Received: 15 January 2020 | Accepted: 17 February 2021

DOI: 10.1111/rssc.12483

ORIGINAL ARTICLE

Clustering and automatic labelling within time series of categorical observations—with an application to marine log messages

Emanuele Gramuglia[1] | **Geir Storvik**[1] | **Morten Stakkeland**[2]

[1]Department of Mathematics, University of Oslo, Oslo, Norway

[2]Department of Mathematics, University of Oslo and ABB AS, Oslo, Norway

Correspondence
Geir Storvik, Department of Mathematics, University of Oslo, Oslo, Norway.
Email: geirs@math.uio.no

Funding information
NRC, Grant/Award Number: 237718

Abstract

System logs or log files containing textual messages with associated time stamps are generated by many technologies and systems. The clustering technique proposed in this paper provides a tool to discover and identify patterns or macrolevel events in this data. The motivating application is logs generated by frequency converters in the propulsion system on a ship, while the general setting is fault identification and classification in complex industrial systems. The paper introduces an offline approach for dividing a time series of log messages into a series of discrete segments of random lengths. These segments are clustered into a limited set of states. A state is assumed to correspond to a specific operation or condition of the system, and can be a fault mode or a normal operation. Each of the states can be associated with a specific, limited set of messages, where messages appear in a random or semi-structured order within the segments. These structures are in general not defined a priori. We propose a Bayesian hierarchical model where the states are characterised both by the temporal frequency and the type of messages within each segment. An algorithm for inference based on reversible jump MCMC is proposed. The performance of the method is assessed by both simulations and operational data.

KEYWORDS
clustering, categorical time series, random observation time, multiple change points, reversible jump MCMC

1 | INTRODUCTION

Log files contain messages that are automatically generated by a machine or computer system. The messages are in many cases compact and coded in order to save storage space, such that expert knowledge or a suitable dictionary is needed to understand the contents. The log files are often used for troubleshooting and fault identification by technical experts. Given the trend towards autonomous systems and remote operations in the marine industry, there is a need to automate system monitoring and fault identification based on this kind of textual data. The literature survey given by Li et al. (2017) describes similar data types and problem statements within the field of computer system management. One common target shared between these applications is to detect and identify structures in the data that are not recognised or marked by the machine or system generating the logs. Identification and classification of faults are of particular interest.

1.1 | Marine variable speed drives

A variable speed drive (VSD) is used to control the torque and speed of the motors on ships with electric propulsion motors, and is an integral part of the propulsion system. In the considered systems, the operation of the VSD is controlled and monitored by a drive control unit (DCU), which outputs the log files.

The messages generated by the DCU reflect changes in the operational mode of the variable speed drive in addition to triggered safety functions like alarms and emergency shutdowns. In this context, a change of the operational mode may for instance be that the drive is powered on or off. The variable speed drive is a subsystem onboard the ship and interacts not only with several other systems, but also with crew members who control the vessel and occasionally perform maintenance and repairs. The state of the drive system is hence influenced by a range of systems and functions, but the output in the log files reflect the state of the drive only.

The intention of the presented work is to be able to automatically infer the operations of the variable speed drive and to cluster the stream of messages into segments that represent modes of operation or states, which can be identified and appropriately labelled by technical subject experts. Labelling of different classes of faults is a main focus, and will be used to automate fault diagnostics in future industrial systems. The derived series of consecutive states or modes of operation can be considered as new time series.

1.2 | Methodological approach

In this paper, we introduce an approach for learning macrostructures in event history data. This is performed by dividing the time interval into segments through a clustering process. These segments are characterised by two distinct properties that both are probabilistically modelled. The first is the temporal frequency of the messages occurring within a cluster of a given state, defined by the number of messages per unit of time. The main rationale is that a high number of messages are generated within a relatively short time when a fault occurs. The second property is related to the types of messages that occur within the segment, which also will be described through a probabilistic model.

Each observation from the log file consists of a timestamp and a message (from a finite number of possible categories), and the log file can hence be considered as a marked point process in time, with a data structure similar to those used in event history analysis (Andersen et al., 2012). See Figure 1 for an example.

Our modelling approach is to introduce a latent stochastic process describing the macro-events. This latent process is modelled as an additional point process which will be assumed locally independent of the observation process (Didelez, 2008). The combined model of the latent and observed processes can be seen as a continuous time state space model (with the observation process also corresponding to a continuous-time process). The latent process is assumed to be piecewise deterministic with discontinuities or jumps (Jacobsen, 2006). Between jumps, segments are characterised by marks which describe the segment behaviour. Our model approach is similar to the joint temporal dynamics model for discrete events of Bhattacharjya et al. (2020) but differ in that we only observe one of the processes.

Our approach to learning structures from the observed messages is to divide the time series into segments of similar structure, where similarity here is both with respect to the frequency and to the type of messages. Division into segments includes change-point detection (Chen & Gupta, 2011) but extends such frameworks to allow segments to be allocated to a smaller set of clusters. The approach can also be seen as a problem of clustering time series (Liao, 2005) of categorical observations (Pamminger & Frühwirth-Schnatter, 2010) but differs from most common settings in that clustering is performed *within* time-series. Since the data we consider is sparse, a model-based approach to clustering will be considered (Fraley & Raftery, 2002; Frühwirth-Schnatter, 2011). Arnesen et al. (2016) considered a Bayesian approach to segmentation of sequences of categorical observations and has several similarities to our approach. However, only discrete time observations were considered by Arnesen et al. (2016) and a fixed set of segment types was assumed to be visited in a prespecified order. In our approach, observations occur as a point-process in time with no restrictions on the order at which segment types are visited.

Although there are similarities both in data type and motivation with the computer system management application described by Li et al. (2017), there are some important differences that have motivated our approach to the marine setting. The number of messages is in general smaller within the considered data, making harder modelling assumptions necessary. Furthermore, in the considered

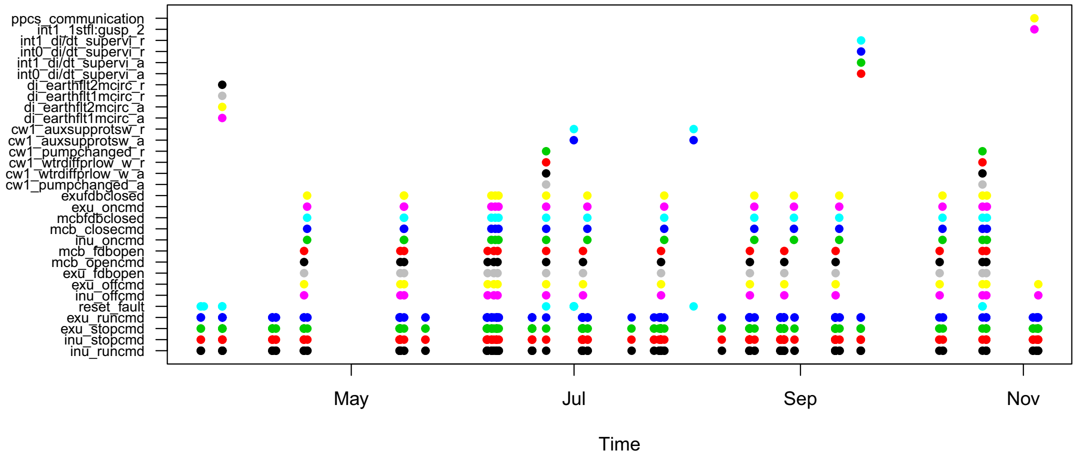

FIGURE 1 A timeline plot of messages. Each point represent a message (many overlapping points due to high frequency periods). The different types of messages are illustrated as different levels along the vertical axis and by different colours

case, the information in each message is limited. Several messages need to be grouped together in order to infer the macro-event, or state, that occurs at a given time.

The rest of the paper is organised as follows: In Section 2, we describe the data at hand. In Section 3, the model is presented while Section 4 describes our Bayesian inference approach. In Section 5, the method is evaluated on simulated data while Section 6 consider a data set from marine vessels. The paper is concluded in Section 7. Details about the algorithms and the experiments are given in the appendices.

