HHS Public Access

Author manuscript

KDD. Author manuscript; available in PMC 2016 October 13.

Published in final edited form as:

KDD. 2016 August ; 2016: 343–352. doi:10.1145/2939672.2939706.

Batch Model for Batched Timestamps Data Analysis with Application to the SSA Disability Program

Qingqi Yue[1], **Ao Yuan**[1], **Xuan Che**[1], **Minh Huynh**[1,2], and **Chunxiao Zhou**[1]

Qingqi Yue: qingqi.yue@nih.gov; Ao Yuan: ao.yuan@nih.gov; Xuan Che: xuan.che@nih.gov; Minh Huynh: mhuynh@impaqint.com; Chunxiao Zhou: chunxiao.zhou@nih.gov

[1]Epidemiology and Biostatistics Section, Rehabilitation Medicine Department, Clinical Center, National Institutes of Health, Bethesda MD 20892, USA

[2]Impac International LLC, Washington, DC 20005, USA

Abstract

The Office of Disability Adjudication and Review (ODAR) is responsible for holding hearings, issuing decisions, and reviewing appeals as part of the Social Security Administration's disability determining process. In order to control and process cases, the ODAR has established a Case Processing and Management System (CPMS) to record management information since December 2003. The CPMS provides a detailed case status history for each case. Due to the large number of appeal requests and limited resources, the number of pending claims at ODAR was over one million cases by March 31, 2015. Our National Institutes of Health (NIH) team collaborated with SSA and developed a Case Status Change Model (CSCM) project to meet the ODAR's urgent need of reducing backlogs and improve hearings and appeals process. One of the key issues in our CSCM project is to estimate the expected service time and its variation for each case status code. The challenge is that the systems recorded job departure times may not be the true job finished times. As the CPMS timestamps data of case status codes showed apparent batch patterns, we proposed a batch model and applied the constrained least squares method to estimate the mean service times and the variances. We also proposed a batch search algorithm to determine the optimal batch partition, as no batch partition was given in the real data. Simulation studies were conducted to evaluate the performance of the proposed methods. Finally, we applied the method to analyze a real CPMS data from ODAR/SSA.

Keywords

batch model; batch information matrix; batched timestamps data; constrained least squares estimation; disability determining process; Case Processing and Management System; service time

The authors and co-authors have no financial or other conflicts of interest to disclose that may bias the reporting of the results presented in this study.

1. INTRODUCTION

The United States Social Security Administration (SSA) administers two of the largest federal programs that provide assistance to people with disabilities: the Social Security disability insurance (SSDI) program and the supplemental security income (SSI) program. SSA's disability determination services (DDS) process is complex and involves five levels. The first level is to make the initial disability determination by state DDS staff after a claimant meets all non-medical eligibility criteria at a SSA field office. There are four levels of appeals, namely reconsideration, hearing, appeals council, and federal court, from one level to the next. A claimant may request a next level appeal if she or he disagrees with the decision of the current level. The Office of Disability Adjudication and Review (ODAR) is responsible for hearing and appeals council. Due to the large number of appeal requests and limited resources, SSA is facing many challenges now. As the Office of Inspector General (OIG) report [20] pointed out, one of the biggest challenges is "While SSA continues focusing on the quality and consistency of hearing decisions, it is facing worsening average processing times and increasing pending hearings". In order to control and process cases, the ODAR has established a Case Processing and Management System (CPMS) to record management information since December 2003. The CPMS provides a detailed case status history for each hearing case, from establishing a case to final decision or dismissal.

Our NIH team collaborated with SSA and developed a Case Status Change Model (CSCM) project to meet the ODAR's urgent need of reducing backlogs and improving hearings process. One of the key issues in our CSCM project is to estimate the expected service time and its variation for each case status code. The major information we extracted from CPMS are case status code (job type), staff ID, staff type, arrival timestamps, and departure timestamps of each status for each case. This is a problem of statistical parameter estimation in queueing model [2], [8], [15], [19], [25]. The work flow recorded by CPMS can be considered as a complex queueing network with multi-stage, multi-task, multi-type, and multi-server [14].

This is a challenging problem, as the complex hearing process system may not follow the typical assumptions used in classical queueing models, such as Poisson arrival processes, exponential service time, and first-come-first-served (FCFS) policy. This is because staff members in the CPMS may freely choose their own service policy; may work on multiple jobs "simultaneously"; they may also switch among multiple unfinished jobs, instead of starting a new job after finished the previous one. However, the CPMS does not record how and when each staff member switches her/ his jobs. We only observe the arrival and the departure timestamps of each job (case status in our system). Note that the arrival time does not necessarily mean the time the staff member starts working on the corresponding job. Since there might be other jobs on the staff member's desk, she/he may continue to work on the unfinished job after the new job arrives. Similarly, the staff member may also hold a finished job for a while and submit it later together with other finished jobs so the departure times may not be the true job finished times.

In this paper, we propose a batch model to make this ill-posed problem tractable. For all the jobs assigned to the same staff, we sort all the departure timestamps of the jobs and cluster

the consecutive ones into groups. Then the jobs whose departure timestamps fall into the same group form a batch. For each batch, we build an equality between the batch time interval, i.e., the time between the latest departure timestamps of the previous batch and of the current batch, and the total service time for all jobs within this batch. Then we can estimate the means and the variances of the service time for each case status code once we have enough equalities. However, due to the complexity of the real data, direct estimation using the least square estimate (LSE) often result in large bias and variance. Our method is based the constrained LSE ([7], [22], [23],[26], [27],), with some special features incorporated, the resulting estimates are accurate and robust as demonstrated by our simulation studies.

The major assumption we use in our batch model is that the staff member approximately spends the whole batch time interval to finish the corresponding batch jobs. The batch service policy can be considered as a batched FCFS in the sense that after the jobs in a batch are processed and recorded, the jobs in the next batch will immediately be processed. We assume that the staff member prefers to finish one batch of jobs before moving to the next batch and have the flexibility to apply arbitrary service policy within a batch. We expect that our assumption may be violated by timestamps recording noise and imperfect batch partition where some jobs may not be completely processed within one batch. We do not require that the batch assumption should be strictly satisfied. Instead, we assume that the time the staff spend on the current batch in past batch time intervals and the time the staff spend on future submitted jobs within the current batch interval follow the same distribution. Then these left-hand and right-hand time truncation error effects may cancel out each other statistically. Our batch model idea has been motivated by mainly two facts. One is the timestamps batch pattern, which is due to some batch processing in the system ([1], [5]); the other is the staff service policy pattern although we cannot observe the ground truth service policy. The departure timestamps for each staff member present apparent clustered pattern. In addition, there are several reasons batches may occur: supervisor may allocate more than one job to a staff member at a time; the staff member may also choose to save finished jobs to be reported jointly to meet periodical job quotas.

