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

August 30, 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 (Cheng, Diakonikolas, and Stewart 2017), digital imaging (Akter, Moon, and Kwon 2019), machine learning (Kamath 2014). One of the popular areas that involved PMD is voting theory. A very simple example is, suppose we have n independent voters to vote for a president from m candidates, each voter has its own chance to vote on a certain candidate. It is not our concern how voters vote, we care only the voting result, that is how many votes each candidate gets. An (n, m) PMD is a perfect model for this example. 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. When (n, m) is small it is possible to calculate the distribution function by using enumeration method. However when n, m get large, we need better method, or algorithm to calculate it.

1.2 Related Literature and Contribution of This Work

Ecological inference

https://www.pnas.org/content/96/19/10578

Some former studies have uncover PMD's structure and properties, Diakonikolas, Kane, and Stewart (2016) shows the Fourier transformation of PMD is sparse and provide a theorem that there exist algorithms to calculate PMD's density. Daskalakis, Kamath, and Tzamos (2015) also prove us PMD is ϵ-cover and Central Limit Theory is valid for PMD. Other papers such as Akter, Moon, and Kwon (2019) shows us some interesting application of PMD and its sparsity property. Due to the huge practical value of PMD, computing its pmf is of great importance. However, there is no available algorithm for computing pmf for PMD. In this case, we are motivated to develop a thorough method to calculate its pmf.

We design three methods to calculate it,

MD-DFT(base on multi-dimensional discrete Fourier transform(MD-DFT)) method: we use Multidimensional Fast Discrete Fourier Transformation algorithm to calculate PMD's exact probability mass function.

Simulation based method: we use multinomial distributions to simulate the process of voting and use the frequency as our result to approximate true result.

Normal approximation based method: when n is large, we apply CLT and use a approximated normal distribution as our result.

We also compare the time efficiency and accuracy of each method, and come up a guide of the best situation for each method. A R package is developed to implement these three methods, users can call certain function to calculate user given (n, m) PMD pmf's.

1.3 Overview

The rest of the paper is organized as follows.

In the coming part, we describe the definition of Poisson Multinomial distribution as well as demonstrate some useful properties. In the third part, we explain three ways of computing the probability mass function by details.

The fourth part of this article compare the accuracy and time efficiency of the listed three methods.

In the application part, we begin with a simple scenario of committee voting where small dimensional PMDs can be applied. Later we show a way of implementing medium size PMDs to AI4I 2020 Predictive Maintenance Dataset which is a large scale aggregated dataet(large n). We also place PMD in a classification scenario using the solar cells EI image data.

2 Poisson Multinomial Distribution

2.1 Definition of the Distribution

Let $(I_{i1}, \ldots, I_{i,m})'$, $i = 1, \ldots, n$ be a sequence of independent indicator vectors where exactly one of m categories successes. That is there is one and only one of those I_{ij}, $j = 1, \ldots, m$ can take value one for each i. Take election as an example, suppose there are several candidates, denoted as $j = 1, \ldots, m$, and some voters, voters are independent to each other, denoted as $i = 1, \ldots, n$, every voter can only vote for one candidate, if the ith voter votes for jth candidate, then $I_{ij} = 1$, else $I_{ij} = 0$. Denote $p_{ij} = \Pr(I_{ij} = 1)$, as the success probability of jth candidate gets a vote from ith voter. The Poisson multinomial random variable $(X_1, \ldots, X_m)'$ is defined as the sum of n independent and non-identical distributed indicator vectors $(I_{i1}, \ldots, I_{i,m})'$, which is the result of election, the total votes each candidate gets. Here $X_j = \sum_{i=1}^{n} I_{ij}$. Assume all probabilities p_{ij}'s are known, we care about the distribution of our election result, which is the probability of a certain result. Because of the sum constraints that, $\sum_{j=1}^{m} I_{ij} = 1$, $\sum_{j=1}^{m} p_{ij} = 1$, and $\sum_{j=1}^{m} X_j = n$, one can drop the last random variable when the focus is on the distribution of $(X_1, \ldots, X_{m-1})'$. That is we can focus on the random vectors $\mathbf{I}_i =$

$(I_{i1}, \ldots, I_{i,m-1})'$ and $\mathbf{X} = (X_1, \ldots, X_{m-1})'$ without the loss of generality. In this section let's still consider $\mathbf{X} = (X_1, \ldots, X_m)'$ and $\mathbf{I}_i = (I_{i1}, \ldots, I_{im})'$ for the purpose of introduction. The distribution of the $\mathbf{X}$ is called the Poisson multinomial distribution (PMD), which is denoted by

$$\mathbf{X} = \sum_{i=1}^{n} \mathbf{I}_i \sim \text{PMD}(\mathbf{P}_{\text{n}\times\text{m}}),$$

where the success probability matrix (SPM) is

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

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,

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

is of interest. Moreover, in the next section, we will focus on $p(x_1, \ldots, x_{m-1})$ because $x_m = n - \sum_{i=1}^{m-1} x_i$ for development of our computation methods.

