MCMC transition operator, and then re-sampling the observable variable, conditioned on the latent variables, from the model likelihood. If the MCMC transition operator is correctly derived and implemented, this process should asymptotically generate joint samples of the latent and observable variables from the prior, like the forward sampler.

In the forward sampler, both observable and unobservable variables are generated using Algorithm 1. In the backward samples, unobservable variables are generated using the sampling equations for inference, which we derived in Section 3.1. For each forward and backward sample that consists of E number of events, we save these statistics:

1. Mean of observed receiver sizes $\|\boldsymbol{r}_e\|_1$ across $e = 1, \ldots, E$,
2. Variance of observed receiver sizes $\|\boldsymbol{r}_e\|_1$ across $e = 1, \ldots, E$,
3. Mean of time increments τ_e across $e = 1, ..., E$,
4. Variance of time increments τ_e across $e = 1, ..., E$,
5. b_p value used to generate the samples $p = 1, ..., P$,
6. η_q value used to generate the samples $q = 1, ..., Q$,
7. σ_τ^2 value used to generate the samples in log-normal distribution

To keep the computational burden of repeated inference manageable, we run the GiR using a relatively small artificial sample, consisting of $E = 100$ events, $V = 5$ nodes, $P = 4$ number of receiver selection features, and $Q = 3$ number of event timing features per each forward or backward sampler, using log-normal distibution for the time increments f_τ. We generated 10^5 sets of forward and backward samples, and then calculated 1,000 quantiles for each of the statistics. We also calculated t-test and Mann-Whitney test p-values in order to test for differences in the distributions generated in the forward and backward samples. Before we calculated these statistics, we thinned our samples by taking every 9th sample starting at the 10,000th sample for a resulting sample size of 10,000, in order to reduce the autocorrelation in the Markov chains. In each case, if we observe a large p-value, this gives us evidence that the distributions generated under forward and backward sampling have the same locations. We depict the GiR results using probability–probability (P–P) plots, in which the empirical CDF values of the forward and backward samples are plotted on the x and y axes, respectively.

