The Poisson Multinomial Distribution and Its Applications in Voting Theory, Ecological Inference, and Machine Learning

Zhengzhi Lin, Yueyao Wang, and Yili Hong

Department of Statistics, Virginia Tech, Blacksburg, VA 24061

September 18, 2021

Abstract

The Poisson Multinomial Distribution (PMD) is the sum of n independent indicators, in which each indicator is an m element vector that follows a multinomial distribution. The PMD is useful in many areas such as, machine learning, uncertainty quantification, and voting theory. The distribution function (e.g., the probability mass function) has been studied by many authors for a long time, but there is no general computing method available for its distribution functions. In this paper, we develop algorithms to compute the probability mass function for the PMD, and we develop an R package that can calculate the probability mass function efficiently. We also study the accuracy of different methods. We illustrate the use of the PMD with three applications from voting theory, ecological inference, machine learning. We also provide examples to demonstrate the use of the R package.

Key Words: Binomial distribution; Classification; Poisson Binomial Distribution; Machine Learning; Uncertainty Quantification; Voting Theory

Contents

1 Introduction

1.1 Motivation

The Poisson Multinomial Distribution (PMD) is defined as sum of different independent Multinomial distributions. It has applications in game theory (**?**), digital imaging (**?**), machine learning (**?**), etc. One intriguing area that involves PMD is political science. It would be good to illustrate PMD a little bit further via a very simple example in election scenario. Suppose a committee with n independent members needs to elect a chairman between m candidates. Each member has a different voting behavior. Mostly, people pay attention to the result of an election. Statisticians focus on the probability for each electoral outcome to happen. An (n, m) PMD is a perfect model for this example by seeing members as independent random variables following different Multinomial distributions. The probability vector of the Multinomial distribution regards to the probabilities for a member voting for each candidates. By this way, we will be allowed to compute the probability of each electoral result which is referred as probability mass function of PMD. Without a doubt, computing the probabilities of electoral results will be of greatest concern. Enumeration could be a fine method when $n \times m$ is small. However, when we have more members(large n) or more candidates(large m), the computing will be a dilemma that is impossible to overcome by enumeration. Thus arises the need of methods that are computationally cheap and precise at the meantime. Additionally, in machine learning, especially in classification context, when we classify n samples one by one into m categories, then the total number of samples assigned to each category follows a PMD. People who study the probability of classification outcomes also encounters same dilemma. Not only in the field of classification and political science, computing pmf is frequently required in scenarios that PMDs are involved. However, there is no existing algorithm for computing pmf for PMD. Hereby it is a necessity to build one that can calculate pmf of PMDs efficiently.

1.2 Related Literature and Contribution of This Work

Some former studies have uncover PMD's structure and properties, **?**) shows the Fourier transformation of PMD is sparse and provide a theorem that there exist algorithms to calculate PMD's density. **?**) also prove us PMD is ϵ-cover and Central Limit Theory is valid for PMD. Other papers such as **?**) illustrates us some interesting applications of PMD and its sparsity property. Motivated by the huge practical value of PMD, we develop an algorithm based on prior studies to compute probability mass function of PMDs. Our algorithm contains three methods. They are $\mathbf{DFT-CF}$ which is based on multi-dimensional Fourier transformation, **SIM** that is a simulation method and **N.A** which applies **C.L.T** to approximate the PMD. The algorithm enables researchers to calculate pmf of PMDs under certain requirements of

accuracy and timing even when n and m gets large. We also construct a R package to perform our algorithm, the functions contained in the package are able to compute pmf and cdf of PMD using all methods we mentioned in this paper as well as generate random samples from given PMD.

1.3 Overview

The rest of the paper is organized as follows. In the second part, we describe the mathematical definition of Poisson Multinomial distribution and list some of its useful properties. In the third part, we introduce three methods that we include in the algorithm to compute PMDs' pmf by details. The fourth part of this article studies the accuracy and time efficiency characteristics of the three methods. In the fifth part, we illustrate application of our algorithm via three examples respectively in the fields of voting theory, statistical inference for aggregated data and classification. In the sixth part, we introduce our R package that implemented with the algorithm and the seventh part of this paper draws the conclusion and prospective research areas.

2 Poisson Multinomial Distribution

2.1 Definition of the Distribution

Let $\mathbf{I}_i = (I_{i1}, \ldots, I_{im})', i = 1, \ldots, n$ be a random indicator vector that follows multinomial distribution with associated probabilities $\boldsymbol{p}_i = (p_{i1}, \ldots, p_{im})'$ and for a fixed i, $\sum_{j=1}^{m} I_{ij} = 1$. Then the sum $\boldsymbol{X} = (X_1, \ldots, X_m)' = \sum_{i=1}^{n} \mathbf{I}_i$ follows a Poisson Multinomial Distribution with corresponding probability matrix $\mathbf{P}_{\mathrm{n}\times\mathrm{m}} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)'$, denoted it as

$$\boldsymbol{X} \sim \mathrm{PMD}(\mathbf{P}_{\mathrm{n}\times\mathrm{m}}).$$

where $X_j = \sum_{i=1}^{n} I_{ij}, j = 1, \ldots, m$. The matrix $\mathbf{P}$ is called Success Probability Matrix (SPM).

$$\mathbf{P}_{\mathrm{n}\times\mathrm{m}} = \begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nm} \end{pmatrix}.$$

It is trivial to find that the random variables, $X_1, \ldots, X_m$ actually have to be constrained under linear equation

$$\sum_{j=1}^{m} X_j = n$$

. Hence we can replace one of them, for instance, X_m with $n - \sum_{j=1}^{m-1} X_j$.

Let vector $\boldsymbol{x} = (x_1, \ldots, x_m)'$ be a realization of a PMD random variable $\mathbf{X}$, the probability mass function (pmf) of PMD,

$$\Pr(\mathbf{X} = \boldsymbol{x}) = \Pr\left(X_1 = x_1, \ldots, X_m = x_{m-1}, X_m = n - \sum_{i=1}^{m} x_i\right)$$

is of interest.