2 | MESSAGES AND THEIR STRUCTURE

The output messages in the considered application are static, in the sense that they are generated from a finite set of messages and contain no numeric or dynamic information. The messages are hence treated as categorical variables, and the observations are a sequence of two-dimensional objects $\{(T_i, E_i), i = 1, 2, \ldots\}$. Here $T_i \in \mathbb{R}^+$ is a timestamp denoting the time each message was generated, while $E_i \in \{1, \ldots, m_e\}$ is a categorical variable specifying the recorded message. We assume observations are available in a time window $[t_0, t_{max}]$ with n denoting the number of messages within this interval. The set of possible messages is finite, and the size of the set is in the order of 100–200. The messages in general reflect changes in the internal state of the equipment rather than periods of stationary operation. For instance, a message will be generated when the drive is powered on, while nothing is output if the drive is operated for days or weeks without interruption, until the drive is powered off. An example of a sequence of messages is given in Table 1. A timeline plot of longer series of messages is given in Figure 1. We see both from the table and the figure that the frequency of messages can vary considerably in different time periods and also that there is a wide range of messages being generated within this time frame. Furthermore, in the timeline plot we also see that while some messages appear frequently in most periods, others appear within some specific periods only.

3 | METHODS

3.1 | Notation

We use upper case for stochastic variables and lower case for fixed quantities. We will however use lower case Greek letters for parameters even though these will be treated as stochastic variables in a Bayesian framework. We will use $p(\cdot)$ to denote a generic continuous density with the arguments defining the variable(s) in question. We similarly use $\Pr(\cdot)$ for discrete variables. In cases of combinations of continuous and discrete variables, we use $p(\cdot)$. We denote densities/distribution functions by, for example Dirichlet($\boldsymbol{\lambda};\boldsymbol{\alpha}$) where the term before the semicolon denotes the stochastic variable and the term(s) after denote parameter(s). We use standard parameterisation except for the Gamma distribution where Gamma($\cdot;\kappa,\phi$) denotes the Gamma density with shape parameter κ and *mean* parameter ϕ.

We assume that time is divided into different *segments*, see Figure 2, and use index j to denote segment number. Each segment is characterised by the duration D_j, the number of messages within the segment, N_j, as well as a two-dimensional state vector (V_j, Z_j) where $V_j \in \{0, \ldots, n_v\}$ describes the *activity* state while $Z_j \in \{0, \ldots, n_z\}$ describes the *frequency* state. The activity state here models macro-events or operational modes, such that one activity state can be related to a particular class of

TABLE 1 Example of a log file snippet. The dashed lines and the state column describe the output from the analysis, to be further described in Section 4.2

Message	Timestamp	State V	Message	Timestamp	State V
reset_fault	02-20 10:17:00	4	mcb_opencmd	02-20 11:22:07	2
cw1_auxsupprotsw_a	02-20 10:17:00		mcb_tripcmd	02-20 11:22:07	
sw_initdone	02-20 10:17:03		exu_offcmd	02-20 11:22:07	
reset_fault	02-20 10:17:03		exu_stopcmd	02-20 11:22:07	
gndswclosed	02-20 10:50:23		inu_offcmd	02-20 11:22:07	
cw1_auxsupprotsw_r	02-20 11:09:09		inu_stopcmd	02-20 11:22:07	
black_box_inact._r	02-20 11:09:09		-aru_modulat	02-20 11:22:07	
doorsopened	02-20 11:10:08		mcb_fdbopen	02-20 11:22:07	
ch0_timeout_a	02-20 11:20:35	7	black_box_inact._a	02-20 11:22:08	
ch0_commloss	02-20 11:20:36		ch0_timeout_a	02-20 11:22:10	2
auxsupplyprotsw_a	02-20 11:20:44		mcb_opencmd	02-20 11:22:10	
auxridethrough_w_a	02-20 11:20:44		mcb_tripcmd	02-20 11:22:10	
auxpowerundervolt_a	02-20 11:20:44		exu_offcmd	02-20 11:22:10	
cw1_pump1control_a	02-20 11:20:45		exu_stopcmd	02-20 11:22:10	
cw1_pump2control_a	02-20 11:20:46		inu_offcmd	02-20 11:22:10	
int:_fault_class_2	02-20 11:20:46		inu_stopcmd	02-20 11:22:10	
amc:_fault_class_2	02-20 11:20:46		-aru_modulat	02-20 11:22:10	
pub:_int0_faulted_a	02-20 11:20:46		mcb_fdbopen	02-20 11:22:10	
pub:_int1_faulted_a	02-20 11:20:46		reset_fault	02-20 11:22:11	
int0_1stfl:gusp_2	02-20 11:20:46		exu:_communication	02-20 11:22:17	7
int1_1stfl:gusp_1	02-20 11:20:46		amc:_fault_class_2	02-20 11:22:17	
ppcs_communication	02-20 11:20:47		emergencyoff	02-20 11:22:27	

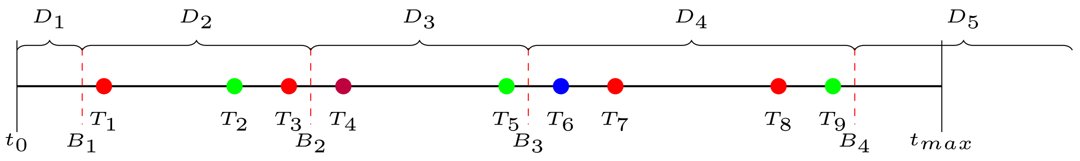

FIGURE 2 Illustration of the process with $n = 9$ messages. The filled circles denote messages positioned at the timestamps T_i with colours describing the types (E_i). B_j's and dashed vertical lines denote break points between segments while the D_j's denote the durations of the segments. Note that $B_5 > t_{max}$ in which case D_5 is unknown (but greater than $t_{max} - B_4$)

failure or a given operation of the equipment. The role of the frequency state is regarded as instrumental; our inferential objective focuses only on the structure of the activity state in terms of messages and its value at specific time points.

As there are long periods in the data set where no messages are output from the system, we define one auxiliary zero state ($V = 0$) to correspond to an *inactive* state in which case also $Z = 0$. For all other states, $V_j \geq 1$ and $Z_j \geq 1$. The inactivity we refer to does not exclusively imply that the system itself is inactive or shut down, but rather the general lack of new incoming messages.

We also define $B_j = t_0 + \sum_{j'=1}^{j} D_{j'}$ to be the *breakpoint* between segments j and $j+1$. Note that there is a one-to-one correspondence between $\{B_j\}$ and $\{D_j\}$ and we will use either representations when convenient.

3.2 | Model

A model-based approach is considered, as the number of messages within the observation interval is limited. We assume a continuous time state space formulation where the segments follow a Markovian structure:

$$\begin{aligned} p(\boldsymbol{V},\boldsymbol{Z},\boldsymbol{D}) &= \prod_{j=1}^{S} p(V_j|V_{j-1})p(Z_j|V_j)p(D_j|Z_j); \\ p(\boldsymbol{N},\boldsymbol{T},\boldsymbol{E}|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{D}) &= \prod_{j=1}^{S} p(N_j|Z_j,D_j)p(\boldsymbol{T}_j|B_{j-1},B_j,N_j)p(\boldsymbol{E}_j|V_j,N_j), \end{aligned} \tag{1}$$

with $B_0 = 0$, $V_0 = \emptyset$. Here S, the number of segments within the observed time window, is defined by the number of breakpoints occurring within the given time window. Note that the last segment will be censored in that its end point will be some time after the last observation. The time dependence between segments is described through the (latent) segment states $\{V_j\}$. The extra discrete state variable Z_j is introduced in order to capture high variability both in the length of the segments and in the number of log messages within the segments. A graphical presentation of the hierarchical structure of the model is given in Figure 3. Each of the conditional distributions, involving several parameters, will be described below.