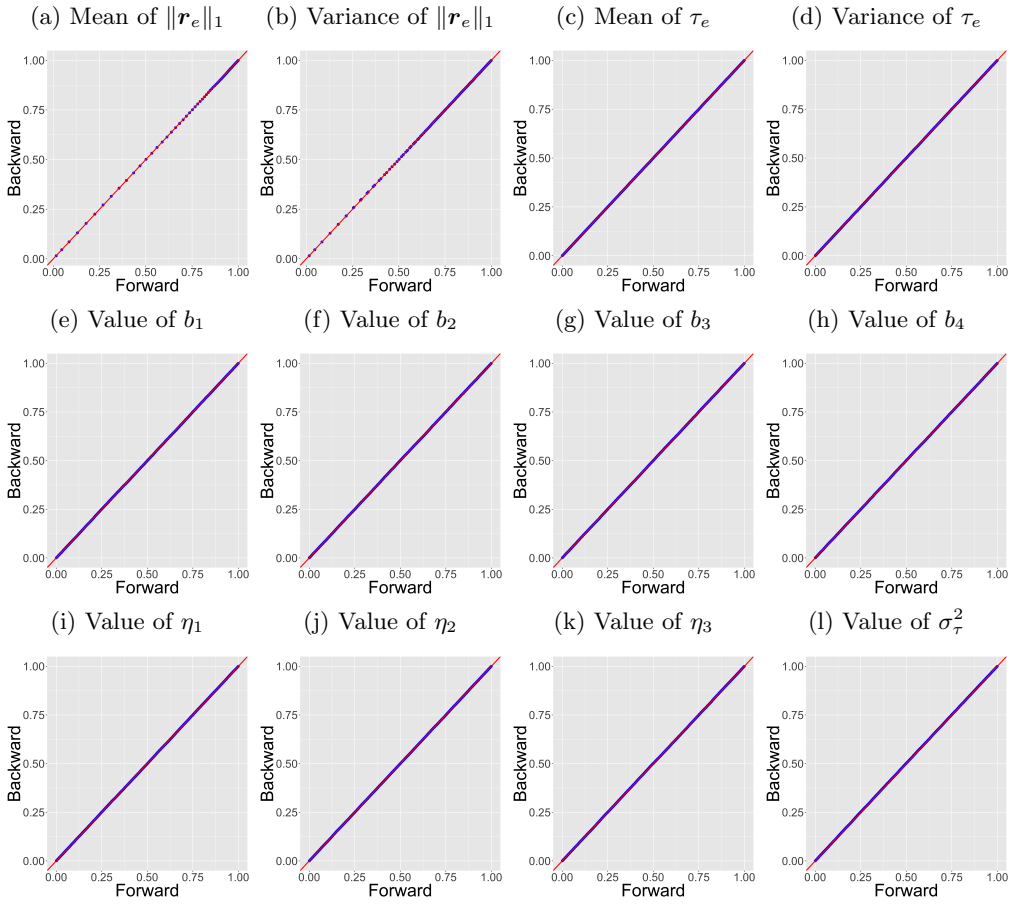


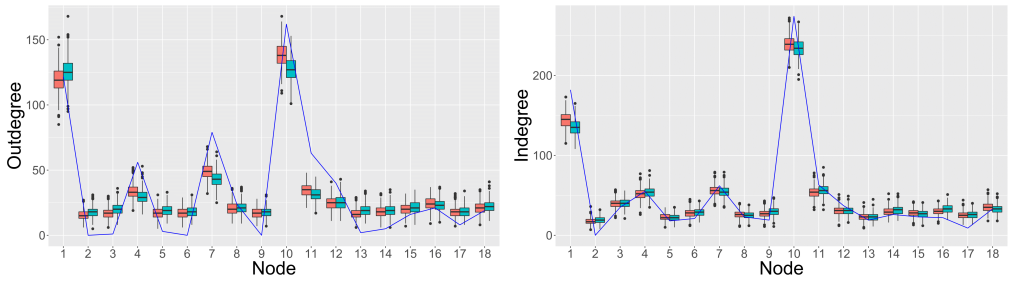
Figure 7: Probability–probability (P–P) plots for the GiR test statistics.

If the two samples are from equivalent distributions, the empirical CDF values should line up on a line with zero y-intercept, and unit slope (i.e., a 45-degree line). The GiR test results are depicted in Figure 7. These results indicate that our sampling equations and software implementation pass the test on every statistic.