Example 1 Suppose there are three candidates and four voters, the result of voting is a random variable $\mathbf{X} \sim \text{PMD}(\mathbf{P})$. Based on prior information,

$$\mathbf{P}_{4\times 3} = \begin{pmatrix} 0.1 & 0.2 & 0.7 \\ 0.5 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.1 \\ 0.8 & 0.1 & 0.1 \end{pmatrix}.$$

There are 15 distinct outcomes of the election, that is, the pmf is composed of 15 distinct non-zero mass points. It is trivial to compute the pmf of $\mathbf{X}$ through enumeration. For instance, the probability of a result that the first candidate gets 4 votes and others gets 0 vote, that is, $\boldsymbol{x} = (4, 0, 0)'$.

$$P\{\mathbf{X} = \boldsymbol{x}\} = 0.1 \times 0.5 \times 0.4 \times 0.8 = 0.016.$$

Also, the probability of $\mathbf{X} = (1, 3, 0)'$ is

$$\begin{aligned} P\{\mathbf{X} = (1,3,0)\} =& 0.1 \times 0.2 \times 0.5 \times 0.1 + 0.5 \times 0.2 \times 0.5 \times 0.1 \\ &+ 0.4 \times 0.2 \times 0.2 \times 0.1 + 0.8 \times 0.2 \times 0.2 \times 0.5 = 0.0236. \end{aligned}$$

□

When dimension of $\mathbf{P}$ is small, enumeration is an exact way to calculate the probability mass function. However, as n × m gets larger enumeration becomes impossible since we will have to compute $\binom{n+m-1}{m-1}$ possible outcomes which is calculated by the number of non-negtive integer solution for equation $x_1 + \cdots + x_m = n$.

Note that when the SPM is identical across all rows, that is, $\mathrm{I_i}$, $i = 1, \ldots n$ are identically distributed, the distribution of X can be simplified as multinomial distribution. Hence, the PMD is a generalization of the multinomial distribution. When $m = 1$, the PMD is reduced to the Poisson binomial distribution as in **?**).

In related literature, **?**) consider the exact and approximate methods for computing the pmf of the Poisson binomial distribution. **?**) introduce the general Poisson binomial distribution and develop an algorithm to compute its distribution functions.

Now let's take a look of some basic properties of Poisson Multinomial Distributions.

2.2 Properties of the Distribution

Property 1 Given random variable $\mathbf{X}$ that follows a Poisson Multinomial distribution with $\mathbf{P}_{\mathrm{n}\times\mathrm{m}}$, the mean of $\mathbf{X}$ is

$$E(\mathbf{X}) = \boldsymbol{\mu} = (p_{\cdot 1}, \ldots, p_{\cdot,m-1}, p_{\cdot m})'$$

where $p_{\cdot k} = \sum_{i=1}^{n} p_{ik}$.
The variance-covariance matrix of $\mathbf{X}$ is an $m \times m$ matrix $\boldsymbol{\Sigma}$ that has entries $\Sigma_{ij}, i = 1, \ldots, m, j = 1, \ldots, m$ defined as

$$\Sigma_{ij} = \begin{cases} \sum_{k=1}^{n} p_{ki}(1 - p_{ki}) & \text{if } i = j \\ -\sum_{k=1}^{n} p_{ki} p_{kj} & \text{if } i \neq j \end{cases}$$

The CF for the PMD is

$$\phi_{\mathbf{X}}(t_1, \ldots, t_{m-1}) = \phi_{(X_1,\ldots,X_m)'}(t_1, \ldots, t_{m-1}) = \sum_{x_1=0}^{n} \cdots \sum_{x_{m-1}=0}^{n} p(x_1, \ldots, x_{m-1}) \exp\left(\boldsymbol{i} \sum_{j=1}^{m-1} t_j x_j \right).$$

where $\boldsymbol{i} = \sqrt{-1}$. □

The derivations of mean and characteristic function are as trivial as following the definitions. The $\boldsymbol{\Sigma}$ can be calculated by observing that for any fix $i = 1, \ldots, n$, I_{ij} and I_{ik} has covariance $-p_{ij}p_{ik}, j = 1, \ldots, m, k = 1, \ldots, m$. One important thing is that the covariance matrix $\boldsymbol{\Sigma}$ is singular because the elements of $\boldsymbol{X}$ are linear dependent.

Let $\boldsymbol{X}^{\star} = (X_1, \ldots, X_{m-1})'$ with corresponding $\mathbf{P}^{\star}$ equals to the first $m-1$ columns of $\mathbf{P}$, then it has a non-singular $(m-1) \times (m-1)$ covariance matrix $\boldsymbol{\Sigma}_0$. Similar to $\boldsymbol{\Sigma}$, the covariance matrix

$$\mathrm{Var}(\boldsymbol{X}^{\star}) = \boldsymbol{\Sigma}^{\star} = \sum_{i=1}^{n} [\mathrm{Diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i \boldsymbol{p}_i']$$

where $\boldsymbol{p}_i$ is the ith row of $\mathbf{P}^{\star}$. Also,

$$\mathbb{E}(\boldsymbol{X}^{\star}) = \boldsymbol{\mu}^{\star} = (p_{\cdot 1}, \ldots, p_{\cdot,m-1})'$$

We call $\boldsymbol{X}^{\star}$ a reduced PMD of $\boldsymbol{X}$ and $\mathbf{P}^{\star}$ a reduced SPM of $\mathbf{P}$. Respectively, $\boldsymbol{\Sigma}^{\star}$ is the reduced covariance matrix of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}^{\star}$.

Property 2 If the SPM $\mathbf{P}$ can be written as a combination of diagonal matrices $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K$, $\mathbf{P} = \mathrm{Diag}(\mathbf{P}_{\mathrm{k}}), k = 1, \ldots, K$. Then the outcome of $\mathbf{P}$, $\boldsymbol{x}$ can hereby be decomposed in to outcomes of the diagonal matrices, $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\mathrm{K}})$. There will also have random variables $\boldsymbol{X}_{\mathrm{i}} \sim \mathrm{PMD}(\mathbf{P}_{\mathrm{i}})$, respectively. The pmf can computed as the product of the corresponding marginal pmfs. That is

$$\Pr(\boldsymbol{X} = \boldsymbol{x}) = \Pr(\boldsymbol{X}_1 = \boldsymbol{x}_1, \ldots, \boldsymbol{X}_{\mathrm{K}} = \boldsymbol{x}_{\mathrm{K}}) = \prod_{\mathrm{k}=1}^{\mathrm{K}} \Pr(\boldsymbol{X}_{\mathrm{k}} = \boldsymbol{x}_{\mathrm{k}}).$$

□

To show this, we assume $\mathbf{P}$ is n $\times$ m and $\mathbf{P}_{\mathrm{k}}$ is $n_k \times m_k$. $\sum_{k=1}^{K} n_k = n$ and $\sum_{k=1}^{K} m_k = m$. Suppose there are n voters to vote m candidates, certain groups of voters only vote for certain groups of candidates and there are no overlaps . Heuristically, we can separate candidates and voters into independent groups, in each group k, $k = 1, \ldots, K$, voters voting for corresponding candidates described by probability matrix $\mathbf{P}_{\mathrm{k}}$. Strict proof can be done by decomposition of characteristic function of $\mathbf{P}$ into characteristic functions of $\mathbf{P}_{\mathrm{k}}$'s.