To the best of our knowledge, this is the first study on estimating service time of batched timestamps data with unknown service policy, which allow switching among unfinished jobs. Our problem is related to the transactional queue inferences, which deals with the instances when only the service times are observed but not the waiting times [8], [15]. However, transactional queues assume FCFS, and it puts distributional assumptions to the arrival and service times. All of these are unknown in our model. For more details, interested readers in this area may consult the comprehensive list composed by Nazarathy and Pollet [19]. This problem is also similar to the blind source separations [6], notably to the applications on independent component analysis (ICA) [6], principal component analysis [6], and non-negative matrix factorizations (NMF) [16], [17]. The objective for blind source separations is to recover unobserved random processes based on partial observation. For instance, when unobserved sources are linearly combined to form observable outputs, ICA can distinguish both the sources and the combination matrix based on the outputs. In order to obtain the estimates, it is required that the outputs to be repeatedly recorded by the same combination mechanics over variable sources. The similarity between blind source

separations and our problem is that we also observe a partial output (service time + waiting time) and attempt to estimate the unobservable sources (service time). The difference is that our batch model does not have repeated measurements based on the same combination of sources. The combination of different status codes within a batch is always varying. If we model our problem as separating service times and waiting times from the arrival/departure time interval, we only have one setup for all paired arrival and departure timestamps. Extra assumptions are still needed to make this problem tractable.

The rest of the paper is organized as follows. In Section 2 we describe our method for estimating the means and variances of the service times for all the job types; In Section 3 we present results from our simulation studies; Application of the method to a real data is demonstrated in Section 4; In the final Section 5, we summarize the results and discuss some of the possible further researches.

2. THE METHODS

As mentioned in the introduction, the data from a batch processing system may not provide all within batch information such as the true job starting/finished time for each job. In the following subsections we will use the inherited batch pattern due to the batch process to extract the batch level information to overcome the challenge.

2.1 The Batch Information Matrix

As some of the terms in this paper are generally used in both social service (eg. US Social Security Administration and Immigration services) and academic research (eg. Queueing Theory and Computer System) environments, we first clarify these terms by a series of definitions so that they can be consistently understood without confusion for the general readers.

Definition 1—A case processing and management system (CPMS) is a (generally computer and human) system which allows its components to control and process cases and to produce management information, including the arrival times, departure times and possible types of jobs (statuses of the cases) in the system. For a CPMS system, the *processing time* for a status (job) is the time difference between the departure time and the arrival time of the job; the *service time* for a job is the time actually spent to serve and finish the job; the *waiting time* for a job is the time difference of the processing time and the service time for the job; the *system idle time* (after all jobs are finished) is the time the system is waiting for the next job to arrive; the *system service time on a time interval I* is the sum of the total service times for all the jobs on I; the *system resting time on a time interval I* is the total time on I when the system is not available; and the *system idle time on a time interval I* is the sum of all the system idle times (after those jobs that finished in I). If the CPMS system allows batch processing where the jobs are grouped in processing through some way, the data from such a system will be called *batched timestamps data*

The system idle time is the time when the system is available to process jobs but there is no job to process; while the system resting time is the time when the system is not available to process jobs no matter if there are jobs to process or not. For example, a law firm processes

its cases and records the processing information in a computer system. If this law firm finished all its cases on May 1, 2015 and did not receive new cases until May 7, 2015, then the time period between May 1, 2015 and May 7, 2015 will be the system idle time for this law firm. If this law firm does not work on weekends, then the weekends will be the system resting time. Noticing that the system idle time and the system resting time may have overlap periods as in the above law firm example, where the seven days' idle time period at least has one day overlap with the weekends' days, the system resting time, we need to take care of the overlap time when we calculate the system's non-service time.

Remark 1—By Definition 1, we have the following relationships:

$$\text{The processing time for a case} = \text{the waiting time for the case} + \text{the service time for the case.}$$

The total time on a time interval I (which is the interval length of I) = the system service time on I + the system resting time on I + the system idle time on I − the overlap time of the system resting on I and the system idle time on I.

Now we introduce some terms related to batched timestamps data, in particular the timestamps batch patterns inherited from the batch processing which partitions the jobs into batches so that each batch of jobs may be processed or recorded by a special pattern. For example, in an education system where each teacher grades students' homeworks and records the scores in the system, if the homeworks are collected and graded weekly, then the homeworks for each week can be viewed as a batch of homeworks which will be likely graded and recorded based on the teacher's own schedule before the next class. If we check for the timestamps for the students' homework scores in this education system, we would find that the timestamps might have some patterns due to the weekly batch (of homeworks) processing in the system. In the following, we will focus on the timestamps pattern due to the job batches in a batch processing system. For simplicity, we only consider the batched timestamps data with three columns: arrival time, departure time and job type, and we only use the departure time to analyze the batch pattern.

Definition 2—Let $X := \{(a_l, d_l, g_l) | l = 1, 2, \ldots, L\}$ be a batched timestamps dataset from a batch processing CPMS system with the departure timestamps column $D = \{d_l | l = 1, 2, \ldots, L\}$, let $J := \{J_i, i = 1, 2, \ldots, n\}$ be the job batches in the CPMS system which is a partition of all the jobs whose timestamps records form the dataset X. A *batch partition* B for the departure timestamps D is a partition of $D := \cup_{i=1}^{n} B_i$, where B_i is the departure timestamps in the job batch J_i, $i = 1, \ldots, n$; the $B_i's$ are also called the batches of the batch partition B; the largest timestamps t_i in B_i, $t_i = max\{B_i\}$, is called the *departure time of* B_i; the number of timestamps in the batch B_i is called the *size of batch* B_i ($i = 1, \ldots, n$); and n the *number of the batches* of B.

The jobs whose departure timestamps are in each B_i are usually processed together in one batch in the CPMS system. For example, if a CPMS system consists of human servers (e.g. reviewers) with computer recording and scheduling program and it allows weekly job

submissions by servers, then the weekly finished jobs can be viewed as the weekly job batches and their submission timestamps will have a weekly batch partition, which will be most likely clustered around the ends of the weeks.

Definition 3—For a given batched timestamps dataset, a batch partition $B := \{B_i, i = 1, 2, \ldots, n\}$ for the departure timestamps *is in order* if $max\{B_{i-1}\} < min\{B_i\}$, for $i = 2, \ldots, n$.

Remark 2—By Definitions 2 and 3, that a batch partition $B := \{B_i, i = 1, 2, \ldots, n\}$ for the departure timestamps is in order is exactly what we mean to say that the corresponding job batches $J := \{J_i, i = 1, 2, \ldots, n\}$ are batched FCFS as we mentioned in the introduction. In the following, we always assume the batched FCFS policy and the batched timestamps data is always sorted by the departure timestamps.

Often a batch contains some departure time(s) of jobs partially finished in the previous batch, and there maybe some jobs unfinished with their departure time(s) moved to the next batch. For many batches, the amount of departure times "inherited" from the previous batch and those moved to the next batch are not equal, while in some other batches, these the "inherited" and moved out departure times are of the same amount, and so their effects are canceled out. This leads to the following two definitions.

Definition 4—For a given batched timestamps dataset, a batch partition $B := \{B_i, i = 1, 2, \ldots, n\}$ for the departure timestamps is called *balanced* if the unfinished jobs in each batch B_i, $i = 1, 2, \ldots, n$, are the same in terms of the consumed service time.