3.2.1 | State dynamics

We assume a simple dynamic structure where the main dependence structure is that two following segments cannot both be in the zero (inactive) state. This will be modelled through dependence on V_{j-1}:

$$\Pr(V_j = v \mid V_{j-1}) = \begin{cases} \nu_v \mathbb{1}(v \geq 1) & \text{if } V_{j-1} = 0; \\ p_0 \mathbb{1}(v=0) + (1-p_0)\nu_v \mathbb{1}(v \geq 1) & \text{if } V_{j-1} \geq 1; \end{cases} \tag{2}$$

$$\begin{tikzcd}
\text{Parameters} & \boldsymbol{\phi} \arrow[d] & p_0, \boldsymbol{\rho} \arrow[d] & \boldsymbol{\alpha}, \boldsymbol{\beta} \arrow[ddl] & \boldsymbol{\nu} \arrow[d] & \boldsymbol{\lambda} \arrow[ddl] \\
\text{Segment features} & D_j \arrow[d] \arrow[dr] & Z_j \arrow[l] \arrow[d] & & V_j \arrow[ll] \arrow[d] & \\
\text{Observations} & \boldsymbol{T}_j & N_j \arrow[l] \arrow[rr] & & \boldsymbol{E}_j &
\end{tikzcd}$$

FIGURE 3 Graphical representation of the model within a segment. Each segment is characterised by two properties, the activity state (type of messages) and the frequency of messages. The first is captured by the model through the specification of the state V_j, while the second is specified through the state Z_j. The upper row lists the global parameters involved in the model which will be given prior distributions in our Bayesian framework. Note that V_j also depend on V_{j-1} which is not included in the graph

$$\Pr(Z_j = z \,|\, V_j = v) = \begin{cases} \rho_z & \text{if } v \geq 1; \\ \mathbb{1}(z=0) & \text{if } v=0. \end{cases} \tag{3}$$

Here $0 \leq p_0, \rho_z, \nu_v \leq 1$, $\sum_{v=1}^{n_v} \nu_v = \sum_{z=1}^{n_z} \rho_z = 1$.

More sophisticated Markov structures could easily be included into the model, but would require longer time series in order to be able to identify the larger set of parameters. Note that two or more segments can belong to the same (active) state V, see for example the right column of Table 1. This behaviour fits with the natural dynamics of the system where the same operation can be executed repeatedly, still maintaining its distinctiveness.

3.2.2 | Segment features

We assume the following density for the duration of a segment:

$$p(D_j \,|\, Z_j = z) = \begin{cases} \text{Gamma}(D_j;\ \kappa_z, \phi_z) & \text{if } z > 0; \\ \sum_{f=1}^{2} \psi_f \text{Gamma}(D_j;\ \kappa_{0,f}, \phi_{0,f}) & \text{if } z=0. \end{cases} \tag{4}$$

The duration of zero-state segments is assumed to follow the shape of a two-component mixture distribution as there is more variability in the duration of these segments.

3.2.3 | Observations

The number of messages within a segment (only relevant if $V_j > 0$) is then assumed to follow a shifted negative binomial (NB) distribution for active states with segment-dependent parameters:

$$\Pr(N_j = n \,|\, Z_j = z, D_j = l) = \begin{cases} \text{NB}(n-x; \alpha_z, \frac{l}{\beta_z + l}) & \text{if } n > x; \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The shift x gives the requirement of at least $x + 1$ messages within an active segment while the negative binomial distribution accounts for overdispersion related to a Poisson distribution.

Conditioned on the duration of the segment and the number of messages, we assume the time-stamps to be an ordered set of independent identically distributed (iid) variables from the uniform distribution over $[B_{j-1}; B_j)$:

$$p(T_{j,1}, \ldots, T_{j,N_j} \,|\, N_j, B_j, B_{j-1}) = \frac{N_j!}{(B_j - B_{j-1})^{N_j}}. \tag{6}$$

The messages $E_{j,1}, \ldots, E_{j,N_j}$ observed within segment j are assumed to be iid. random variables with

$$\Pr(E_{j,i} = e \,|\, V_j = v) = \lambda_{v,e}, \tag{7}$$

where $\boldsymbol{\lambda}_v = (\lambda_{v,1}, \ldots, \lambda_{v,m_e})$ is a state-dependent probability vector. Also in this case a very simple model is assumed, ignoring the order of the occurrence of messages within segments. Li et al. (2017) defines this as a *parallel* episode. Segments will mostly contain a limited number of messages, making more parameter rich models (e.g serial episodes) difficult to identify.

3.3 | Prior distributions

A fully Bayesian approach is employed for inference. Whenever suitable, we specify conjugate prior distributions on the parameters for computational simplicity. We will define $\boldsymbol{\theta}$ to be the full set of parameters, that is $\boldsymbol{\theta} = \{p_0, \boldsymbol{\rho}, \boldsymbol{\nu}, \{\boldsymbol{\lambda}_v\}, \{\kappa_z\}, \{\phi_z\}, \{\alpha_z\}, \{\beta_z\}\}$.

The probability vectors for the state components as well as the state-dependent probabilities for the messages are assumed to follow *Dirichlet* distributions;

$$p(p_0) = p(\boldsymbol{\rho}) = p(\boldsymbol{\nu}) = p(\boldsymbol{\lambda}_v) = \text{Dirichlet}(\,\cdot\,; \boldsymbol{a}), \quad v = 1, \ldots, n_v. \tag{8}$$

where $\boldsymbol{a}$ is a vector of $a's$ with appropriate length. Note that for p_0 this reduces to the Beta(a, a) distribution. This choice implies no prior knowledge and each possible combination (Z_j, V_j) is *a priori* equally likely. Similarly, each state has an *a priori* structure assigning equal weight to all possible messages. In our experiments we have used $a = 1$.

Concerning (α_z, β_z) for the distribution on the number of messages within segments, and (κ_z, ϕ_z) for the lengths of the segments, we utilise that with a reparameterisation to (α_z, μ_z), (κ_z, η_z) with $\mu_z = \alpha_z/\beta_z$ and $\eta_z = \kappa_z/\phi_z$, a conjugate prior for μ_z and η_z will be an inverse Gamma distribution. In particular we assume $p(\alpha_z, \mu_z) = p(\alpha_z)p(\mu_z)$ and $p(\kappa_z, \eta_z) = p(\kappa_z)p(\eta_z)$ with

$$\begin{aligned} p(\alpha_z) &= \text{Gamma}(1, 1), \quad p(\mu_z) = \text{Inv-Gamma}(1, 1), \\ p(\kappa_z) &= \text{Gamma}(1, 1), \quad p(\eta_z) = \text{Inv-Gamma}(1, 1). \end{aligned}$$

3.4 | Identifiability and relabelling

A well-known obstacle in Bayesian cluster models is the *label switching problem* (Jasra et al., 2005). If both the likelihood function and the prior are invariant under permutations of the labels (in our case the states), the posterior distribution will also be invariant to permutations. The marginal distribution for the parameters of each cluster component will then be identical. Several approaches have been proposed to solve the problem, see for example Jasra et al. (2005) and Rodriguez and Walker (2014). In all experiments, we compare four of the most used relabelling algorithms: *Stephens's method* (Stephens, 2000), *pivotal reordering algorithm* (Marin et al., 2005), *ECR iterarive 1* and *ECR iterative 2* (Papastamoulis & Iliopoulos, 2010; Rodriguez & Walker, 2014). All methods are implemented in the `label.switching` package in R (Papastamoulis, 2016). Our inferential purpose focuses on the state composition in terms of messages and the value of the V state at specific time points $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots)$ called *hotspots* (see Section 4.3 for details), namely that we want to infer $p(\boldsymbol{\lambda}_1 \,|\, \mathcal{D}), \ldots, p(\boldsymbol{\lambda}_{n_v} \,|\, \mathcal{D})$ and $p(V(\boldsymbol{\tau}) \,|\, \mathcal{D})$. The role of the Z state is considered instrumental. On this premise we will carry out the relabelling solely on the above mentioned variables.

4 | BAYESIAN INFERENCE

Full information about the processes is given through the posterior distribution based on the observed timestamps and messages $\{(t_i, e_i), i = 1, \ldots, n\}$:

$$p(S, \boldsymbol{D}_S, \boldsymbol{V}_S, \boldsymbol{Z}_S, \boldsymbol{\theta} \mid \mathcal{D})$$

where S is the number of segments within the observed period, $\boldsymbol{D}_S$ is the set of durations and similarly $\boldsymbol{Z}_S$ and $\boldsymbol{V}_S$ are the set of frequency and activity states, respectively. Furthermore, $\mathcal{D} = \{t_1, \ldots, t_n, e_1, \ldots, e_n\}$ is the set of observations. We will use the notation $\boldsymbol{\psi}_S = (\boldsymbol{D}_S, \boldsymbol{V}_S, \boldsymbol{Z}_S)$ in the following both to simplify notation and to provide a generic presentation of the algorithm.