3 Computation of The Probability Mass Function

We introduce three methods for computing the pmf, which are the method based on multidimensional discrete Fourier transform ($\mathbf{DFT-CF}$), the normal approximation ($\mathbf{N.A}$) method, and simulation based method($\mathbf{SIM}$). $\mathbf{DFT-CF}$ is an exact method while the other two are approximate methods.

3.1 The DFT − CF Method

Although PMD is of great importance, its pmf is obscure and no straightforward form has been found so far. However, its characteristic function(cf) can be wrote down in an accessible form. The cf is just a Fourier transformation of pmf, thus we can obtain pmf via Fourier transformed cf. Notice the pmf is discrete, so the discrete Fourier transformation will be used. In this section we provide an exact method to compute the pmf of the PMD using multidimensional discrete Fourier transformation. To speed up the computing, we implement Fast Fourier Transformation algorithm($\mathbf{FFT}$).

Suppose a random variable $\boldsymbol{X} = (X_1, \ldots, X_m)' \sim \mathrm{PMD}(\mathbf{P})$, where $\mathbf{P} = (p_{ij}), i = 1, \ldots, n, j = 1, \ldots, m$. Consider $\boldsymbol{X}^\star = (X_1, \ldots, X_{m-1})'$ will be equivalent since the last element of $\boldsymbol{X}$ is redundant. The cf of $\boldsymbol{X}^\star$ is

$$\phi(t_1, \ldots, t_{m-1}) = \mathbb{E}\left[\exp\left(\boldsymbol{i}\sum_{j=1}^{m-1} t_j X_j\right)\right] = \mathbb{E}\left[\exp\left(\boldsymbol{i}\sum_{i=1}^{n}\sum_{j=1}^{m-1} t_j I_{ij}\right)\right]. \tag{1}$$

Here $\boldsymbol{i} = \sqrt{-1}$. We notice

$$\mathbb{E}\left[\exp\left(\boldsymbol{i}\sum_{j=1}^{m-1} t_j X_j\right)\right] = \sum_{x_1=0}^{n} \cdots \sum_{x_{m-1}=0}^{n} p(x_1, \ldots, x_{m-1}) \exp\left(\boldsymbol{i}\sum_{j=1}^{m-1} t_j x_j\right). \tag{2}$$

By definition,

$$\exp\left(\boldsymbol{i}\sum_{j=1}^{m-1} t_j X_j\right) = \exp\left(\boldsymbol{i}\sum_{i=1}^{n}\sum_{j=1}^{m-1} t_j I_{ij}\right) \tag{3}$$

The expectation of RHS of (3) can be expressed as

$$\mathbb{E}\left[\exp\left(\boldsymbol{i}\sum_{i=1}^{n}\sum_{j=1}^{m-1}t_jI_{ij}\right)\right]=\mathbb{E}\left[\exp\left(\boldsymbol{i}\sum_{j=1}^{m-1}t_jI_{1j}+\cdots+\boldsymbol{i}\sum_{j=1}^{m-1}t_jI_{nj}\right)\right].$$

$$=\prod_{i=1}^{n}\mathbb{E}\left[\exp\left(\boldsymbol{i}\sum_{j=1}^{m-1}t_jI_{ij}\right)\right]=\prod_{i=1}^{n}\left[(1-\sum_{j=1}^{m-1}p_{ij})+\sum_{j=1}^{m-1}p_{ij}\exp(\boldsymbol{i}t_j)\right]. \tag{4}$$

We know (4) is equivalent to (2). Therefore we get

$$\sum_{x_1=0}^{n}\cdots\sum_{x_{m-1}=0}^{n}p(x_1,\ldots,x_{m-1})\exp\left(\boldsymbol{i}\sum_{j=1}^{m-1}t_jx_j\right)=\prod_{i=1}^{n}\left[(1-\sum_{j=1}^{m-1}p_{ij})+\sum_{j=1}^{m-1}p_{ij}\exp(\boldsymbol{i}t_j)\right].$$

Let $t_j=\omega l_j$, $l_j=0,\ldots,n$, $\omega=2\pi/(n+1)$. Then the equation becomes

$$\frac{1}{(n+1)^{m-1}}\sum_{x_1=0}^{n}\cdots\sum_{x_{m-1}=0}^{n}p(x_1,\ldots,x_{m-1})\exp\left(\boldsymbol{i}\omega\sum_{j=1}^{m-1}l_jx_j\right)=\frac{1}{(n+1)^{m-1}}q(l_1,\ldots,l_{m-1}), \tag{5}$$

where

$$q(l_1,\ldots,l_{m-1})=\prod_{i=1}^{n}\left[(1-\sum_{j=1}^{m-1}p_{ij})+\sum_{j=1}^{m-1}p_{ij}\exp(\boldsymbol{i}\omega l_j)\right].$$

Note that $q(l_1,\ldots,l_{m-1})$ can be computed directly. The left side of equation (5) is the inverse multi-dimensional discrete Fourier transform of the sequence $p(x_1,\ldots,x_{m-1}), x_i=0,\ldots,n$. Therefore we can apply Multi Dimensional Discrete Fourier Transformation(MD-DFT) on both sides to recover the sequence, the pmf can be obtained as

$$p(x_1,\ldots,x_{m-1})=\frac{1}{(n+1)^{m-1}}\sum_{l_1=0}^{n}\cdots\sum_{l_{m-1}=0}^{n}q(l_1,\ldots,l_{m-1})\exp\left(-\boldsymbol{i}\omega\sum_{j=1}^{m-1}l_jx_j\right) \tag{6}$$

Let $\ell=(l_1,\ldots,l_{m-1})$, then we will have $(n+1)^{m-1}$ different ℓ as l_i values from 0 to n. For example, if we have $n=4$, $m=4$, then ℓ can be $(0,0,0),(0,0,1),\ldots,(4,4,4)$, 125 different vectors in total. Now we have $q(l_1,\ldots,l_{m-1})=q(\ell)$. We design to use these vectors to generate respective $p(x_1,\ldots,x_{m-1})$. For each ℓ, we get a $p(x_1,\ldots,x_{m-1})$.