Appendix D: Comparison of PPC results: log-normal vs. exponential

(a) Outdegree distribution

(b) Indegree distribution



(c) Receiver size distribution

(d) P–P plot for time increments

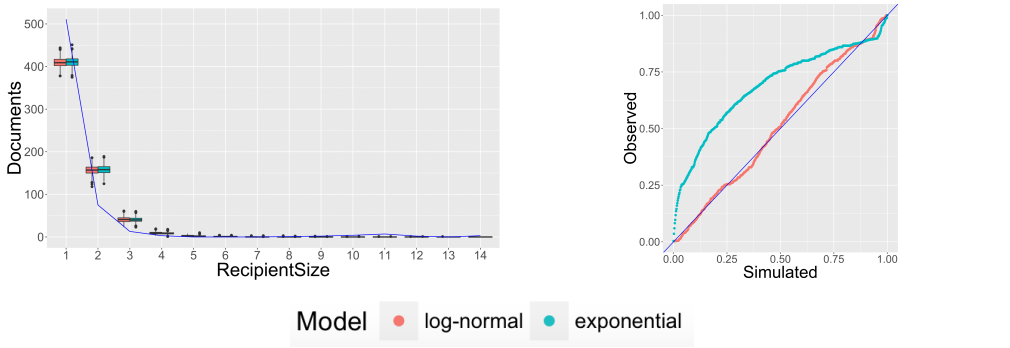
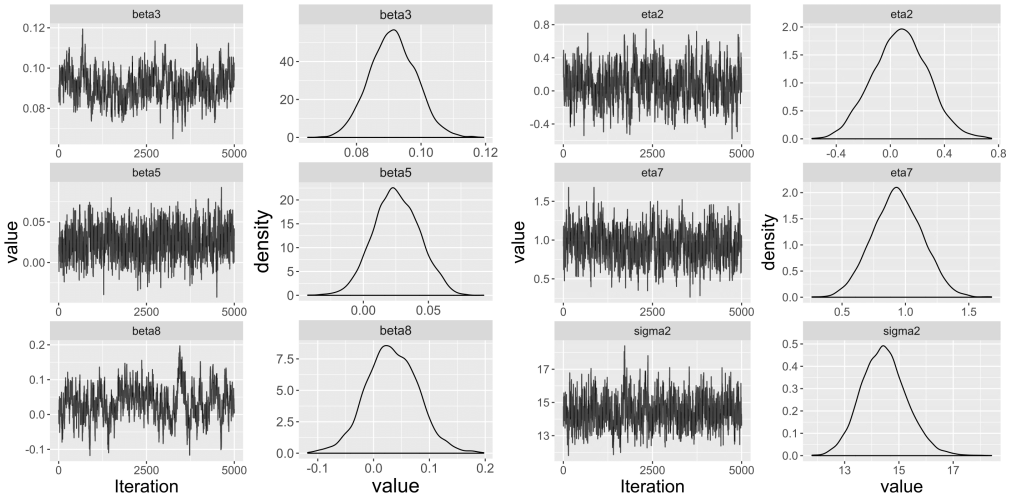


Figure 8: Comparison of PPC results between log-normal (*red*) and exponential (*green*) distributions. Blue lines denote the observed statistics in (a)–(c) and denotes the diagonal line in (d).

Appendix E: Convergence diagnostics

(a) Traceplots of $\boldsymbol{b}$ (b) Traceplot of $\boldsymbol{\eta}$



(c) Geweke diagnostics for $\boldsymbol{b}$ (d) Geweke diagnostics for $\boldsymbol{\eta}$ and σ_τ^2

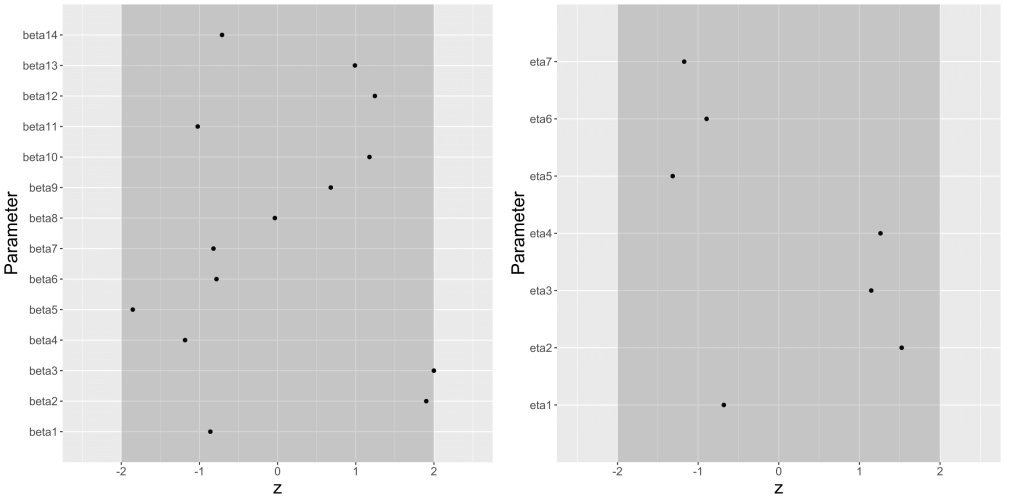


Figure 9: Convergence diagnostics from log-normal distribution.