The posterior distribution is analytically intractable and sampling methods such as Markov chain Monte Carlo (MCMC, Gilks et al., 1995) have to be applied. A further complication in this case is that the dimension of the unknowns depends on the number of segments, which is a random variable. We apply a reversible jump MCMC (Green, 1995) for dealing with this trans-dimensional problem. Moves between different dimensions are obtained by moves, additions and removals of breakpoints.

4.1 | Standard reversible jump MCMC

For notational simplicity, we will suppress the conditioning on $\boldsymbol{\theta}$ in the following. We will here consider a setting where $\boldsymbol{\psi}_S = (\boldsymbol{\psi}_{1,S}, \boldsymbol{\psi}_{2,S})$ and simulation from $p(\boldsymbol{\psi}_{2,S} \mid S, \boldsymbol{\psi}_{1,S}, \boldsymbol{\theta}, \mathcal{D})$ is possible. We assume there exists an auxiliary variable $u_{S \to S^\star}$ such that

$$(\boldsymbol{\psi}_{1,S^\star}, u_{S^\star \to S}) = G(\boldsymbol{\psi}_{1,S}, u_{S \to S^\star})$$

for a one-to-one mapping G. One step of a reversible jump MCMC algorithm is described in Algorithm 1. The acceptance ratio is derived by

$$\begin{aligned}
r_{S \to S^\star} & |\boldsymbol{J}_{S^\star \to S}(\boldsymbol{\psi}_{S^\star}, u_{S^\star \to S})|^{-1} \\
&= \frac{p(S^\star, \boldsymbol{\psi}_{S^\star} | \mathcal{D}) q_S(S|S^\star) q_u^S(u_{S \to S^\star}) p(\boldsymbol{\psi}_{2,S} | S, \boldsymbol{\psi}_{1,S}, \mathcal{D})}{p(S, \boldsymbol{\psi}_S | \mathcal{D}) q_S(S^\star|S) q_u^{S^\star}(u_{S^\star \to S}) p(\boldsymbol{\psi}_{2,S^\star} | S^\star, \boldsymbol{\psi}_{1,S^\star}, \mathcal{D})} \\
&= \frac{p(S^\star, \boldsymbol{\psi}_{S^\star}) p(\mathcal{D}|S^\star, \boldsymbol{\psi}^\star) q_S(S|S^\star) q_u^S(u_{S \to S^\star}) p(\boldsymbol{\psi}_{2,S} | S, \boldsymbol{\psi}_{1,S}, \mathcal{D})}{p(S, \boldsymbol{\psi}_S) p(\mathcal{D}|S, \boldsymbol{\psi}) q_S(S^\star|S) q_u^{S^\star}(u_{S^\star \to S}) p(\boldsymbol{\psi}_{2,S^\star} | S^\star, \boldsymbol{\psi}_{1,S^\star}, \mathcal{D})} \qquad (9) \\
&= \frac{p(S^\star, \boldsymbol{\psi}_{1,S^\star} \mid \mathcal{D}) q_S(S \mid S^\star) q_u^S(u_{S \to S^\star})}{p(S, \boldsymbol{\psi}_{1,S} \mid \mathcal{D}) q_S(S \mid S^\star) q_u^{S^\star}(u_{S^\star \to S})} \qquad (10)
\end{aligned}$$

and $\boldsymbol{J}_{S^\star \to S}(\boldsymbol{\psi}_{S^\star}, u_{S^\star \to S})$ is the Jacobian accounting for the transdimensional move. Equation (9) will be the most convenient form for calculation of the acceptance ratio while Equation (10) is easier for interpretation. Note in particular that if $\boldsymbol{\psi}_{1,S}$ is empty, the algorithm reduces to the idealised one (an MCMC algorithm that only consider the marginal posterior distribution of S). In general, we would like $\boldsymbol{\psi}_{1,S}$ to be low-dimensional compared to $\boldsymbol{\psi}_{2,S}$ in order to obtain an algorithm close to the idealised one. For the specific application, we will define $\boldsymbol{\psi}_{1,S} = \boldsymbol{D}_S$ while $\boldsymbol{\psi}_{2,S} = (\boldsymbol{Z}_S, \boldsymbol{V}_S)$. Note that due to the way $\boldsymbol{\psi}_{2,S^\star}$ is generated, the Jacobian will only involve the transition $\boldsymbol{\psi}_{2,S} \to \boldsymbol{\psi}_{2,S^\star}$.

As stated by e.g Karagiannis and Andrieu (2013), practical efficient implementation of trans-dimensional sampling algorithms is challenging. Preliminary experiments with the ordinary RJMCMC algorithm demonstrated, however, that changes in $(\boldsymbol{V}_s, \boldsymbol{\lambda})$ were problematic, while moving around in the $(S, \boldsymbol{D}_S, \boldsymbol{Z}_s)$ as well as the rest of the hyperparameters worked reasonably well. We have therefore considered an addition of annealing/tempering steps that allowed for large changes in the $(\boldsymbol{V}_s, \boldsymbol{\lambda})$ space while keeping the remaining parameters/variables constant, see supporting materials A.3 for details.

Algorithm 1 Ordinary reversible jump move

```
1: Sample $S^\star \sim q_S(S^\star|S)$
2: Sample $u_{S^\star \to S} \sim q_u^{S^\star}(\cdot)$ and put $(\boldsymbol{\psi}_{1,S^\star}, u_{S^\star \to S}) = G(\boldsymbol{\psi}_{1,S}, u_{S \to S^\star})$
3: Sample $\boldsymbol{\psi}_{2,S^\star} \sim p(\boldsymbol{\psi}_{2,S^\star}|S^\star, \boldsymbol{\psi}_{1,S^\star}, \mathcal{D})$
4: Accept $(S^\star, \boldsymbol{\psi}_{S^\star})$ with probability $r_{S \to S^\star}$ from (9).
```

4.2 | Storage and extraction of results

The reversible jump MCMC algorithm gives simulations $\{S^q, \boldsymbol{D}^q, \boldsymbol{Z}^q, \boldsymbol{V}^q, \boldsymbol{\theta}^q\}$ for $q = 1, \ldots, Q$ where Q is the number of iterations of the algorithm. Summarising global variables not subjected to the change of dimension, such as S or the elements of $\boldsymbol{\theta}$, is straightforward based on the given simulated values. Summarising the segment-dependent variables is more difficult since these are continuous-time processes.

In principle, we would like to present estimates of quantities such as $\Pr(V(\tau) = v|\text{Data})$ for all time points τ, but for the sake of simplicity these quantities will only be given for a selected number of time points, which we call *hotspots*. Simple discretisation of the observation interval is problematic since messages are recorded at very variable frequencies. Our strategy is therefore to define some data-driven hotspot locations corresponding to time points that with high probability belong to intervals containing no breakpoints (so that these time points mostly have a common segment membership across the simulations). State probabilities are reported at these hotspot locations. The details are given in the supporting materials B.

5 | EXPERIMENTS ON SIMULATED DATA

In this section, we demonstrate the performance of the methodology through simulation experiments. In order to make the simulation scenarios realistic, data were simulated from a model with parameters estimated from the data set considered in Section 6. For these experiments, evaluation was focused on reconstruction both segment placement and cluster relation. Convergence diagnostics will be discussed within the operational data analysis in the next section.

Two different scenarios were simulated, in which different amounts of data were available to the estimation process. In the first case, we considered a time window of approximately 18 weeks (3000 h). We generated 10 data sets with sizes ranging from 3350 to 7970 observations, with numbers of segments varying between 83 and 240. For the second scenario, the time window was enlarged to 36 weeks ($\approx$6000 h) resulting in 10 larger data sets (5800–11,870 observations) and consequently more segments (173–378). Both scenarios shared the same set of parameters (see Table 2); the frequency state Z is assumed to have two modalities (in addition to the inactive one) while for V the number of states is 8, each composed by 110 types of possible messages.