Let's see a simple example, we have three candidates and four voters, the result of voting is a random variable $\mathbf{X} \sim \text{PMD}(\mathbf{P})$ where

$$\mathbf{P}_{4\times3} = \begin{pmatrix} 0.1 & 0.2 & 0.7 \\ 0.5 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.1 \\ 0.8 & 0.1 & 0.1 \end{pmatrix}.$$

It is trivial to compute the probability mass function of $\mathbf{X}$ through enumeration. For instance, the probability of a result that candidate 1 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

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

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.

Note that when the SPM is identical across all rows, that is, 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 Hong (2013a).

In related literature, Hong (2013a) consider the exact and approximate methods for computing the pmf of the Poisson binomial distribution. Zhang, Hong, and Balakrishnan (2018) 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

Theorem 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* $\boldsymbol{\mu} = (p_{\cdot 1}, \ldots, p_{\cdot,m-1}, p_{\cdot m})'$*, where* $p_{\cdot k} = \sum_{i=1}^{n} p_{ik}$*.*
The variance-covariance matrix of $\mathbf{X}$ *is a* $m \times m$ *matrix* $\boldsymbol{\Sigma}$ *that has entries* Σ_{ij} *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 the above properties are trivial. One important thing is that we realize the covariance matrix $\boldsymbol{\Sigma}$ is singular, but we still can get a non-singular $n \times (m-1)$ matrix, say $\boldsymbol{\Sigma}_0$, to given $\mathbf{P}_{\mathrm{n}\times\mathrm{m}}$ by deleting the last column of $\boldsymbol{\Sigma}$ due to the fact that the last column is linear dependent on the first $m-1$ columns. Similarly, the last element of the expectation vector of X is also redundant since it equals to n minus summation of the first $m-1$ elements. Therefore we can denote $\mathbb{E}(X)$ by replacing $\boldsymbol{\mu}$ to $\boldsymbol{\mu}_0 = (p_{\cdot 1}, \ldots, p_{\cdot,m-1})'$.

Theorem 2 *If the SPM,* $\mathbf{P}$ *can be written as a combination of diagonal matrices* $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K$*,* $\mathbf{P} = Diag(\mathbf{P}_{\mathrm{k}}), k = 1, \ldots, K$*. Then the outcome of* $\mathbf{P}$*,* $\boldsymbol{x}$ *can hereby be decomposed in to outcomes of the diagonal matrices,* $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\mathrm{K}})$*. The pmf can computed as the product of the corresponding marginal pmfs. That is*

$$p(\boldsymbol{x}|\mathbf{P}) = p(\boldsymbol{x}_1 \cap \cdots \cap \boldsymbol{x}_{\mathrm{K}}|\mathbf{P}_1, \ldots, \mathbf{P}_K) = \prod_{k=1}^{K} p(\boldsymbol{x}_k|\mathbf{P}_{\mathrm{k}}).$$

To show this, we assume $\mathbf{P}$ is n × m and $\mathbf{P}_{\mathrm{k}}$ is $n_k \times m_k$. $\sum_{k=1}^{K} n_k = n$ and $\sum_{k=1}^{K} m_k = m$. To demonstrate the situation here, 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 multi-dimensional discrete Fourier transform (MD-DFT), the normal approximation (NA) method, and simulation based method.

3.1 The MD-DFT Method

In this section we provide an exact formula to compute the pmf of the PMD. The formula is based on the characteristic function (CF) of the PMD and the MD-DFT.

Multidimensional Fourier Transform

The CF of $\mathbf{X} = (X_1, \ldots, X_{m-1})'$ is

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

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

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

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

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

Therefore we get

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

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

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

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 (1) 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 MD-DFT on both sides to recover the sequence, we obtain the pmf as

$$p(x_1, \ldots, x_{m-1}) = \frac{1}{(n+1)^{m-1}} \sum_{l_1=0}^{n} \cdots \sum_{l_{m-1}=0}^{n} q(l_1, \ldots, l_{m-1}) \exp\left(-\boldsymbol{i}\omega \sum_{j=1}^{m-1} l_j x_j\right) \tag{2}$$

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 get all $p(x_1, \ldots, x_{m-1})$ with respect to given n, $m-1$ and $P_{n\times(m-1)}$, we apply the fast Fourier transformation (FFT) algorithm from GSL Scientific Library to make calculation efficient. This FFT algorithm is C language based, we implemented it and wrote a new MD-DFT algorithm. Our MD-DFT algorithm can calculate all values of distribution function as long as we input our $P_{n\times(m-1)}$ matrix, and it can be called from R.