To speed up the computing, we apply the Fast Fourier transformation (FFT) algorithm from GSL Scientific Library. The FFT algorithm is C language based, we implement it with R and make a new algorithm named $\mathbf{DFT-CF}$ algorithm. Our $\mathbf{DFT-CF}$ algorithm can calculate all values of distribution function as long as we input our $P_{\mathrm{n\times m}}$ matrix.

3.2 Normal-Approximation Based Method

Since covariance matrix of any $\boldsymbol{X} \sim$ PMD is singular, we use the reduced PMD $\boldsymbol{X}^\star$ to establish the normal approximation. From **?**) we know that Central Limit Theory can be applied to PMDs, the following theorem studies the error bound, or converge rate of normal approximation of PMD.

Theorem 1 *For a Poisson-Multinomial random variable* $\mathbf{X} = (X_1, \ldots, X_m)'$ *that has a reduced mean vector* $\boldsymbol{\mu}^\star = (p_{\cdot 1}, \ldots, p_{\cdot,m-1})'$ *and non-singular reduced covariance matrix* $\boldsymbol{\Sigma}^\star$*. For each possible outcome* $\boldsymbol{x}_r$*, and its neighbourhood interval* $\mathcal{N}_{\boldsymbol{x}_r} = [\boldsymbol{x}_r - 0.5, \boldsymbol{x}_r + 0.5]$*. There exists a non-singular matrix* $\boldsymbol{C}$ *such that* $\boldsymbol{\Sigma}^\star = CC'$ *and the error bound of Central Limit Theory approximation is*

$$|P(\mathbf{X} \in \mathcal{N}_{\boldsymbol{x}_r}) - P(\boldsymbol{Z} \in \mathcal{N}_{\boldsymbol{x}_r - \boldsymbol{\mu}^\star})| \leq b(m-1)^{\frac{1}{4}} \sum_{i=1}^{n} \mathbb{E}|C^{-1}\boldsymbol{I}_i|^3$$

where $\boldsymbol{Z}$ *is normal with mean 0 and covariance matrix* $\boldsymbol{\Sigma}^\star$*.*

By central limit theorem (CLT),

$$\left(\frac{\boldsymbol{X}^\star}{n} - \boldsymbol{p}\right) \dot{\sim} \mathrm{N}\left(\mathbf{0}, \frac{1}{n}\boldsymbol{\Sigma}^\star\right).$$

where $\boldsymbol{p} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i$. To show **Theorem1**, notice

$$\boldsymbol{X}^\star = (X_1, \ldots, X_{m-1})' = \sum_{i=1}^{n} \mathbf{I}_\mathbf{i}^\star = \sum_{i=1}^{n} (I_{i1}, \ldots, I_{i,m-1})'$$

Intuitionally, $\mathbf{I}_\mathbf{i}^\star - \mathbb{E}\mathbf{I}_\mathbf{i}^\star$ has mean 0. Hereby $\boldsymbol{X}^\star - \mathbb{E}\boldsymbol{X}^\star = \sum_{i=1}^{n}(\mathbf{I}_\mathbf{i}^\star - \mathbb{E}\mathbf{I}_\mathbf{i}^\star)$ also has mean 0. The covariance of $\boldsymbol{X}^\star$ is $\boldsymbol{\Sigma}^\star = CC'$.

By extended **Berry-Esseen theory** (V. Yu. Bentkus, A Lyapunov-type bound in Rd), there existing a constant c such that

$$|P\left(\mathbf{X} \in \mathcal{N}_{\boldsymbol{x}_i}\right) - P\left(\boldsymbol{Z} \in \mathcal{N}_{\boldsymbol{x}_i - \boldsymbol{\mu}_0}\right)| \leq c(m-1)^{\frac{1}{4}} \sum_{i=1}^{n} \mathbb{E}\left|C^{-1}\boldsymbol{I}_i\right|^3$$

Where $\boldsymbol{Z}$ is **CLT** multivariate normal distribution of with 0 mean and covariance $\boldsymbol{\Sigma}^\star$.

3.3 Simulation-Based Method

One can simulate $\mathbf{I}_i$ from multinomial distribution and then compute $\boldsymbol{X} = \sum_{i=1}^{n} \mathbf{I}_i$. Repeat this many times to generate enough samples for $\boldsymbol{X}$. Then use the sample distribution to approximate the true distribution. To be specific, let $\boldsymbol{x}$ be a given realization of $\boldsymbol{X}$, the probability $\Pr(\boldsymbol{X} = \boldsymbol{x})$ can be calculated by following steps,

Step 1 for each $i = 1, \ldots, n$ randomly generate $\mathbf{I}_i$ once with given $\boldsymbol{p}_i$ using multinomial distribution, calculate $\boldsymbol{X} = \sum_{i=1}^{n} \mathbf{I}_i$.

Step 2 repeat step 1 B times to get $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(\mathrm{B})}$.

Step 3 count the frequency of $\boldsymbol{x}$ showing up in $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(\mathrm{B})}$ as the desired probability.

Consequently, we can calculate each point one by one and eventually the pmf can be calculated by this way. Alternatively, another scheme to do this is by

Step 1 generate $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(\mathrm{B})}$.

Step 2 calculate the frequency of each probability mass point showing up in $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(\mathrm{B})}$ as the pmf.

We design a C++ based algorithm to perform this simulation process. Our algorithm uses the first scheme to calculate probabilities of $\boldsymbol{x}$s as user input and use the second scheme to calculate pmf if $\boldsymbol{x}$ is no specified.

The following theorem studies the expected error bound of the simulation method by given $\mathbf{P}$ and B.