TABLE 2 Hyper-parameters related to the frequency states used for the simulations

		Parameters						
Z state	**Mixture**	κ	η	α	β	$\boldsymbol{p_0}$	ρ	ψ
0	1	0.302	6.085	—	—	0.93	—	0.83
	2	1.932	218.759					0.17
1		1.020	1.280	5.991	0.0187		0.82	—
2		3.442	0.009	20.204	0.0004		0.18	

The probability of observing an inactive state after an active state was assumed to be $p_0 = 0.93$ while the probability vector for the activity states is obtained by smoothing the original vector from the analysis of the recorded operational data, $\nu_v = \hat{\nu}_v^{0.3} / \sum_{v=1}^{8} \hat{\nu}_v^{0.3}$ ($\hat{\nu}_v$ is the estimated value)

$$\boldsymbol{\nu} = (0.202,\ 0.130,\ 0.101,\ 0.129,\ 0.079,\ 0.145,\ 0.073,\ 0.138).$$

The $\boldsymbol{\lambda}_v$'s parameters are shown in Table 6.

On each data set, the performance of the method was evaluated in terms of (i) the positions of the estimated breakpoints, (ii) the estimated composition of the activity states and (iii) the estimated state allocations along the timeline. The processes involved are in continuous time. In order to obtain workable evaluation measures, some time discretisation is needed. To evaluate the prediction of breakpoints, we consider all intervals between *uniquely* observed timestamps, $[\tilde{t}_{h-1}, \tilde{t}_h), h = 1, \ldots, H$ (some messages are recorded with equal timestamps). Within each such interval, the maximum number of breakpoints that can occur is two (since we do not allow two following segments to both be inactive). Defining $I_h^{true} \in \{0,\ 1,\ 2\}$ to be the true number of breakpoints in interval $[\tilde{t}_{h-1},\ \tilde{t}_h)$ and I_h its corresponding random variable within the model, we define the loss

$$F_B = \frac{1}{H} \sum_{h=1}^{H} \Pr(I_h \neq I_h^{true} \mid \text{Data}).$$

This loss function represents the average probability of misplacement. To evaluate how well the model predicts the activity states along time we define

$$F_V = \frac{1}{n} \sum_{k=i}^{N} \Pr(V(t_i) \neq V^{true}(t_i) \mid \text{Data})$$

where, as before, $t_1, \ldots, t_n$ are the observed time stamps. Since $\{V(t)\}$ is obtained through a clustering procedure, a relabelling using the true state descriptions as pivots (see Section 3.4) is performed before F_V is calculated.

Finally, for assessing the model performance in terms of the composition of the activity states, we used a symmetric divergence measure based on the *Shannon entropy* (Lin, 1991) between two discrete distributions $\{p_x\}, \{q_x\}$ defined on a common finite space $\mathbb{X}$:

$$ShE(p, q) = \frac{1}{2}(KL(p, q) + KL(q, p))$$

where

$$KL(p, q) = \sum_{x \in \mathbb{R}} p_x \log\left(\frac{2p_x}{p_x + q_x}\right)$$

is a modified Kullback–Leibler divergence measure. An important property of *ShE* is that it is bounded between 0 and 2, the minimum is reached if the two distributions are equal. We used this measure between λ_v^{true} and $\hat{\lambda}_v = E[\lambda \mid \text{Data}]$ for each v.

5.1 | Results

The values of F_B and F_V are shown in Table 3. In all cases, the loss F_B did not exceed 0.006 with little difference between the two scenarios. This indicates that both the *number* of breakpoints and their *positions* are inferred with high accuracy. Concerning F_V, Table 3 shows that for both scenarios the model is able to infer the correct state to a large extent. For scenario 1, the average misplacement error across the 10 simulated data sets is 10% while for the data-richer scenario the error decreases to 2.5%.

In order to investigate the misplacement, confusion matrices for the prediction of I_h^{true} are provided in Table 4 for the cases with the lowest (data set 4) and highest (data set 5) values of F_B within scenario 2. The adjusted Rand indexes (Hubert & Arabie, 1985) are also included. These are measures between zero and one with one corresponding to full agreement between clusters. The estimated number of breakpoints in interval $[\tilde{t}_{h-1}, \tilde{t}_h)$, $\hat{I}_h$, has been chosen by selecting the value with the highest probability. For data set 4, the model has correctly estimated 278 out of 310 breakpoints (the numbers corresponding to I_h^{true} or $\hat{I}_h$ equal to two are doubled). In data set 5, the model performed well in finding the intervals hosting two breakpoints; however, the number of intervals with only one breakpoint was overestimated.

In Table 5, we report how the model estimates the composition of the states by comparing the true parameter vector $\lambda_1^{true}, \ldots, \lambda_8^{true}$ with the obtained estimates; the weighted average takes into account the true number of segments within each state. In all data sets, the *ShE* measures are close to 0 indicating that the model has been able to faithfully infer the states. Also in this case, we see an improvement

TABLE 3 F_B and F_V measures for the simulation experiments. Sc is scenario, D is data set (within the given scenario), M is the measure used, A_D is the average over data sets

		D										**A_D**
Sc	**M**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	F_B	0.006	0.004	0.005	0.006	0.005	0.005	0.004	0.005	0.005	0.005	0.005
	F_V	0.100	0.106	0.100	0.111	0.077	0.040	0.084	0.106	0.159	0.125	0.100
2	F_B	0.005	0.003	0.003	0.003	0.006	0.003	0.003	0.003	0.005	0.005	0.004
	F_V	0.159	0.018	0.006	0.004	0.004	0.009	0.005	0.023	0.011	0.011	0.025

TABLE 4 Confusion Matrices for two data sets within scenario 2. The adjusted Rand indexes (ARI) are included in the last row

	Data set 4			**Data set 5**		
$I_h^{true} \backslash \hat{I}_h$	**0**	**1**	**2**	**0**	**1**	**2**
0	9302	11	1	11,295	14	0
1	1	8	1	3	8	0
2	4	11	135	10	14	130
ARI		0.945			0.915	

TABLE 5 $ShE(\lambda_v^{true}, \widehat{\lambda}_v)$ for the simulation experiments. Here $\widehat{\lambda}_v$ is the posterior mean. Sc is scenario, D is data set, A_D is average over the 10 data sets, A_V is average over states, WA_V is the weighted average over states with weights proportional to the (true) number of segments within states

		Activity (V) state									
Sc	**D**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**A_V**	**WA_V**
1	1	0.005	0.036	0.042	0.029	0.242	0.025	0.020	0.011	0.051	0.086
	2	0.014	0.017	0.032	0.019	0.020	0.014	0.104	0.011	0.029	0.084
	3	0.051	0.027	0.024	0.031	0.035	0.030	0.018	0.044	0.032	0.126
	4	0.017	0.020	0.021	0.079	0.030	0.026	0.036	0.016	0.031	0.119
	5	0.008	0.015	0.010	0.014	0.019	0.028	0.019	0.036	0.019	0.068
	6	0.012	0.023	0.036	0.029	0.018	0.023	0.028	0.010	0.022	0.072
	7	0.007	0.018	0.031	0.035	0.244	0.018	0.033	0.011	0.050	0.109
	8	0.011	0.010	0.084	0.019	0.057	0.040	0.020	0.030	0.034	0.111
	9	0.022	0.036	0.035	0.013	0.027	0.023	0.036	0.020	0.026	0.116
	10	0.043	0.070	0.019	0.034	0.091	0.069	0.019	0.021	0.046	0.170
	A_D	0.019	0.027	0.033	0.030	0.078	0.030	0.033	0.021	0.034	0.106
2	1	0.014	0.058	0.019	0.015	0.071	0.006	0.032	0.009	0.028	0.097
	2	0.003	0.011	0.008	0.010	0.033	0.009	0.019	0.009	0.013	0.033
	3	0.003	0.012	0.009	0.008	0.012	0.009	0.010	0.006	0.009	0.035
	4	0.004	0.008	0.013	0.011	0.034	0.007	0.012	0.011	0.012	0.035
	5	0.003	0.008	0.008	0.010	0.014	0.006	0.010	0.009	0.008	0.034
	6	0.002	0.007	0.012	0.007	0.029	0.007	0.027	0.008	0.012	0.037
	7	0.006	0.011	0.019	0.028	0.035	0.015	0.014	0.013	0.018	0.056
	8	0.007	0.007	0.013	0.010	0.017	0.018	0.018	0.010	0.012	0.049
	9	0.003	0.009	0.009	0.009	0.011	0.005	0.015	0.008	0.009	0.031
	10	0.003	0.010	0.010	0.008	0.011	0.008	0.017	0.004	0.009	0.029
	A_D	0.005	0.014	0.012	0.012	0.027	0.009	0.017	0.009	0.013	0.044

in scenario 2 compared to scenario 1, indicating that a reasonable amount of data is needed in order to estimate these distributions properly.