Definition 5—For a given batched timestamps dataset, a batch partition $B := \{B_i, i = 1, 2, \ldots, n\}$ for the departure timestamps is called *complete* if there is no unfinished jobs in each batch B_i, $i = 1, 2, \ldots, n$.

For a batched timestamps dataset, we find that a particular batch partition will be associated with such a dataset, which is inherited from the batch processing in the corresponding CPMS system. As the timestamps within each batch in the batch partition may not be the true job finished timestamps and tend to clustered at the ends of the batches, batch level information may be more reliable for batched timestamps data. Since a batched timestamps dataset often associates a batch partition which can be viewed as a grouping variable to extract the batch level information as in the following two definitions.

Definition 6—Let $X := \{(a_l, d_l, g_l) | l = 1, 2, \ldots, L\}$ be a batched timestamps dataset with sample size L, where a_l, d_l, and g_l are the arrival time, departure time and the type of the l-th job. Let $B := \{B_i, i = 1, 2, \ldots, n\}$ be a batch partition for the departure timestamps $T := \{d_l | l = 1, 2, \ldots, L\}$, and m be the number of types in the type column $\{g_l | l = 1, 2, \ldots, L\}$ of X and $G_1, \ldots, G_m$ be the m job types. Then the batch counting matrix, $C := (c_{ij})_{n \times m}$ is defined by c_{ij} = the number of elements in $\{(a_l, d_l, g_l) | d_l \in B_i, g_l = G_j, l = 1, 2, \ldots, L\}$ i.e. c_{ij} is the number of all the j-th type G_j jobs whose departure times are in the i-th batch B_i for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

Definition 7—Let $B := \{B_i, i = 1, 2, \ldots, n\}$ be a batch partition for the departure timestamps $\{d_l | l = 1, 2, \ldots, L\}$, $t_i = \max\{t | t \in B_i\}$, for $i = 1, 2, \ldots, n$ are the batch departure times, the interval $(t_{i-1}, t_i]$ is called the interval of B_i for $i = 1, 2, \ldots, n$; the batch service time $b := (b_1, \ldots, b_n)$ is defined to be the system service times on each of the batch intervals $(t_{i-1}, t_i]$ of B, where $t_0 = \min\{t | t \in B_1\}$.

By Definition 7 and Remark 1, the batch service time $b := (b_1, \ldots, b_n)$ can be expressed as $b_i = t_i - t_{i-1}$ − the system idle time on $(t_{i-1}, t_i]$ − the system resting time on $(t_{i-1}, t_i]$, for $i = 1, 2, \ldots, n$, where t_0 is the initial time to start the service for the first case in the dataset, or $t_0 = \min\{t | t \in B_1\}$ if the service initial true starting time is not given.

Proposition 1—Let $X := \{(a_l, d_l, g_l) | l = 1, 2, \ldots, L\}$ be a batched timestamps dataset from a CPMS system. If the CPMS system has the policy that when the system is in idle state (i.e. there is no case to process), it will follow the FCFS policy to serve the first coming case, then the system idle time after the case $(a_{l-1}, d_{l-1}, g_{l-1})$ can be calculated by

$$v_{l,l-1} = \max\left\{0, \min\left\{a_i \middle| i > l-1\right\} - d_{l-1}\right\}.$$

Definition 8—Let $X := \{(a_l, d_l, g_l) | l = 1, 2, \ldots, L\}$ be a batched timestamps dataset with a batch partition $B := \{B_i, i = 1, 2, \ldots, n\}$, $C := (c_{ij})_{n \times m}$ be the batch counting matrix corresponding to B, and $b := (b_1, b_2, \ldots, b_n)'$ be the batch service time vector. We call the combined $n \times (m + 1)$ matrix $[C, b]$ the *batch information matrix* of X corresponding to the batch partition B.

Proposition 2—Let $X := \{(a_l, d_l, g_l) | l = 1, 2, \ldots, L\}$ be a batched timestamps dataset from a CPMS system which has no system resting time on any time intervals between $\min\{a_l | l = 1, 2, \ldots, L\}$ and $max\{d_l | l = 1, 2, \ldots, L\}$. Let $B := \{B_i, i = 1, 2, \ldots, n\}$ be a batch partition with initial service arrival time. If the CPMS system has the policy that is FCFS when it is in idle state, then the batch information matrix $[C, b]$ of X with the batch partition B can be obtained, where C is the batch counting matrix and the batch service time vector $b := (b_1, b_2, \ldots, b_n)'$ is calculated by $b_i = t_i - t_{i-1} - \sum_{d_l \in B_i} v_{l,l-1}$ with $t_i = \max\{t | t \in B_i\}$ as in Definition 6, 7 and Proposition 1.

In the following, we will use the batch information matrix for further analysis.

2.2 The Batch Model

We will continue to use the aforementioned notations. Let $[C, b]$ be the batch information matrix for a given batched timestamps dataset with a given batch partition B. Let S_{ijk} be the random variables representing the k-th service time in the i-th batch for the j-th job type, $(k = 1, 2, \ldots, c_{ij})$, then one can model the batch service times, viewed as random variables, to be the sum of the individual job service times within the batches, i.e.

$$b_i = \sum_{j=1}^{m} \sum_{k=1}^{c_{ij}} S_{ijk} + \varepsilon_i, \quad (i = 1, 2, \ldots, n) \tag{1}$$

where ε_i is the total error including the net time spent for all the partially finished jobs in the i-th batch B_i.

If the batch $B := \{B_i, i = 1, 2, \ldots, n\}$ is complete, then there will be no unfinished job and the error terms ε_i's can be dropped. If $B := \{B_i, i = 1, 2, \ldots, n\}$ is balanced then the net time on all the unfinished jobs can be canceled out and the error terms ε_i's also can be dropped. In general, we may assume that the ε_i's are zero-mean independent random noises. To further simplify model (1), we assume that the service time random variables S_{ijk}'s are independent and stationary across (i, k) for the 1st and 2nd moments, i.e.

$$E[S_{ijk}] = \mu_j, \quad Var[S_{ijk}] = \sigma_j^2$$

For all (i, k)'s, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$; $k = 1, 2, \ldots, c_{ij}$.

In the next subsection, we will use batch model (1) to construct two linear models whose parameters are the means and variances of the service times for each job type.