Theorem 2 *Given repeating time B and* $\mathrm{n} \times \mathrm{m}$ *matrix* $\mathbf{P}$*, there will be totally* $N = \binom{n+m-1}{m-1}$ *different results, denote them as* $\boldsymbol{x}_r, r = 1, \ldots, N$*. Let the probability for a specific result* $\boldsymbol{x}_r$ *be* $p_{\boldsymbol{x}_r}$*,* $i = 1, \ldots, N$*. The estimate of* $p_{\boldsymbol{x}_r}$ *using simulation method is* $\widehat{p}_{\boldsymbol{x}_r}$*. We have the following expected error given by Central Limit Theory,*

$$\mathbb{E}|p_{\boldsymbol{x}_r} - \widehat{p}_{\boldsymbol{x}_r}| = \sqrt{\frac{2p_{\boldsymbol{x}_r}(1 - p_{\boldsymbol{x}_r})}{\pi B}}$$

The expected total absolute error,

$$\sum_{r=1}^{N} \mathbb{E}|p_{\boldsymbol{x}_r} - \widehat{p}_{\boldsymbol{x}_r}| \leq \sqrt{\frac{2(N-1)}{\pi B}}$$

For any result $\boldsymbol{x}_r, i = 1, \ldots, N$. Consider a Bernoulli **r.v** Y_r with probability $p_{\boldsymbol{x}_r}$ to be 1, and $1 - p_{\boldsymbol{x}_r}$ to be 0. By repeating the trail for B times, we get **i.i.d** random variables $Y_r^{(1)}, \ldots, Y_r^{(\mathrm{B})}$. By **WLLN**

$$\bar{Y_r} - p_{\boldsymbol{x}_r} \xrightarrow{d} N(0, \sigma^2)$$

where $\sigma^2 = \frac{p_{\boldsymbol{x}_r}(1-p_{\boldsymbol{x}_r})}{B}$. Then the expectation of absolute error for a single $\boldsymbol{x}_r$

$$\mathbb{E}|\bar{Y} - p_{\boldsymbol{x}_r}| = \frac{\sqrt{2}}{\sqrt{\pi}}\sigma = \sqrt{\frac{2}{\pi B}p_{\boldsymbol{x}_r}(1 - p_{\boldsymbol{x}_r})}$$

Let $c = \sqrt{\frac{2}{\pi B}}$ for simplicity, then

$$\begin{aligned}
&\sum_{r=1}^{N} \mathbb{E}|\bar{Y_r} - p_{\boldsymbol{x}_r}| = c\sum_{r=1}^{N} \sqrt{p_{\boldsymbol{x}_r}(1-p_{\boldsymbol{x}_r})} = cN\sum_{r=1}^{N} \frac{\sqrt{p_{\boldsymbol{x}_r}(1-p_{\boldsymbol{x}_r})}}{N} \\
&\leq cN\sqrt{\sum_{r} p_{\boldsymbol{x}_r}(1-p_{\boldsymbol{x}_r})/N} = c\sqrt{N}\sqrt{1-\sum p_{\boldsymbol{x}_r}^2} \\
&\leq c\sqrt{N}\sqrt{1-1/N} = c\sqrt{N-1} \\
&= \sqrt{\frac{2(N-1)}{\pi B}}
\end{aligned}$$

The equality will be achieved if $p_{\boldsymbol{x}_1} = \cdots = p_{\boldsymbol{x}_N}$. For a fixed B,because of the sparsity of PMD, the expected total absolute error goes up that as the dimension of $\mathbf{P}$ goes up. Therefore for higher dimensional $\mathbf{P}$ we need higher B to maintain accuracy regardless of time efficiency.

4 Method Comparisons

In this section, we compare the three methods in terms of numerical accuracy and the time efficiency, we also give out recommendations of under what condition which method will be preferred. At the first three subsections, we show the accuracy of the methods we mention earlier. Here the major accuracy criterion we use is maximum absolute error(MAE). Suppose we have an PMD matrix with $n \times m$ dimension. Hence there will be $(n+1)^{m-1}$ probability mass points notated as a set $X = \{x_1, \ldots, x_N\}$, where $N = (n+1)^{m-1}$. By this way, MAE is defined as following,

$$\text{MAE} = \max_{X} |p(x) - p_{\text{true}}(x)|$$

which is the max value of the differences between probability densities calculated by our methods and true ones. We also use total absolute error (TAE) as a supplemental criterion where

$$\text{TAE} = \sum_{X} |p(x) - p_{\text{true}}(x)|$$

For MD-DFT method, we test its accuracy by using small PMD matrix with given inputs due to the complexity of calculation. For other methods, let the densities calculated by MD-DFT to be the true densities and compare them with our goal method. The PMD matrices we use here are generated randomly.

4.1 Accuracy of MD-DFT Method

As we know already, MD-DFT is a analytic proved method. For large n and $m = 2$, which is the Binomial Passion Distribution, the accuracy is justified by **?**). Thus for convenience and

Table 1: Accuracy of MD-DFT method.

(n, m)	$\min(p_{ij})$	$\max(p_{ij})$	MAE
(2,2)	0.14	0.86	$\leq 10^{-16}$
(4,3)	0.09	0.56	$\leq 10^{-16}$
(5,3)	0.02	0.6	$\leq 10^{-16}$
(6,3)	0.1	0.59	$\leq 10^{-16}$
(4,4)	0.08	0.41	$\leq 10^{-16}$
(5,4)	0.005	0.39	$\leq 10^{-16}$
(6,4)	0.007	0.44	$\leq 10^{-16}$

incapability of calculating the true density of large m, here we show the accuracy of MD-DFT by using given small (n, m) pairs. For those pairs, we can work out their probability densities by hand, and compare them with the results computed by MD-DFT method. As we see from Table 1, the MAE are all less or equal to machine epsilon. This shows us that MD-DFT is accurate enough and the error is only due to the computation limit of devices. Therefore, it is convincing to use MD-DFT as true probability densities in the following subsections.