6 | ANALYSIS OF DATA FROM AN OPERATING SHIP

We considered a data set consisting of 7569 messages recorded from a ship over a period of four and a half years (the actual dates have been removed to anonymise the data). The number of unique message types observed is $m_e = 110$; after some preliminary experiments, we assumed a model with $n_z = 2$ active frequency states and $n_v = 8$ activity states. The main results for the analysis can be split into two parts; the *structural* results, focusing on describing the *global* parameters and the *behavioural* results, displaying the evolution of the system over time. Before analysing the results, we investigated the convergence of the employed MCMC.

6.1 | MCMC diagnostic

In order to assess the convergence of the MCMC algorithm 4, parallel simulations were performed, each of length 100,000, starting from different initial points derived by allocating a different number of breakpoints in different locations along the time axis. Here, one iteration contains one tempering step, five updates of hyperparameters and 100 updates of segment-specific features. The first 70,000 iterations were considered as burn in, and hence discarded. Moreover, a thinning schedule was applied by storing only every fourth iteration. The final tempering schedule was assessed after several trials, the goal being to achieve label switching multiple times at the highest temperature (Celeux et al., 2000). A total number of $r = 30$ increasing temperatures were chosen with the flattest target distribution raised to the power of 1/30. Relying on the four parallel simulations, the Gelman–Rubin statistic (Gelman & Rubin, 1992) was calculated for S, the number of segments. The choice for the analysis of this variable is motivated by the fact that while it provides information about the overall status of the MCMC it is not affected by label switching. The estimated Gelman–Rubin statistics then ended up at about 1.1, see also Figure 4. For other variables/parameters, similar statistics are more difficult to estimate due to the relabelling aspects. However, we have inspected the contents of the states, which were very stable across runs, see supporting materials C.

Figure 5 shows the score of the first principal component for each of the original posterior means of the $\boldsymbol{\lambda}_v$ vectors (raw MCMC output) and after each relabelling method has been employed (ECR iterative 1, ECR iterative 2, Stephens and PRA) for one of the four simulations. The original MCMC output presents several *switching points*, proving that the designed algorithm has good mixing properties. ECR1, ECR2 and Stephens are successful in finding the right permutation pattern while the PRA method seems to fail. The results reported below are based on the output provided by one single chain; the chosen relabelling algorithm is the Stephens method.

6.2 | Structural results

The posterior estimates of the probability vectors $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{n_v}$ describe the composition for the activity states. Table 6 reports the 10 most probable messages and their probabilities of occurrence for each state. Some states are composed by a few dominant messages while others reflect more complex structures. The

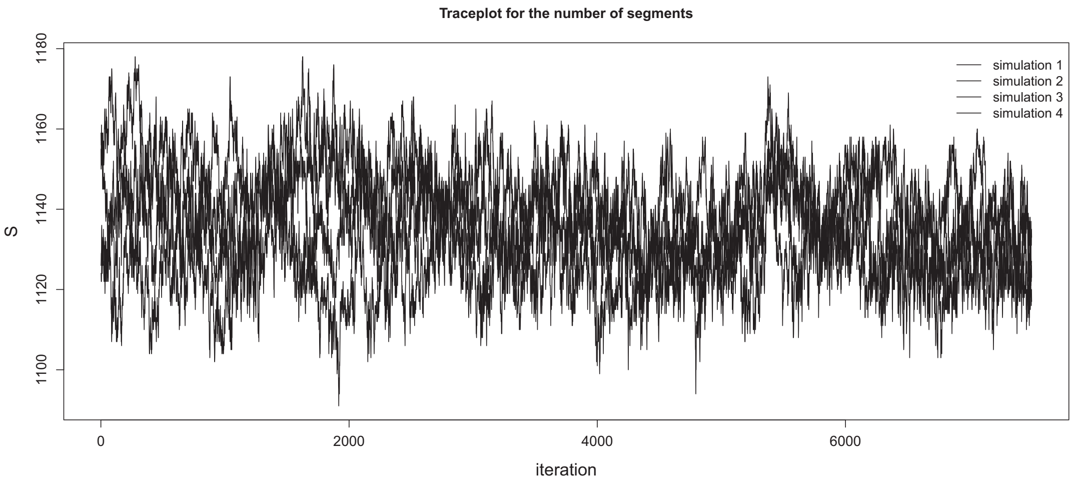

FIGURE 4 Traceplot for the number of segment, different runs given by different line styles

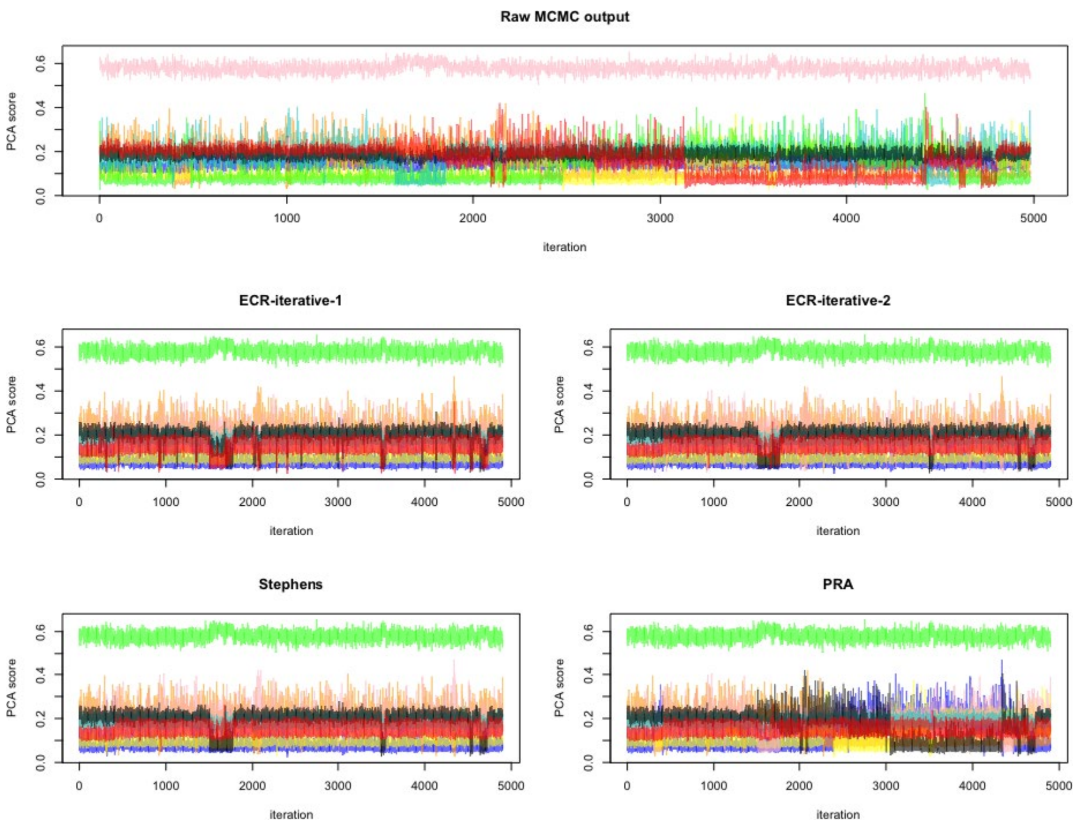

FIGURE 5 Score of the first Principal Component computed from $\lambda_1, \ldots, \lambda_{n_v}$ (each defined by a different color) before and after the relabelling based on methods ECR-1, ECR-2, Stephens and PRA

three most common states, $V = 1, 2$ and 3, identify routine states, and are indeed characterised by standard messages. State $V = 1$, completely defined by only 4 messages, `inu_runcmd`, `inu_stopcmd`, `exu_runcmd` and `exu_stopcmd`, reflects the necessary steps performed related to start/stop operations of the engines. States $V = 2$ and $V = 3$ correspond to the states where the system is initialised and terminated, respectively. Note that these two states are mainly defined by five messages, describing opposite actions.