2.3 The Mean and Variance Estimations by the Least Square Estimator (LSE)

With the above batch model assumptions, we take the expectations and variances respectively on both sides of equation (1) to obtain the following two linear system of equations:

$$E[b_i] = \sum_{j=1}^{m} c_{ij} \mu_j, \quad (i = 1, 2, \ldots, n), \tag{2}$$

$$Var[b_i] = \sum_{j=1}^{m} c_{ij} \sigma_j^2 + Var[\varepsilon_i], \quad (i = 1, 2, \ldots, n), \tag{3}$$

In matrix forms, (2) and (3) can be rewritten as

$$C\mu = E[b], \text{ and } C\sigma^2 = Var[b] - Var[\varepsilon], \tag{4}$$

where $C := (c_{ij})_{n \times m}$ is the batch counting matrix, $\mu = (\mu_1, \ldots, \mu_m)'$ is the vector of the mean service times for the m job types, $\sigma^2 = \left(\sigma_1^2, \ldots, \sigma_m^2\right)'$ is the vector of the corresponding variances, $E[b] = (E[b_1], \ldots, E[b_n])'$, and $Var[b] = (Var[b_1], \ldots, Var[b_n])'$, $Var[\varepsilon] = (Var[\varepsilon_1], \ldots, Var[\varepsilon_n])'$, are the vectors of the means and the variances for the service times of the n

batches. From the systems of equations (4), we can use the Least Square method to obtain the solutions as follows:

$$\mu=(C'C)^{-1}C'E[b], \quad \text{and} \quad \sigma^2=(C'C)^{-1}C'(Var[b]-Var[\varepsilon]). \quad (5)$$

The mean and variance of the batch service time $E[b]$ and $Var[b]$ as well as the variance of the batch error $Var[\varepsilon]$ are unknown. We use the batch service time $b=(b_1, \ldots, b_n)'$ as an approximation to $E[b]$ in (4) so that μ can be estimated by

$$\hat{\mu}=(C'C)^{-1}C'b \quad (6)$$

Similarly we can obtain an estimation for σ^2 as follows. From (4), $E[b]$ can be estimated by

$$\widehat{E[b]}=C\hat{\mu}=C(C'C)^{-1}C'b.$$

Now we use

$$\left[b-\widehat{E[b]}\right]^2=[b-C(C'C)^{-1}C'b]^2=[(I_n-C(C'C)^{-1}C')b]^2$$

as an approximation to $Var[b]-Var[\varepsilon]$ in (5) so that σ^2 can be estimated by

$$\hat{\sigma}^2=(C'C)^{-1}C'[\left(I_n-C(C'C)^{-1}C'\right)b]^2 \quad (7)$$

where I_n is the $n \times n$ identity matrix.

With the above assumptions and notations, we obtain the estimated mean service times $\hat{\mu}$ (for the job types) and the estimated variances $\hat{\sigma}^2$ from (6) and (7), which depend on the given batch partition B. The following proposition shows that for a special batch partition, the estimation formulas will be reduced into the sample mean and variance formulas.

Proposition 3—If the given batch partition has the Batch Counting Matrix C of an n by 1 matrix of 1's, then the mean and variance estimation formulas (6) and (7) become

$$\overline{m}=1/n\sum_{i=1}^{n}b_i, \quad \text{and} \quad \hat{s}^2=1/n\sum_{i=1}^{n}(b_i-\overline{m})^2.$$

Now we consider the quality of the batch based mean and variance estimators (6) and (7). Since our estimators are based on a two-step procedure: the batch service time modeling (1) and the linear regression batch service time model $b=C\mu+\eta$, the quality of estimator (6) depends on modeling (1) assumptions, the Batch Counting Matrix C and the variance of the

zero-mean error η. Under certain regularity conditions, estimator (6) can be proved to be consistent.

As all samples are finite, the standard errors of the estimators $\hat{\mu}$ and $\hat{\sigma}^2$ for a given sample may provide actual information about the quality of the estimators. From the theory of generalized LSE, the estimated standard errors of the estimators $\hat{\mu}$ and $\hat{\sigma}^2$ are approximated as below.

Proposition 4—The standard errors of $\hat{\mu}$ and $\hat{\sigma}^2$ are approximated by

$$se(\hat{\mu}) \approx \sqrt{diag\{C'C)^{-1}C'D(\hat{\mu})C(C'C)^{-1}\}},$$
$$se(\hat{\sigma}^2) \approx \sqrt{diag\left\{(C'C)^{-1}C'D(\hat{\sigma}^2)C(C'C)^{-1}\right\}}$$

where $\mathrm{D}(\hat{\mu})=Diag\left\{[(I_n - C(C'C)^{-1}C')b]^2\right\}$ and

$\mathrm{D}(\hat{\sigma}^2)=diag\{[(I_n - C(C'C)^{-1}C')[(I_n - C(C'C)^{-1}C')b]^2]^2\}$ with the diag(M) being the vector of diagonal elements of the matrix M, DiagV being the diagonal matrix obtained form the vector V, and for a non-negative vector $a = (a_1, \ldots, a_m)$, $\sqrt{a} = (\sqrt{a_1}, \ldots, \sqrt{a_m})$.

Corollary 1—If the diagonal matrices $\mathrm{D}(\hat{\mu})$) and $\mathrm{D}(\hat{\sigma}^2)$) in the above Proposition are replaced by the scaler matrices $k_1 I_n$ and $k_2 I_n$ respectively, where

$$k_1 = \|[\left(I_n - C(C'C)^{-1}C'\right)b]^2\|^2/(n-m),$$

and $k_2 = \|[(I_n - C(C'C)^{-1}C')[(I_n - C(C'C)^{-1}C')b]^2]^2\|^2/(n-m)$ with $\|V\|^2$ being the L^2 norm, then the above approximations are reduced to

$$\overline{se}(\hat{\mu}) \approx \sqrt{diag\left(k_1(C'C)^{-1}\right)}, \quad \overline{se}(\hat{\sigma}^2) \approx \sqrt{diag\left(k_2(C'C)^{-1}\right)}$$

which are the estimation formulas for the corresponding Ordinary Least Square Estimators (OLEs).

2.4 The Mean and Variance Estimations by the LSE with Constraints

In section 2.3, we formulated the mean and variance as the coefficient parameters in linear models and use the LSE to estimate them as in (6) and (7). In real data, there are some natural constraints the parameters have to follow. Since our estimated parameters are the mean service times, so their variances should satisfy the constraint of non-negativity. As the mean service time is part of the processing time, it is an upper bound for the parameters. Another natural constraint is based on the global service time decomposition that the total service time (for all the jobs in the dataset) is the sum of the sub-total service time for each of the job types. If the Batch Information Matrix [*C, b*] is given, then the total service time

in the dataset, and that of the jobs for all the job types can be easily obtained by the sums of the columns of [*C*, *b*], from which the total service time for each type of job can be calculated if the mean service time for each job is given. With those constraints, the mean estimation formula (6) can be improved by the following constrained LSE:

$$\begin{cases} b=C\mu+\eta \\ 0 \le \mu_j \le \pi_j, \;\; j=1\,,\,2,\,\ldots,\,m \\ \sum_{j=1}^{m} C_j\mu_j=\sum_{i=1}^{n} b_i \end{cases} \tag{9}$$

where $\eta = b - E[b]$, $C_j=\sum_{i-1}^{n} c_{ij}$ is the sum of the *j*-th column of the batch counting matrix; π_j is the mean processing time for the type *j* jobs, $j = 1, 2\ldots, m$, which can be easily calculated as the processing time (defined as departure time − arrival time) are observable for each data record. Let $\tilde{\mu}$ be the least squares estimate of μ under model (5) with constraints (9), and $\tilde{\sigma}^2$ be its estimated variance, which can be solved by the R package limSolve [23].