4.2 Accuracy of Normal Approximation and Simulation Method

Among the three computing algorithms we proposed, one of them is able to calculate the exact probability while the other two are approximation methods. Therefore, it is necessary to explore accuracy of the approximation methods under different circumstances. The metric here we use is MAE and TAE. Probability mass points computed by MD-DFT are considered as true probabilities, denote it as p_{true}. MAE and TAE for normal approximation are given by

$$\text{MAE} = \max_{\boldsymbol{x}} |p_{\text{NA}}(\boldsymbol{x}) - p_{\text{true}}(\boldsymbol{x})|, \quad \text{TAE} = \sum_{\boldsymbol{x}} |p_{\text{NA}}(\boldsymbol{x}) - p_{\text{true}}(\boldsymbol{x})|$$

For simulation method MAE and TAE are given via

$$\text{MAE} = \max_{\boldsymbol{x}} |p_{\text{SIM}}(\boldsymbol{x}) - p_{\text{true}}(\boldsymbol{x})|, \quad \text{TAE} = \sum_{\boldsymbol{x}} |p_{\text{SIM}}(\boldsymbol{x}) - p_{\text{true}}(\boldsymbol{x})|$$

To test the accuracy, we need to generate $\mathbf{P}_{\text{n}\times\text{m}}$ randomly for different settings of (n, m). Also for a fixed (n, m) we repeat the random generation of $\mathbf{P}_{\text{n}\times\text{m}}$ for 5000 times to eliminate the noise effect. As the dimension of $\mathbf{P}$ increases, the distribution gets sparse. As a result, MAE would become smaller no matter what computation method we use. To include the effect of sparsity, we introduce another method called origin. Origin is a method just as simple as estimating all probability mass point with value 0. Thus the TAE will always be 1 and MAE will be the $\max |p_{\text{true}}(\boldsymbol{x})|$. The MAE of origin will show us a good picture for sparsity of PMD distribution as its dimension growth. Comparison between origin and other methods as

shown in Figure 1 will illustrate how effective normal approximation and simulation method are regardless of sparsity.

Figure 1 and Figure 2 exhibit the accuracy test results for different (n, m) pairs. Our computations are done on Linux server with AMD EPYC 7702 chips(128 cores, 2GHz) and 256GB RAM. For simulation method, the repeating time are set to be 10^4 and it is easy to anticipate that with the growth of repeating time the accuracy of the method will be better. Also it is not necessary to test with large repeating time because of the computation capacity limits. Due to the computation limits of device, we are able to test (n, m) pairs with $(n+1)^{m-1}$ less than 2^{31}.

Figure 1 shows accuracy comparisons in a perspective of MAE. It can be seen that when m is small(less than 5), normal approximation is barely better than origin method as their curves are almost overlapped, it means the accuracy of normal approximation method is no much better than estimating all probability mass points with 0. While m is larger or equal to 5, normal approximation is sufficiently better than simulation method regardless of sparsity. Simulation method under this situation is slightly exceeding origin.

In Figure 2, when m is smaller than 5 and using TAE as y axis metric, the dashed line that represents simulation method is beneath the red line which stands for normal approximation. The gap between two curves are significant. When m is larger than 4. Normal approximation outperforms simulation method. As n grows, the differences between two methods get larger.

To conclude, in the concern of accuracy simulation method is a much better choice for user when m is less than 5. Conversely, normal approximation method is considerably more accurate than simulation method if m is larger than 4.

4.3 Time Efficiency of DFT-CF method

In this section, we show computation efficiency of simulation method and MD-DFT for calculating all probability mass points as (n, m) gets large. We generate random matrices with respect to given (n, m), record the calculating time of each method. As for normal approach, it is a asymptotic method, we only need to calculate the asymptotic normal distribution.

Study the computing of the three methods.

4.4 Recommendations

For small m and moderate n, the MD-DFT can be used.

For large m and moderate n, the simulation-based method can be used.

For large n, the NA-based method can be used.

Table 2: Accuracy of simulation method.

(n, m)	MD-DFT	Simulation based (10^6)
(4, 3)	0.023	0.542
(6, 3)	0.002	0.852
(8, 3)	0.003	1.177
(10, 3)	0.003	1.634
(20, 3)	0.014	4.532
(50, 3)	0.098	22.349
(100, 3)	0.49	80.243
(200, 3)	2.834	307.350
(1000, 3)	249.449	7992.507

5 Applications

5.1 Calculation of Voting Probability

Do some example like this:

https://stats.stackexchange.com/questions/274211/calculating-the-probability-of-someone-winning-from-a-poll

In voting scenarios, people always pay attention to the election result. The most thing we usually care about is who will win the election and how many chances each candidate has to win the election. A Poisson Multinomial distribution can fit the situation perfectly under some assumptions.

Suppose a election has n voters and m candidates, there will be $N = \binom{n+m-1}{m-1}$ different results $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, respectively. Assume we know the $\mathbf{P}$ based on prior polls, for example, we can always estimate the approval rate of each candidate in a certain constituency from the monthly polls or exit polls. Then we are able to compute the notional result.

To demonstrate that, suppose we have ten electoral voters and three candidates with $\mathbf{P}$ matrix with means of column one, two and three are $0.3631, 0.3405$ and 0.2964, the first five rows are as following,

$$\mathbf{P} = \begin{pmatrix} 0.071 & 0.589 & 0.340 \\ 0.365 & 0.195 & 0.440 \\ 0.445 & 0.505 & 0.050 \\ 0.353 & 0.382 & 0.265 \\ 0.620 & 0.111 & 0.269 \end{pmatrix}$$

To compute the probability of each candidate winning the election, just need to introduce constraints. For instance, under the constraint $\chi_1 = \{x_1 > x_2\} \cap \{x_1 > x_3\}$, we are able to

compute the winning rate of the first candidate is

$$p(\text{the 1st candidate wins}) = \sum_{\boldsymbol{x}\in\chi_1} p(\boldsymbol{x}) = 0.3429$$

Similarly, the probabilities for the second candidate and the third candidate to win are 0.2745 and 0.2001. Additionally, the most possible result is $\boldsymbol{x} = (4, 3, 3)$, which has probability 0.08546.