3.2 Normal-Approximation Based Method

Let $\boldsymbol{p}_i$ be the ith row of the SPM P. Let $\boldsymbol{p} = \sum_{i=1}^{n} \boldsymbol{p}_i/n$ be the average of the rows of $\mathbf{P}$.

Theorem 3 *For a Poisson-Multinomial random variable $\mathbf{X} = (X_1, \ldots, X_{m-1})'$ with mean $\boldsymbol{\mu}_0$ and non-singular covariance matrix $\boldsymbol{\Sigma}_0$. For each outcome $\boldsymbol{x}_i$, and its neighbourhood interval $\mathcal{N}_{\boldsymbol{x}_i} = [\boldsymbol{x}_i - 0.5, \boldsymbol{x}_i + 0.5]$. There exists a non-singular matrix $\boldsymbol{C}$ such that $\boldsymbol{\Sigma}_0 = CC'$ and the error bound of Central Limit Theory approximation is*

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

where $\boldsymbol{Z}$ is normal with mean 0 and covariance matrix $\boldsymbol{\Sigma}_0$.

We first show that CLT can be applied to

$$\frac{\mathbf{X}}{n} - \boldsymbol{p} = \frac{(X_1, \ldots, X_{m-1})}{n} - \boldsymbol{p} = \frac{(X_1 - \sum_{i=1}^n p_{i1}, \ldots, X_{m-1} - \sum_{i=1}^n p_{i,m-1})}{n}$$

We just need to show that for any $j = 1, \ldots, m-1$, $X_j - \sum_{i=1}^n p_{ij}$ satisfies conditions of CLT. Where $X_j = \sum_i I_{ij}$
Notice that for any $\delta > 0$

$$1 \geq p_{ij}(1 - p_{ij}) = \mathrm{Var}(I_{ij}) \geq E(|I_{ij}|^{2+\delta})$$

Therefore, let $s_n^2 = \sum_{i=1}^n \mathrm{Var}(I_{ij})$, we have

$$\frac{1}{s_n^{2+\delta}} \sum_{i=1}^n E|I_{ij}|^{2+\delta} \leq \frac{1}{s_n^{2+\delta}} \sum_{i=1}^n \mathrm{Var}(I_{ij}) = \frac{1}{s_n^{\delta}}$$

As $n \to \infty$, $s_n \to \infty$, so the above equation converges to 0. Therefore X_j satisfies Lyapunov condition, thus CLT can be applied to X_j.
To compute the covariance matrix for $\mathbf{X}$, observe that for any fix i, I_{ij} and I_{ik} has covariance $-p_{ij}p_{ik}$. Therefore, it is trivial to get the covariance matrix

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

By central limit theorem (CLT),

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

To show **Theorem3**, we notice

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

Easy to get $\boldsymbol{I}_i - \mathbb{E}\boldsymbol{I}_i$ has mean 0. Hereby $\mathbf{X} - \mathbb{E}\mathbf{X} = \sum(\boldsymbol{I}_i - \mathbb{E}\boldsymbol{I}_i)$ also has mean 0. The covariance of $\mathbf{X}$ is $\boldsymbol{\Sigma} = CC'$.
By extended **Berry-Esseen theory** (V. Yu. Bentkus, A Lyapunov-type bound in Rd), there existing a constant b such that

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

Where $\boldsymbol{Z}$ is **CLT** multivariate normal distribution of with 0 mean and covariance $\boldsymbol{\Sigma}_0$.
When n is sufficiently large, our R package use the normal distribution to calculate our probability by demand of users.

3.3 Simulation-Based Method

One can simulate I_i from multinomial distribution and then compute X. Repeat this many times to generate enough samples for X. Then use the sample distribution to approximate the true distribution. To be specific,

Step 1 randomly generate I_i with given p_i using multinomial distribution.

Step 2 repeat step 1 to generate $I_1, \ldots, I_n$. Calculate $X = (X_1, \ldots, X_m)'$, where $X_j = \sum_{i=1}^{n} I_{ij}, j = 1, \ldots, m$.

Step 3 repeat step 1 and step 2 for T times, and calculate the frequency of X as to form our distribution density.

Example 1 *Suppose we are given* $n = 3$, $m = 2$

$$P_{\mathrm{n\times m}} = P_{3\times 1} = \begin{pmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \\ 0.3 & 0.7 \end{pmatrix}.$$

We first generate I_1 *by using multinomial distribution with respect to probability vector* $(0.1, 0.9)$, *we get* $I_1 = (0, 1)$ *,then we generate* I_2 *using multinomial distribution with probability vector* $(0.5, 0.5) \to I_2 = (1, 0)$, *and also generate* I_3 *by using prob vector* $(0.3, 0.7) \to I_3 = (0, 1)$. *Now we have a simulation result* $X = I_1 + I_2 + I_3 = (1, 2)$. *We repeat above process* 10^4 *times to get* 10^4 *results, finally if we want to know probability of (1,2), we can calculate the frequency of it in the* 10^4 *results.*

We design a C based algorithm to perform this simulation process. Our algorithm automatically calculate probabilities of all possible results and it can be called from R.