2.5 A Batch Searching Algorithm for 1-dimensional batched timestamps data

In previous subsections, we obtained the mean (service time) and variance estimation formulas for batched timestamps data with a given batch partition. However, in real timestamps data from a CPMS system the batch partition is not given and needs to be found. Even though we know that the timestamps data are batch submitted, due to various reasons (flexible scheduling, individual variabilities, job complexities, etc.) the batch patterns can be complicated and difficult to detect. From our Definition 1, a batch partition is just a special clustering on 1-dimensional data, and in theory any clustering algorithm can be applied for a batch searching. Also, the change detection analysis can be used to find the boundaries of the batches as the change-points. However, as our batch based estimators use the LSE in the second step modeling, and the consistency of the LSE requires that the number of batches should be able to go to infinity in order for the estimators to approach the true parameter values. Therefore, the number of batches should be large in general, as in weekly or daily batch patterns, and the sizes of the batches can be as small as 1, which are not the features that the existing clustering or change detection methods ([3], [4], [9], [10], [11], [12], [13], [18], [21], [24]) are designed for. In the following, we propose a simple batch searching algorithm for 1-dimensional data. From Remark 2, we know that the pattern for a batch partition that is in order should consists of a series of ordered between-batch “jumps” that separate the batches so that the between-batch “jumps” are significantly larger than all those of the within batch “jumps”. The key point is to find the most significant minimum between-batch “jumps” so that we have maximum number of significant batches. The main idea is to use the data’s natural order to define a kind of “z-value” for each “jump” between two successive numbers in the sorted data. Using those “z-values”, we can select the minimum “jump” that is the most significant or by any given significant level, which determine a unique batch partition so that the between batch “jumps” are greater or equal to the minimum “jump”. To define the “ z-value”, we sort all the jumps in decreasing order and compare each jump with all the other smaller jumps, which can be viewed as all the within

batch jumps. The first level "z-value" for each jump can be defined as the normalized z-value by the within batch jumps. As we try to find as many as possible number of batches, it is important to determine the last significant z-value like a waterfall for a mountain stream. We use two way to find this "waterfall location" by using the differences of successive first level z-values to define the final "z-value" for each original jump. This final "z-value" will be used to select the minimum between batch jump.

Let $X := \{x_i | i = 1, 2, \ldots, L\}$ be a numeric vector of length L and $\alpha (< 1)$ be a given significance level, or a range of K is given instead of α. The batch searching algorithm is as follows:

Step 1: Sort X in increasing order $X := \{x_{(i)} | i = 1, 2, \ldots, L\}$.

Step 2: Form the "jumps" as a new numeric vector of length $L - 1$, $Y := \{y_i = x_{(i+1)} - x_{(i)} | i = 1, 2, \ldots, L - 1\}$, where $y_i \geq 0$, $i = 1, 2, \ldots, L - 1$.

Step 3: Choose the unique numbers in Y and sort the resulting vector in decreasing order to get $U := \{u_k | k = 1, 2, \ldots, K\}$, where $u_k > u_{k+1}$, for $k = 1, 2, \ldots, K - 1$, and $u_K \geq 0$.

Step 4: Define $V := \{v_k | k = 1, 2, \ldots, K - 1\}$ by

$$v_k = \frac{u_k - mean(U_k)}{\sqrt{Var(U_k)}},$$

where $U_k = \{y_i | y_i < u_k, i = 1, 2, \ldots, L - 1\} \cup \{u_k exp(-n_k u_k)\}$, n_k is the length of $\{y_i | y_i < u_k, i = 1, 2, \ldots, L - 1\}$, which is viewed as a vector, and the mean(·) and Var(·) are the sample mean and variance estimators.

Step 5: Define $\overline{V} := \{\overline{v}_k | k = 1, 2, \ldots, K - 2\}$ by $\overline{v}_1 = v_1 - v_2$, $\overline{v}_{K-2} = v_{K-1} - v_{K-2}$, $\overline{v}_k = (2v_k - v_{k+1} - v_{k-1})/2$, for $k = 2, \ldots, K - 3$,

Step 5′: Define $\overline{V} := \{\overline{v}_k | k = 1, 2, \ldots, K - 2\}$ by $\overline{v}_k = v_k - v_{k+1}$, for $k = 1, 2, \ldots, K - 3$.

Step 6: Normalize the $\overline{V} := \{\overline{v}_k | k = 1, 2, \ldots, K - 2\}$ to get the "z-values", $Z := \{z_k | k = 1, 2 \ldots, K - 2\}$, where

$$z_k = \frac{\overline{v}_k - mean\left(\overline{V}\right)}{\sqrt{Var\left(\overline{V}\right)}}.$$

Step 7: Let $k_\alpha = \max\{k | z_k > Z_\alpha\}$, where Z_α is the one-sided cut-off z-value of the standard normal corresponding to the significant level α with default value of 0.05, or $k_\alpha = \arg\max\{z_k | k \in R_k\}$ if a range of k, R_k, is given instead of a significant level α, and let

$Y_\alpha = \{y_{i_r} = x_{(i_r+1)} - x_{(i_r)} | y_{i_r} \geq u_{k_\alpha}, r = 1, 2, \ldots, N_\alpha\}$ then the batch partition with the given significant level α can be defined by

$$B_\alpha=\{\min(X) \le x_{(i_1)}<x_{(i_2)}<\cdots<x_{(i_{N_\alpha})}<x_{(L)}=\max(X)\},$$

and the corresponding N_a + 1 batches are given by

$$B_1=\{x_{(i)}|i \le i_1\},\ \ B_r=\{x_{(i)}|i_{r-1}<i \le i_r\},\ \ r=2,\ldots,N_\alpha,$$
$$B_{N_\alpha+1}=\{x_{(i)}|i_{N_\alpha}<i \le L\}.$$

Remark 3—The above algorithm provides an alternative Step (5′) to Step (5) while the rest of Steps being the same to search the most significant batch partition, which we will call kBatch1 and kBatch2 in the simulation section below.

In this section, we established the theoretical basis and practical procedures for the analysis of the batched timestamps data from a CPMS system. Using our batch searching algorithm, we can find a batch partition B if it is not provided. With reasonable conditions (Proposition 2), we can calculate the corresponding Batch Information Matrix [C, b]. Using our batch model (1) (see Section 2.1), we formulate the mean service time $\hat{\mu}$ as the coefficient parameter in the linear regression model $b = C\mu + \eta$. Similar formulas are also provided for the service time variance $\hat{\sigma}^2$. With some assumptions, the LSE (6) or the constrained LSE (9) will be a consistent estimator $\hat{\mu}$ for μ. The quality of the estimator $\hat{\mu}$ depends on the degree of how well the assumptions are met.

3. SIMULATION STUDY

The simulations consist of the following four parts of evaluations: 1) the performance of the LSE estimators $\hat{\mu}$ and $\hat{\sigma}^2$ in (6) and (7); 2) the performance using the constrained LSE estimators $\tilde{\mu}$ and $\tilde{\sigma}^2$ in (9) when the true batch partitions are given; 3) the performance of our batch searching algorithms by comparing with other existing clustering methods; and 4) the performance of the estimators $\hat{\mu}$ and $\hat{\sigma}^2$ when the batch partitions are not given and are determined by our batch searching algorithms.