5.2 Statistical Inference for Aggregated Data

First introduce the out "Logistic-like" model to fit a given aggregated dataset with selected features and a categorical response variable that has m categories. By dividing the rows of our data into H groups $G_1, \ldots, G_S$, the group size of each group is $s_i, i = 1, \ldots, S$. Each s_i are positive integer but not necessarily to be equal. Let the quantity for each category of group G_i be $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_m^{(i)})'$, m is the category number. Denote $\boldsymbol{H}^{(i)}$ to be the covariate matrix of G_i, then $\boldsymbol{H}^{(i)} = (\mathbf{1}, \boldsymbol{h}_1^{(i)}, \ldots, \boldsymbol{h}_{s_i}^{(i)})'$ is a $s_i \times v$ matrix with first column being $\mathbf{1}$, where v equals to the number of covariates plus one. Let $\mathbf{P}^{(i)} = (p_{jk}^{(i)})$ be the SPM for group G_i, $i = 1, \ldots, S$, $j = 1, \ldots, s_i$ and $k = 1, \ldots, m$. Let the probability of getting $\boldsymbol{x}^{(i)}$ for group G_i be $p(\boldsymbol{x}^{(i)})$. The total log-likelihood for all groups can be computed as

$$\ell = \sum_{i=1}^{S} \ell_i = \sum_{i=1}^{S} \log p(\boldsymbol{x}^{(i)})$$

Where $p(\boldsymbol{x}^{(i)})$ can be computed via Poisson Multinomial distribution with SPM $\mathbf{P}^{(i)}$.
Set category m as baseline and use softmax function to form the $\mathbf{P}^{(i)}$ for each group through parameter $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{m-1})$ as

$$p_{jk}^{(i)} = \frac{\exp\left(\boldsymbol{h}_j^{(i)}\boldsymbol{\beta}_k\right)}{1 + \sum_{k=1}^{m-1} \exp\left(\boldsymbol{h}_j^{(i)}\boldsymbol{\beta}_k\right)} \quad k \neq m \quad \text{and} \quad p_{i,m}^{(i)} = \frac{1}{1 + \sum_{k=1}^{m-1} \exp\left(\boldsymbol{h}_j^{(i)}\boldsymbol{\beta}_k\right)}$$

Then we are able to estimate our parameters on the direction of minimizing total log-likelihood and finally get our estimate $\widehat{\mathbf{P}}^{(i)}$.
In the rest of this part, we apply the model to the dataset "ai4i". The dataset "ai4i" is a synthetic machine failure dataset that reflects real predictive maintenance data encountered in industry. The data consists of 10000 products(rows) and 14 features(columns) including Product type, Air temperature, Process temperature, Rotational speed and others.

For demonstration purpose, here we only use the first 1000 rows and Product type as response variable, Air temperature and Process temperature as covariates. The feature Product type is categorical and has three levels, "M", "L", "H", we denote them as category 1, 2, 3

for simplicity. The number of products that fall in category 1, 2, and 3 are 285, 601, and 114. The other two features are continuous.

We randomly divide the dataset into 100 groups $G_1, \ldots, G_{100}$, the smallest group has size of three rows and the largest one has 18 rows. Note that readers can use other criterion to divide the dataset by their own. Notice the covariate number is three including intercept and the response has three categories, thus the dimension of our parameter matrix $\boldsymbol{\beta}$ will be 3×2 if we set category 3 as baseline.

The estimates of the parameters are

$$\widehat{\boldsymbol{\beta}} = \begin{pmatrix} 1.07484986 & 2.2820922 \\ 1.62342045 & 1.9108976 \\ -0.06277732 & -0.8455964 \end{pmatrix}$$

The corresponding $\widehat{\mathbf{P}}^{(1)}$ and $\widehat{\mathbf{P}}^{(5)}$ for group 1 and group 5 are

$$\widehat{\mathbf{P}}^{(1)} = \begin{pmatrix} 0.11914 & 0.63310 & 0.24776 \\ 0.12326 & 0.57268 & 0.30406 \\ 0.14809 & 0.47560 & 0.37631 \\ 0.14504 & 0.56971 & 0.28525 \\ 0.12451 & 0.54095 & 0.33454 \\ 0.04170 & 0.55944 & 0.39886 \\ 0.03559 & 0.53568 & 0.42873 \\ 0.04890 & 0.54668 & 0.40442 \end{pmatrix}, \quad \widehat{\mathbf{P}}^{(5)} = \begin{pmatrix} 0.15604 & 0.70766 & 0.13630 \\ 0.14469 & 0.73976 & 0.11555 \\ 0.11246 & 0.67372 & 0.21382 \\ 0.12036 & 0.64885 & 0.23079 \\ 0.11263 & 0.61304 & 0.27433 \\ 0.11563 & 0.55029 & 0.33408 \\ 0.11774 & 0.61672 & 0.26554 \\ 0.15071 & 0.52510 & 0.32419 \\ 0.13878 & 0.56645 & 0.29477 \\ 0.08194 & 0.54561 & 0.37245 \\ 0.06307 & 0.58191 & 0.35502 \end{pmatrix}$$

For group 1 and 5, $\boldsymbol{x}^{(1)} = (1, 5, 2)'$ and $\boldsymbol{x}^{(5)} = (2, 6, 3)'$, so $p(\boldsymbol{x}^{(1)}) = 0.070$, $p(\boldsymbol{x}^{(5)}) = 0.067$.

5.3 Uncertainty Quantification in Classification

Confusion matrix

Use Theorem for independent.

In machine learning classification problem with multiple labels, for each unit in the test set, the probability of the unit belongs to each class is computed. Usually, the predicted class is assigned as the highest probability. In the classifiers (i.e., soft classification), the unit class is randomly assigned according to the predicted probabilities, leading to randomness in the confusion matrix. The PMD can be used to characterize the distribution of the counts in the confusion matrix.

In this section, we consider an Electroluminescence (EL) image classification example to illustrate the usage of PMD in machine learning classification problems. In the photovoltaic (PV) reliability study, the EL image is an important data type that reveal information about

the PV health status. Because disconnected parts do not irradiate, the darker areas in EL images indicate defective cells. The EL imaging provide visual inspection of solar panels and is a non-destructive technology for failure analysis of PV modules (**?**).

The work of **?**),**?**), and **?**) provide a public dataset of solar cells extracted from high resolution EI images of PV modules (`https://github.com/zae-bayern/elpv-dataset`). In total there are 2624 images. All images are preprocessed with respect to size and are eliminated distortion induced by the camera lens used to capture the EL images. Each image is manually labeled with its degree of defectiveness. The degree of defectiveness is determined by two questions. The first is how do evaluators think the status of the solar cells, functional or defective; the second is how they are confident about their assessments. In total there are four labels as shown in Table 3 and we marked them as Class A-D.