More interesting insights are gathered from states 4 to 8. They tend to be observed in situations where the system is going away from the normal settings. Based on expert domain input, state 4 is identified as the set of operations recorded during maintenance and inspection phases. States 5 and 6 indicate a malfunction of the cooling system of the ship. State 7 concerns a fault situation related to the power system. Finally, state 8 represents an abrupt shut down of the system due to a specific failure. In general, each state concentrates its mass around a few messages, demonstrating the ability of the model in recognising the different recurrent operative phases experienced by the system.

6.3 | Behavioural results

The composition of the states provides fundamental information about the structure of the system. However, considering the inferred states as a time series of known states and observing the dynamics over time provides a framework for monitoring the operations of the system, including faults and

TABLE 6 The 10 most probable messages within each activity state where $\widehat{\lambda}_{v,e}$ is the estimated posterior mean of observing the corresponding message e within state v. Also the estimated posterior mean of ν_v is included at the top of each state description and the states are ordered in accordance with these probabilities

Activity state 1, $\widehat{\nu}_1 = 0.436$		**Activity state 2, $\widehat{\nu}_2 = 0.134$**		**Activity state 3, $\widehat{\nu}_3 = 0.106$**		**Activity state 4, $\widehat{\nu}_4 = 0.106$**	
Message *e*	$\widehat{\lambda}_{1,e}$	**Message *e***	$\widehat{\lambda}_{2,e}$	**Message *e***	$\widehat{\lambda}_{3,e}$	**Message *e***	$\widehat{\lambda}_{4,e}$
inu_stopcmd	0.2452	mcb_closecmd	0.1238	inu_offcmd	0.0980	reset_fault	0.3219
inu_runcmd	0.2434	mcbfdbclosed	0.1238	mcb_opencmd	0.0968	doorsopened	0.1200
exu_stopcmd	0.2400	exufdbclosed	0.1237	mcb_fdbopen	0.0944	gndswclosed	0.0417
exu_runcmd	0.2365	exu_oncmd	0.1236	exu_offcmd	0.0937	mcb_on_prevented_a	0.0302
reset_fault	0.0032	inu_oncmd	0.1235	exu_fdbopen	0.0851	inu_onprev	0.0289
inu_offcmd	0.0020	inu_runcmd	0.0560	inu_stopcmd	0.0653	cw1_wtrcondhigh_f	0.0251
exu_offcmd	0.0014	exu_runcmd	0.0553	exu_stopcmd	0.0639	exu_fdbopen	0.0194
mcb_opencmd	0.0013	reset_fault	0.0282	inu_runcmd	0.0425	mcb_on_prevented_r	0.0176
mcb_fdbopen	0.0013	inu_stopcmd	0.0203	exu_runcmd	0.0403	+aru_modulat	0.0144
exu_fdbopen	0.0013	exu_stopcmd	0.0198	reset_fault	0.0340	emergencyoff	0.0132

Activity state 5, $\widehat{\nu}_5 = 0.095$		**Activity state 6, $\widehat{\nu}_6 = 0.075$**		**Activity state 7, $\widehat{\nu}_7 = 0.031$**		**Activity state 8, $\widehat{\nu}_8 = 0.017$**	
Message *e*	$\widehat{\lambda}_{5,e}$	**Message *e***	$\widehat{\lambda}_{6,e}$	**Message *e***	$\widehat{\lambda}_{7,e}$	**Message *e***	$\widehat{\lambda}_{8,e}$
auxpowerundervolt_a	0.0884	reset_fault	0.1000	reset_fault	0.2469	reset_fault	0.1882
auxridethrough_w_a	0.0881	cw1_wtrdiffprlow_w_r	0.0580	cw1_nowtrpumpon	0.2007	auxridethrough_f	0.1630
reset_fault	0.0725	cw1_wtrdiffprlow_w_a	0.0580	inu_tripped	0.1984	mcb_tripcmd	0.1514
auxridethrough_w_r	0.0651	cw1_pumpchanged_a	0.0580	cw1_leakagedetect_a	0.1139	mcb_opencmd	0.1511
auxpowerundervolt_r	0.0650	cw1_pumpchanged_r	0.0565	cw1_leakagedetect_r	0.1113	inu_tripped	0.1374
auxsupplyprotsw_a	0.0340	cw1_wtrcondhigh_w_r	0.0377	cw1_pump2overload_a	0.0059	auxridethrough_w_a	0.0060
doorsopened	0.0285	cw1_wtroutpprlow_w_r	0.0354	cw1_pump1overload_a	0.0059	auxpowerundervolt_a	0.0059
cw1_pump1control_a	0.0260	cw1_wtroutpprlow_w_a	0.0353	doorsopened	0.0055	auxridethrough_w_r	0.0058
cw1_pump2control_a	0.0220	inu_tripped	0.0352	sw_initdone	0.0044	auxpowerundervolt_r	0.0057
amc:_fault_class_2	0.0202	cw1_wtrinpprlow_w_a	0.0348	gndswclosed	0.0036	inu_offcmd	0.0045

unwanted incidents. By applying the strategy exposed in section 3570 active segments were identified in the data set; the whole sequence of segments is shown on the top plot of Figure 6. We notice a well-defined pattern involving states 1, 2 and 3; such pattern identifies normal operative conditions. In the centre plot, we observe a snippet of about 8 weeks including 55 segments containing a total of 1143 observations of 87 different types. Among the normal 1-2-3 patterns we can observe unwanted incidents identified by the light green ($V = 7$) circles. Note that some of these apparent faults are generated by maintenance activities. By zooming further, the bottom plot shows an unexpected shutdown situation: after turning off the system (state 3) and on again (state 2) a sequence of faults is observed (4-5-6-7) after which the system is shut down. The clusterisation process provides therefore a simplified framework for performing failure diagnostic; also it makes easier to investigate the implications of the failure on the system by monitoring the states observed immediately after the failure.