3.1 The Performance of $\hat{\mu}$ and $\hat{\sigma}^2$ with Given True Batch Partitions

In the simulations, we used the ideal service policy, FCFS, to generate the batched timestamps data with different parameters using two service time distributions, Exponential (for Poisson process) and Normal, with the Uniform distribution for arrival times.

We ran 100 replications to report the average estimation results ($\hat{\mu}$, $\hat{\sigma}^2$) for the estimations of the mean and the variance respectively. The parameters in the simulated data for the replications are as follows: the samples size is 2110 with four job types with frequencies 980, 910, 170, 50, the true means are 6, 8, 10, 11 respectively for the four job type in both the Poisson and Normal data, and the variances of 4, 9, 16, 9 for the Normal. The number of batches in the data are 250. The parameters are chosen based on a yearly workload with daily batches. The 250 batches are determined by equal length intervals so that each batch consists of all the jobs whose departure timestamps fall into each of the equal interval. As the within batch timestamps can only provide information for the job types and batch

departure time, we only use the batch information by Eq. (6) and Eq. (7) to estimate the means and variances for the four job types. We use all the true simulated timestamps to get the sample mean and variance (μ_1 and σ_1^2) as the "Ground truth" for comparisons. The true mean and variance parameters (μ_0 and σ_0^2) as well as the frequency parameters (Frq) for each type (Typ) in the simulation are also listed in Table 1.

The mean and variance estimations have the following estimated standard errors as listed in Table 2, where $se(\mu_1)$ and $se(\sigma_1^2)$, which serve as the "Ground truth" for error estimation comparison, are the standard deviations of the 100 mean and variance estimations used in Table 1, $\overline{se}(\hat{\mu})$ and $\overline{se}(\hat{\sigma}^2)$ are the means of the estimated standard errors from Corollary 1, $se(\hat{\mu})$ and $se(\hat{\sigma}^2)$ are the means of the 100 estimated standard errors by Proposition 4.

Similar results for Normal distribution are listed in Tables 3 and 4.

Note. The frequencies in the tables are close to a real dataset while the true mean and variance parameters are hypothetical. As the number of batches ($n = 250$ as in the above Tables) is the actual sample size in the LSE (6) based on the linear model $b = C\mu + \eta$, the accuracy of the estimations will depend on the number of batches.

3.2 The Improvements of the Constrained LSE (Eq. 9)

The constrained LSE (Eq. 9) is more appropriate due to the meanings of the linear coefficient parameters in our batch model. With the same mean and variance parameters as in Tables 1–4, we calculated the mean and variance estimations by using the constrained LSE ($\tilde{\mu}$ and $\tilde{\sigma}^2$) as well as $\hat{\mu}$, $\hat{\sigma}^2$, μ_1 and σ_1^2 as in Tables 1 and 3 for each one of the 100 simulated datasets. Tables 5 and 6 list their mean values for Poisson and Normal distributions respectively.

From these Tables, we see that the simulations show improvements of the (unconstrained) LSE over the Constrained LSE (Tables 5–6, column 3 and 4 respectively), in particular for the Poisson distribution (Table 5). Especially, the Constrained LSE method overcomes the weakness of the LSE method in variance estimation for the Poisson distributed data as we noticed from the last subsection. Thus, we improved the method so that the estimations will not be affected too much due to the known data distributions.

3.3 The Performance of the Batch Searching Algorithm

With each of the given mean number of centers, we simulated 100 batched datasets whose true number of batches are generated by a Poisson sampling with the given mean number of centers as the parameter. The first part of the simulation is to see if our new methods (kBatch1 and kBatch2) can find the true number of batches. Table 7 lists the results based on the 100 datasets in each of the five mean centers(MC) for five different methods [3], [9], [10], [11], [12], [13], where the square root of Mean Square Error (MSE) is reported, which can be viewed as the (number of center) estimation standard error.

The speed for a searching algorithm is also an important issue. During the above simulations, we recorded the times for each methods and each mean centers on our machine, as in the following Table 8.

The second part of the simulation is to see the accuracy of the batch (cluster) classification when the true number of batches is given. With the same simulated datasets as above, we compare the classification errors of our methods with two commonly used methods, the Mclust and kmeans. Table 9 lists the mean classification error rates for the four methods.

Based on the simulated results in this subsection, we see that our batch searching methods (kBatch1 and kBatch2) perform better than the commonly used existing methods. Table 1 shows that our methods are more accurate in finding the true numbers of clusters (batches); Table 2 shows that our methods are quicker to find the numbers of centers; and Table 3 shows that when the true number of clusters are given, our methods have smaller classification errors.

4. REAL DATA APPLICATION

As the methodology is motivated from the data from the SSAs CPMS system, we apply the method to one dataset from the system. The dataset consists of all the jobs from one reviewer within a specified period. It has three columns: arrival time (or receiving time), departure time (or submission time) and types of the jobs. Each job type represents a sub-job, which we call the status of a case, for the job of processing the whole case. Figure 1 below plots about 100 samples in the data, where each horizontal segment represents the arrival and departure times and its color and line style represent the job type (or status). The submission times (departure times) in the data show an approximately daily batch pattern. The vertical dotted blue lines are drawn at the ends of the batches, which are determined by our batch searching algorithm (see section 2.4) and labeled by the submission dates.

From the Figure, we see that the jobs within each of the batches were submitted at almost the same time, and the submitted time for each individual job in each batch might not be the true finished time of the job. Therefore, the sample mean and variance calculations cannot be directly applied to the data.

For illustration, a monthly batch (combining the batches whose departure times are in the same month) is constructed and the estimated and observed batch times are plotted in the following Figure 2.

In the Figure, the estimated batch times, the blue triangles, are calculated as $\hat{b}_i = \sum_{j=1}^{m} c_{i,j}\hat{\mu}_j$, where $C = (c_{i,j})_{14,6}$ is the Batch Counting Matrix (data not shown here), $\hat{\mu} = (\hat{\mu}_1, \hat{\mu}_2, ..., \hat{\mu}_6)'$ is the vector of the estimated mean service times for the 6 job types. The estimated batch times would reflect the equivalent working time for the jobs in the batches if the true mean working time were the estimated mean, and the observed batch times, the red circles, are the calendar times of the batches. So the interpretation of the Figure is that if a blue triangle (the estimated batch time) is above the corresponding red circle (the observed batch time), the reviewer worked more than his/her own yearly average on that batch (month). For example,

in Figure 2, it seems that this reviewer worked more on most of the months except on the two Novembers (29Nov12, 21NOV13), further information is needed to fully understand the monthly variations for the reviewer.

Based on the simulation in the last section, our Batch method can be used to estimate the means and variances (of the service time in the process system) for all the major job types in the batched timestamps data. The following results are based on our Batch method applied on this real dataset. In brackets are the estimated standard errors.