Table 3: Partitioning of solar cells into functional and defective, with an additional self-assessment on the rater's confidence after visual inspection. Non-confident decisions obtain a weight lower than 100% for the evaluation of the classifier performance

Condition	Confident?	Label p	Weight w	Class
functional	✓	functional	0%	A
	✗	defective	33%	B
defective	✗	defective	67%	C
	✓	defective	100%	D

We split our data to training data (80%) and test data (20%), then train a CNN model on the training set. In CNN model, we use Relu activation function and set the kernel size 3×3 and stride 1×1. For each image in the testing set, the model provides a probability that the prediction belongs to each class. Table 4 provides a subset of the CNN model output. For true class for the first sample in Table 4 is A. If we predict the class of the first sample using the highest probability, then the prediction is A and there's no randomness. If we allow the model to make predictions based on the probability vector as shown in the first row then there are randomness in the confusion matrix. We can consider the confusion matrix to follow a PMD distribution then we can quantify the uncertainty in confusion matrix using PMD.

Figure 1 shows the marginal probability of each cell. For example, in the first row first column panel cell in Figure 1, we know the possible counts that the model correctly predict class A as well as the corresponding probability. In this way, we have a uncertainty quantification in the confusion matrix.

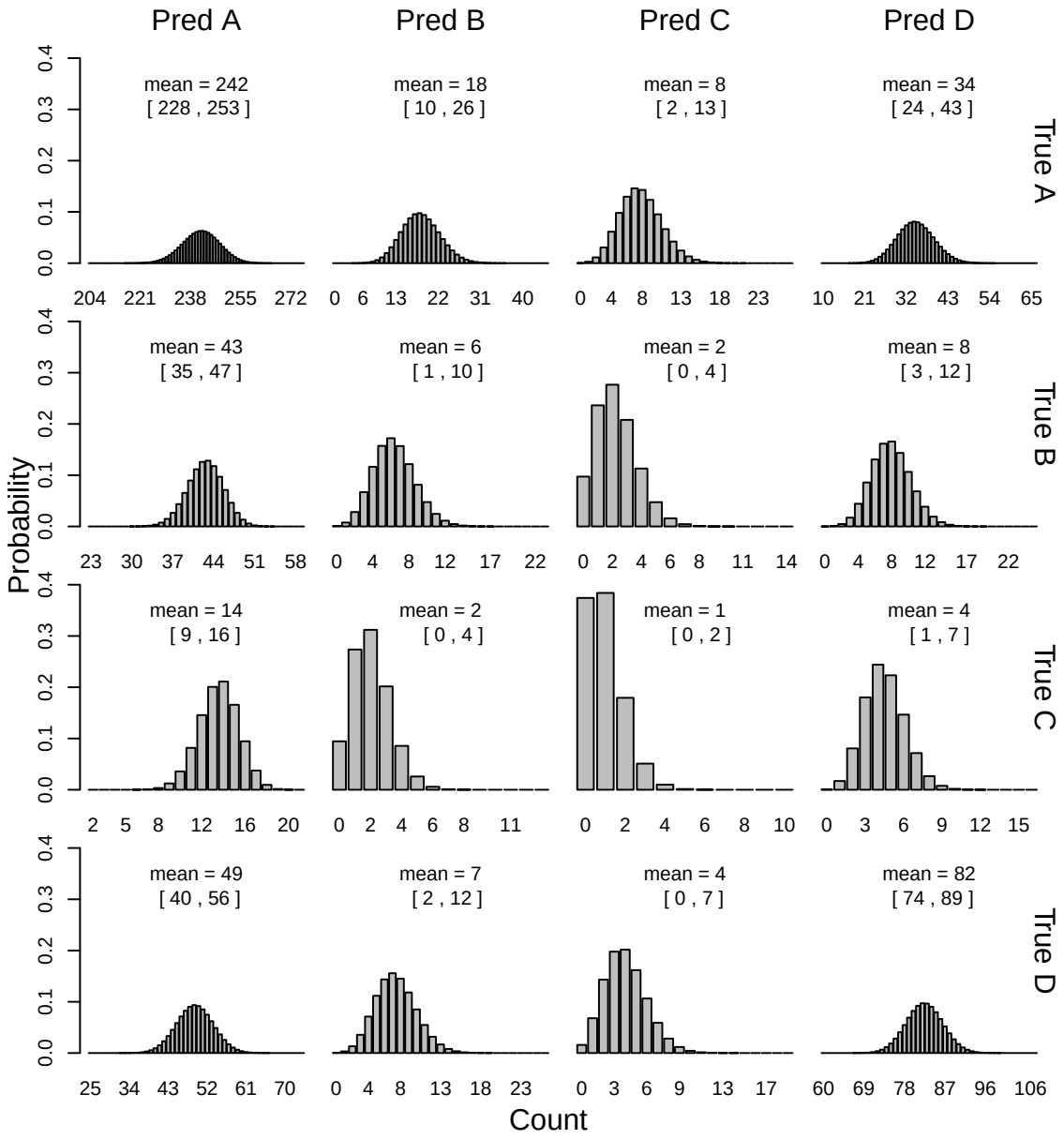

Figure 1: The barplot shows the confusion matrix prediction. For each panel cell, the plot shows the corresponding prediction counts fall in this cell as well as their probability. We also present the mean and 95% naive prediction interval.

Table 4: An example of the probability vectors of a subset in testing set from the trained CNN model.

	A	B	C	D
1	0.9230	0.0366	0.0107	0.0297
2	0.0736	0.0802	0.0513	0.7950
3	0.0000	0.0016	0.0006	0.9978
4	0.9170	0.0537	0.0062	0.0231
5	0.9579	0.0239	0.0070	0.0112
6	0.8991	0.0347	0.0132	0.0530

6 Illustrations of the R Package

6.1 Examples

Illustrate the use of major functions in the R package.

6.2 Benchmark of R Packages for Poisson Binomial Distribution

7 Conclusions and Areas for Future Research

We develop algorithm that can be useful for computing the pmf of the PMD distribution, which is challenging to compute but useful in many application scenarios.