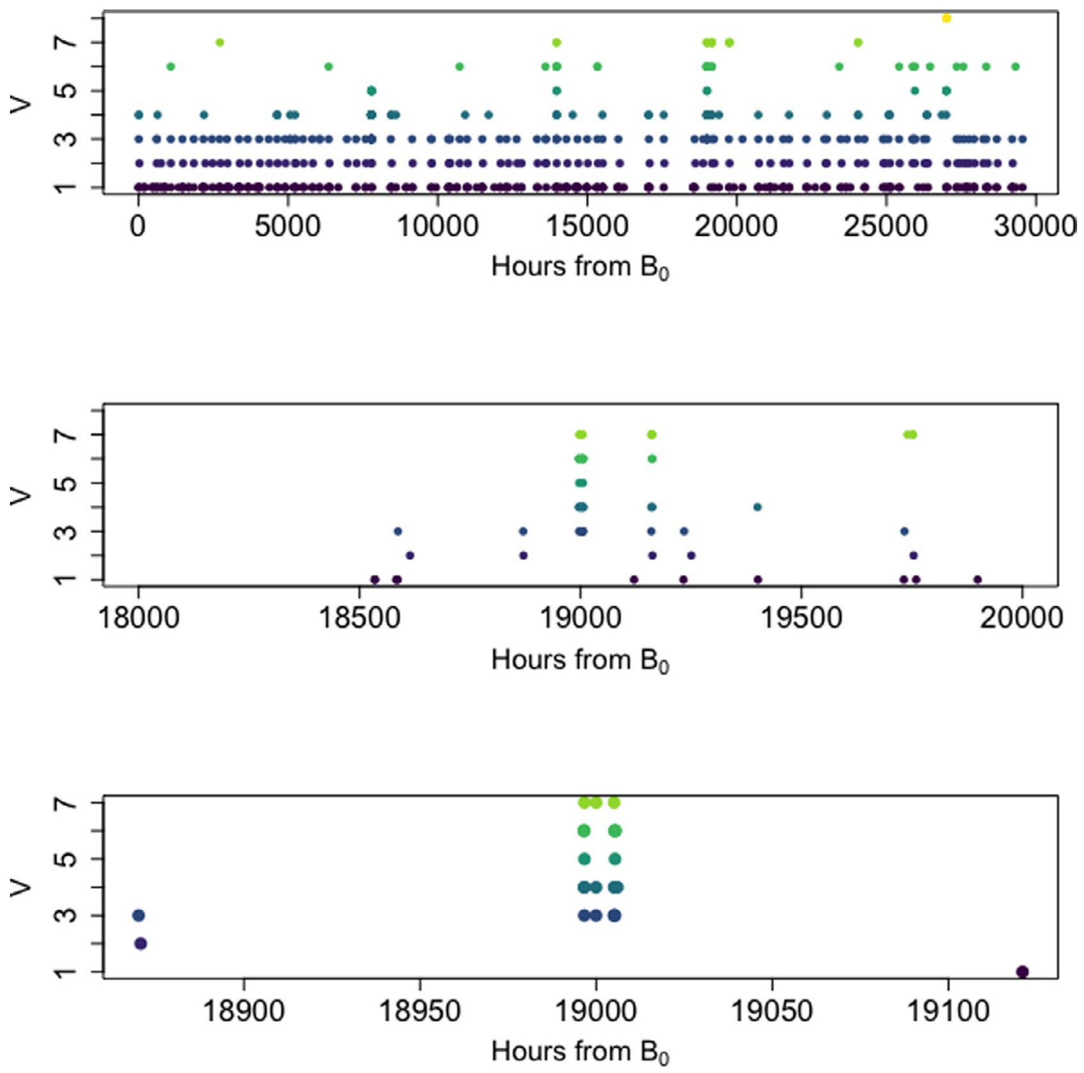

FIGURE 6 Estimated sequence of active V states. The top plot shows the whole sequence of segments. The plot in the center represents a snippet of 55 segments containing both normal and abnormal situations. On the bottom a zoom on an abnormal situation, containing three faults. Each color identifies a different V state

7 | SUMMARY AND DISCUSSION

In this paper, we have proposed a method for clustering observations occurring as a marked point process in time. The clustering process creates segments along the timeline that are assumed to belong uniquely to one of a set of n_v states with differing structural contents. In addition, an inactive state (periods with no observations) is included. Although the number of states is fixed a priori, the composition of the states is unknown and estimated from the data. The objective of the method is to cluster the observations in subsequences while simultaneously learning the structure of the states. We have taken a Bayesian approach and developed an algorithm based on reversible jump MCMC for performing inference in an offline mode. Tempering ideas have been incorporated into the algorithm for speed of convergence.

In the current model, a simple Markovian structure has been considered, partly since the amount of data has been limited and partly because the clustering approach is mainly motivated by an explorative/learning setting. A natural extension of this work will be to transfer the algorithm to a dynamic online setting in which more detailed description of the dynamics is built in to the model. A dynamic setting will also provide the possibility for prediction of states, with the added possibility for prediction of faults/failures.

A more sophisticated model for the content of states can also be considered. In this work, we have only considered the occurrences of messages, not their order within each segment. The ordering certainly provides information, but utilising this information will require more/longer data sets than available at the current time. Combining data from similar but not identical types of equipment is certainly of interest, but is complicated since different units can provide messages with different content. Combining such data in an automatic way will require the use of some semantic similarity measures.

A simulation study has been conducted for evaluating the performance of the method, showing that the model is able to accurately segment the observations and predict the different states with high precision. The exact positions of the breakpoints between segments are however in some cases difficult to infer.

Data from an operational ship has also been analysed. This analysis demonstrates the potential of the method for enabling better understanding of the series of messages, including automated recognition of different complex fault scenarios.

ACKNOWLEDGEMENT

We are grateful to the two referees, the associate editor and the editor for valuable comments to earlier versions, which certainly has improved the paper. We also thank Ingrid Kristine Glad and Inger Skrede for proofreading the paper. We thank BigInsight (`biginsight.no`) and NRC project no. 237718 for support to this project. An R package with an implementation of the algorithm is available at github.uio.no/emanueg/EMACMC.

REFERENCES

Andersen, P.K., Borgan, O., Gill, R.D. & Keiding, N. (2012) *Statistical models based on counting processes*. Berlin: Springer Science & Business Media.

Arnesen, P., Holsclaw, T. & Smyth, P. (2016) Bayesian detection of changepoints in finite-state Markov chains for multiple sequences. *Technometrics*, 58(2), 205–213.

Bhattacharjya, D., Subramanian, D. & Gao, T. (2020) State variable effects in graphical event models. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4291–4297.

Celeux, G., Hurn, M. & Robert, C.P. (2000) Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, 95(451), 957–970.

Chen, J. & Gupta, A.K. (2011) *Parametric statistical change point analysis: With applications to genetics, medicine, and finance*. Berlin: Springer Science & Business Media.

Didelez, V. (2008) Graphical models for marked point processes based on local independence. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1), 245–264.

Fraley, C. & Raftery, A.E. (2002) Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458), 611–631.

Frühwirth-Schnatter, S. (2011) Panel data analysis: A survey on model-based clustering of time series. *Advances in Data Analysis and Classification*, 5(4), 251–280.

Gelman, A. & Rubin, D.B. (1992) Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4), 457–472.

Gilks, W.R., Richardson, S. & Spiegelhalter, D. (1995) *Markov chain Monte Carlo in practice*. Boca Raton: Chapman and Hall/CRC.

Green, P.J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4), 711–732.

Hubert, L. & Arabie, P. (1985) Comparing partitions. *Journal of Classification*, 2(1), 193–218.

Jacobsen, M. (2006) *Point process theory and applications: Marked point and piecewise deterministic processes*. Berlin: Springer Science & Business Media.

Jasra, A., Holmes, C.C. & Stephens, D.A. (2005) Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statistical Science*, 20(1), 50–67.

Karagiannis, G. & Andrieu, C. (2013) Annealed importance sampling reversible jump MCMC algorithms. *Journal of Computational and Graphical Statistics*, 22(3), 623–648.

Li, T., Zeng, C., Jiang, Y., Zhou, W., Tang, L., Liu, Z. & Huang, Y. (2017) Data-driven techniques in computing system management. *ACM Computing Surveys (CSUR)*, 50(3), 1–45.

Liao, T.W. (2005) Clustering of time series data—A survey. *Pattern Recognition*, 38(11), 1857–1874.

Lin, J. (1991) Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1), 145–151.

Marin, J.-M., Mengersen, K. & Robert, C.P. (2005) Bayesian modelling and inference on mixtures of distributions. *Handbook of Statistics*, 25, 459–507.

Pamminger, C. & Frühwirth-Schnatter, S. (2010) Model-based clustering of categorical time series. *Bayesian Analysis*, 5(2), 345–368.

Papastamoulis, P. (2016) label.switching: An R Package for dealing with the label switching problem in MCMC outputs. *Journal of Statistical Software*, 69 (1), 1–24.

Papastamoulis, P. & Iliopoulos, G. (2010) An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. *Journal of Computational and Graphical Statistics*, 19(2), 313–331.

Rodriguez, C.E. & Walker, S.G. (2014) Label switching in Bayesian mixture models: Deterministic relabeling strategies. *Journal of Computational and Graphical Statistics*, 23(1), 25–45.

Stephens, M. (2000) Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4), 795–809.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

How to cite this article: Gramuglia E, Storvik G, Stakkeland M. Clustering and automatic labelling within time series of categorical observations—with an application to marine log messages. *J R Stat Soc Series C*. 2021;00:1–19. https://doi.org/10.1111/rssc.12483