In the table, the column names (Type1, Type2, Type3, Type4, Type5, Type6) are the job types (the true status codes are not used), where Type6 is the combined other job types in the dataset with small frequencies. The time unit in the table is Calendar day. For example, in average this reviewer can finish “Type3” about 8 (≈ 1/0.12352) “Type3” jobs per Calendar day while the same reviewer could finish about 80 “Type2” jobs per Calendar day if s/he would have worked on a single job type. For this reviewer, the flexible working schedule (Type5 = 0.32 day) seems more efficient than the regular schedule (Type4 = 0.38 day), where Type4 and Type5 are the same type of job with different working schedules. From the standard errors of the mean estimations, one can see the estimations for Type1,Type4 and Type5 are more accurate than those for Type2, Type3, and Type6, which may reflect the nature of the job types. The estimations for the variances and their standard errors have similar pattern but with less accuracy than that for the mean estimation as we observed in the simulations of the method. Since we used the non-negative constraints in our parameter estimation, the estimation of 0 for the variance of Type2 is likely due to the small mean estimation of Type2, the non-negative constraints and some possible outlier batches where the reviewer might take leaves. As we do not have further information about the reviewers working time over all the periods covered in the data, we can not give any further explanations. The batch information matrix, including the number of jobs for each type, the batch time and the batch departure time, is not shown here and can be obtained by request. Using the estimated mean service times as in Table 5 and the numbers of jobs for each job types, we can estimate batch time for each batch.

5. DISCUSSION

We introduced a batch model for batched timestamps data analysis using the batch information matrix to overcome the challenge due to the unreliable information within each batch. In particular, when the batched timestamps data comes from a case processing system with three columns (arrival time, departure time, and job type), our method can be used to estimate the means and variances of the service time for each of the major job types. When each data record is a true observation (the arrival time and departure time are the true arrival and departure time of the job), one can use the single timestamps batch partition and our method becomes the sample mean and variance formula. Our simulations shown the method gives reasonable estimations under various typical situations. As the batch partition is not given in real data, we developed a batch searching algorithm, which fully take the advantage of the 1-dimensional batched timestamps data to make the algorithm more accurate and computationally more efficient than some existing clustering algorithms. The results from the real data application will be further verified. As our batch method just uses the LSE for

linear model with the batch information matrix extracted from batched timestamps data, there are certainly a lot of different ways to further improve the method. We suggest two directions for further developments. The first is to find ways to improve the quality of the batch information matrix. The more accurate the time spent on the jobs within a batch represents the time of the batch, the higher the quality is. The second is to find more effective ways for the analysis of the batch information matrix extracted from some batched timestamps data. While we choose the generalized LSE method due to the fact that in the batch information matrix, the numbers of the jobs for the types are linearly related to the total times of the batches, there are certainly many other ways, parametric or non-parametric, to explore for the improvements of the method. After the method is fully developed, it can be applied to analyze the large scale batched timestamps data like the one we currently have as well as to other Federal Agency's data.

Acknowledgments

This research was supported by Social Security Administration and National Institutes of Health (SSA-NIH) Interagency Agreements and by the NIH intramural research program.

References

1. Ahmadi JH, Ahmadi RH, Dasu S, Tang CS. Batching and scheduling jobs on batch and discrete processors. Operations Research. 1992; 39:750–763.
2. Armero C, Conesa D. Statistical performance of a multiclass bulk production queueing system. European Journal of Operational Research. 2004; 158(3):649–661.
3. Bodenhofer U, Kothmeier A, Hochreiter S. APCluster: an R package for affinity propagation clustering. Bioinformatics. 2011; 27:2463–2464. DOI: 10.1093/bioinformatics/btr406 [PubMed: 21737437]
4. Chen, J.; Gupta, AK. Parametric statistical change point analysis. Birkhauser; 2000.
5. Cheng TCE, Wang G. Batching and scheduling to minimize the makespan in the two-machine flowshop. IIE Transactions. 1998; 30:447–453.
6. Comon, P.; Jutten, C. Handbook of Blind Source Separation: Independent component analysis and applications. Academic press; 2010.
7. Coons SA. Constrained least-squares. Computers & Graphics. 1978; 3(1):43–47.
8. Daley DJ, Servi LD. Exploiting Markov chains to infer queue length from transactional data. Journal of applied probability. 1992:713–732.
9. Fraley C, Raftery AE. Model-based clustering, discriminant analysis and density estimation. Journal of the American Statistical Association. 2002; 97:611–631.
10. Fraley, C.; Raftery, AE.; Murphy, TB.; Scrucca, L. mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation. Department of Statistics, University of Washington; 2012. Technical Report No. 597
11. Frey BJ, Dueck D. Clustering by passing messages between data points. Science. 2007; 315:972–977. DOI: 10.1126/science.1136800 [PubMed: 17218491]
12. Hennig C. Asymmetric linear dimension reduction for classification. Journal of Computational and Graphical Statistics. 2004; 13:930–945.
13. Hennig, C. A method for visual cluster validation. In: Weihs, C.; Gaul, W., editors. Classification – The Ubiquitous Challenge. Springer; Heidelberg: 2005. p. 153-160.
14. Kleinrock, L. Queueing Systems Volume 1–2: Theory and Computer Applications. John Wiley & Sons; New York: 1976.
15. Larson RC. The queue inference engine: deducing queue statistics from transactional data. Management Science. 1990; 36(5):586–601.

16. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999; 401(6755):788–791. [PubMed: 10548103]
17. Lee DD, Seung HS. Algorithms for non-negative matrix factorization. Advances in neural information processing systems. 2001; 13:556–562.
18. Liu, H.; Liu, T.; Wu, J.; Tao, D.; Fu, Y. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2015. Spectral ensemble clustering; p. 715-724.
19. Nazarathy, Y.; Pollet, PK. Parameter and State Estimation in Queues and Related Stochastic Models: A Bibliography. 2011. http://www.maths.uq.edu.au/pkp/papers/Qest/QEstAnnBib.pdf
20. OIG report: Fiscal Year 2015 Inspector General Statement on the Social Security Administration's Major Management and Performance Challenges. 2015. https://oig.ssa.gov/sites/default/files/audit/full/pdf/A-02-16-50118_0.pdf
21. Qahtan, AA.; Alharbi, B.; Wang, S.; Zhang, X. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2015. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams; p. 935-944.
22. Rao CR, Toutenburg H, Shalabh H. Linear Models and Generalizations: Least Squares and Alternatives. 2008
23. Soetaert K, Van den Meersche K, van Oevelen D. limSolve: Solving Linear Inverse Models. 2009 R-package version 1.5.1.
24. Song, X.; Wu, M.; Jermaine, C.; Ranka, S. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2007. Statistical change detection for multi-dimensional data; p. 667-676.
25. Sutton C, Jordan MI. Bayesian inference for queueing networks and modeling of internet services. The Annals of Applied Statistics. 2011; 5(1):254–282.
26. Van De Geer, S. Encyclopedia of Statistics in Behavioral Science. John Wiley & Sons, Ltd; Chichester: 2005. Least squares estimation.
27. Wakefield, J. Bayesian and frequentist regression methods. Springer Science & Business Media; 2013.

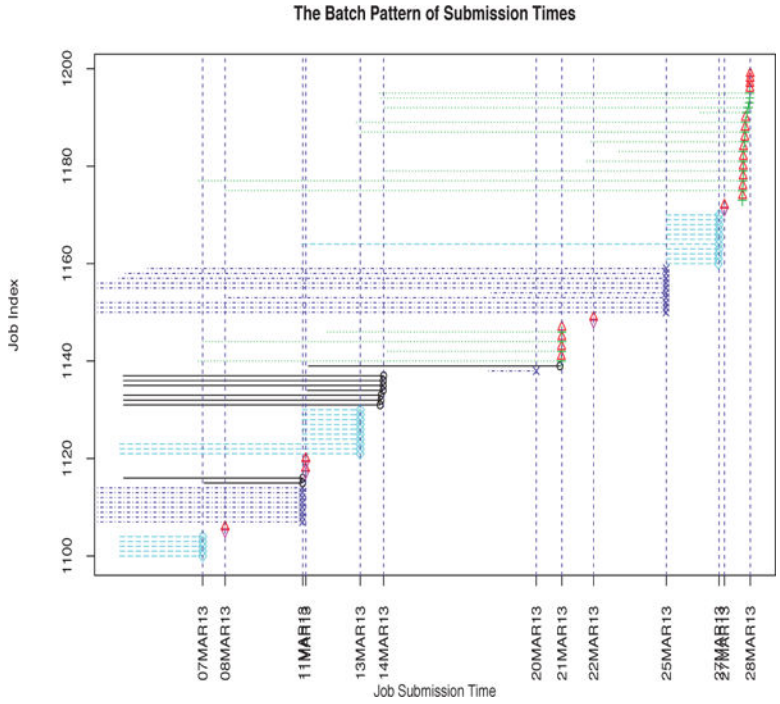

Figure 1.
The plot for job arrival and departure times

Figure 2.
Monthly batch times (Estimated vs. Observed)

Table 1

The means of the 100 mean and variance estimations for the Poisson distribution with number of batches $n = 250$.

Typ	Frq	$\hat{\mu}$	μ_1	μ_0	$\hat{\sigma}^2$	σ_1^2	σ_0^2
1	980	5.33	5.96	6.00	26.01	35.53	36.00
2	910	7.28	8.00	8.00	49.71	64.19	64.00
3	170	9.20	10.03	10.00	76.98	99.53	100.00
4	50	10.59	11.01	11.00	83.57	118.59	121.00

Table 2

The estimation (mean and variance) standard errors with the same 100 simulated datasets as in Table 1.

Typ	Frq	$se(\mu_1)$	$\overline{se}(\hat{\mu})$	$se(\hat{\mu})$	$se(\sigma_1^2)$	$se(\hat{\sigma}^2)$	$se(\hat{\sigma}^2)$
1	980	0.51	0.50	0.47	13.30	15.38	12.57
2	910	0.48	0.54	0.52	16.26	16.64	14.21
3	170	1.57	1.55	1.53	45.10	47.46	43.89
4	50	2.63	2.97	2.86	73.34	90.86	80.94

Table 3

The means of the 100 mean and variance estimations for the Normal distribution with number of batches $n = 250$.

Typ	Frq	$\hat{\mu}$	μ_1	μ_0	$\hat{\sigma}^2$	σ_1^2	σ_0^2
1	980	5.92	5.99	6.00	3.87	4.00	4.00
2	910	7.87	8.01	8.00	8.79	9.01	9.00
3	170	9.83	9.97	10.00	14.30	15.77	16.00
4	50	11.01	11.03	11.00	9.23	8.92	9.00

KDD. Author manuscript; available in PMC 2016 October 13.

Table 4

The estimation (mean and variance) errors with the same 100 simulated datasets as in Table 3.

Typ	Frq	$se(\mu_1)$	$\overline{se}(\hat{\mu})$	$se(\hat{\mu})$	$se(\sigma_1^2)$	$se(\hat{\sigma}^2)$	$se(\hat{\sigma}^2)$
1	980	0.18	0.19	0.18	1.99	2.05	1.90
2	910	0.19	0.20	0.21	2.13	2.21	2.25
3	170	0.66	0.58	0.59	6.59	6.22	6.44
4	50	0.97	1.09	1.06	10.13	11.75	11.27

KDD. Author manuscript; available in PMC 2016 October 13.

Table 5

For Poisson distribution.

Typ	Frq	$\hat{\mu}$	$\tilde{\mu}$	μ_1	$\hat{\sigma}^2$	$\tilde{\sigma}^2$	σ_1^2
1	980	5.33	5.90	5.96	26.08	35.10	35.53
2	910	7.28	8.03	8.00	49.86	63.83	64.19
3	170	9.20	10.11	10.03	76.60	95.11	99.53
4	50	10.59	11.60	11.01	79.33	99.10	118.59

Table 6

For Normal distribution.

Typ	Frq	$\hat{\mu}$	$\tilde{\mu}$	μ_1	$\hat{\sigma}^2$	$\tilde{\sigma}^2$	σ_1^2
1	980	5.92	6.01	5.99	3.90	4.09	4.00
2	910	7.87	7.99	8.01	8.83	9.18	9.01
3	170	9.83	9.97	9.97	14.35	14.87	15.77
4	50	11.01	11.17	11.03	7.41	7.74	8.92

Table 7

The square roots of MSE for the batch number determinations

MC	apclust	pamk	Mclust	kBatch1	kBatch2
5	0	0.1	3.33	0	0
20	9.17	11.53	1.11	0	0
50	39.03	48.38	1.36	0	0
100	88.68	99.13	13.90	0	0
200	186.46	197.57	58.17	0.1	0.1

Table 8

Computational times (in seconds) for the above simulations for each methods

MC	apclust	pamk	Mclust	kBatch1	kBatch2
5	295.04	192.19	14.78	3.62	3.90
20	325.23	128.21	143.85	3.6	4.00
50	368.09	188.31	831.62	4.15	4.00
100	399.56	271.51	6487.86	4.38	4.43
200	527.41	426.01	21071.60	5.24	5.38

Table 9

Mean classification error rates for 4 methods

MC	Mclust	kmeans	kBatch1	kBatch2
5	0	0.37	0	0
20	0.09	0.37	0	0
50	0.14	0.37	0	0
100	0.16	0.36	0	0
200	0.17	0.35	0	0

Table 10

The mean and variance estimation based on our Batch model method

Est.	Type1	Type2	Type3	Type4	Type5	Type6
mean	0.12	0.01	0.12	0.38	0.32	0.17
se_m	(0.02)	(0.19)	(0.19)	(0.03)	(0.04)	(0.11)
var	0.10	0.00	0.21	0.25	0.13	1.01
se_v	(0.04)	(0.38)	(0.39)	(0.07)	(0.08)	(0.34)
Freq	585	493	422	367	330	232