In our package, as n and m get large, the total number of probability mass points $(n+1)^{m-1}$ will be extensively great. For time efficiency purpose, we only calculate the probability as customer demand. User will be asked to input the resulting vector, and the probability for the vector will be calculated.

Theorem 4 *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* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. *Let the probability for a specific result* $\boldsymbol{x}_i$ *be* $p_{\boldsymbol{x}_i}$, $i = 1, \ldots, N$. *The estimate of* $p_{\boldsymbol{x}_i}$ *using simulation method is* $\widehat{p}_{\boldsymbol{x}_i}$. *We have the following expected error given by Central Limit Theory,*

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

The expected total absolute error(expected TAE) will be smaller or equal to $\sqrt{\frac{2(N-1)}{\pi B}}$.

For any result $\boldsymbol{x}_i, i = 1, \ldots, N$. Consider a Bernoulli **r.v** Y_i with probability $p_{\boldsymbol{x}_i}$ to be 1, and $1 - p_{\boldsymbol{x}_i}$ to be 0. By repeating the trail for B times, we get **i.i.d** random variables $Y_1, \ldots, Y_B$. By **WLLN**

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

where $\sigma^2 = \frac{p_{\boldsymbol{x}_i}(1-p_{\boldsymbol{x}_i})}{B}$. Trivally, we get the expectation of absolute error for a single $\boldsymbol{x}_i$

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

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

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

The equality will be achieved if $p_{\boldsymbol{x}_1} = \cdots = p_{\boldsymbol{x}_N}$.

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

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	$\leq 10^{-16}$

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 Possion Distribution, the accuracy is justified by Hong (2013b). Thus for convenience and 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

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

For simulation method MAE and TAE are given via

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

To test the accuracy, we need to generate $\mathbf{P}_{\mathrm{n\times m}}$ randomly for different settings of (n, m). Also for a fixed (n, m) we repeat the random generation of $\mathbf{P}_{\mathrm{n\times m}}$ 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 $\max|p_{\mathrm{true}}(\boldsymbol{x})|$. 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.

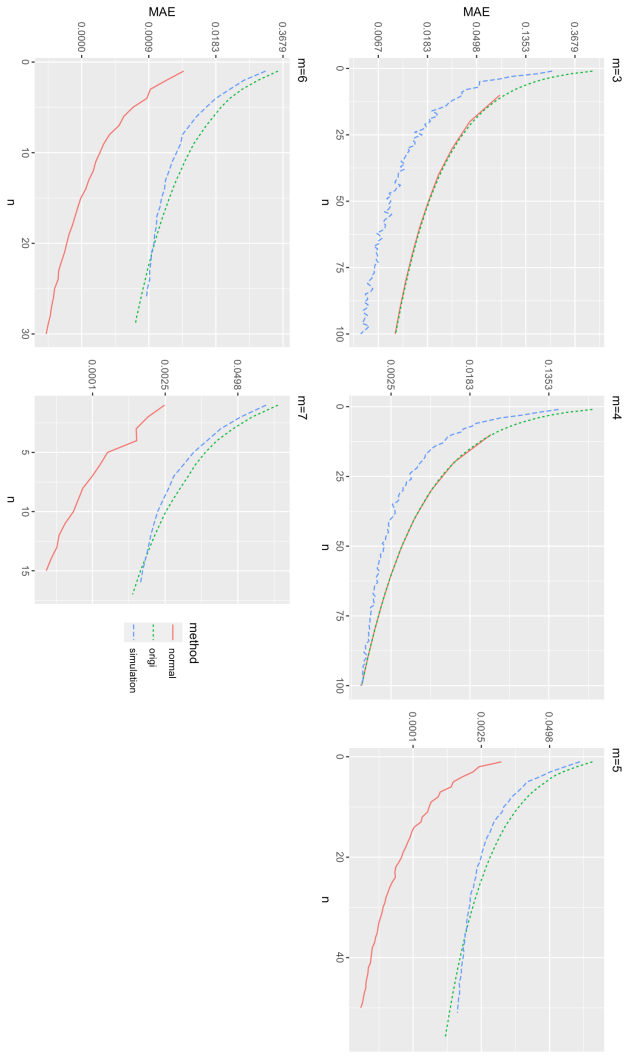

Figure 1: MAE given different (n, m) settings

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}.

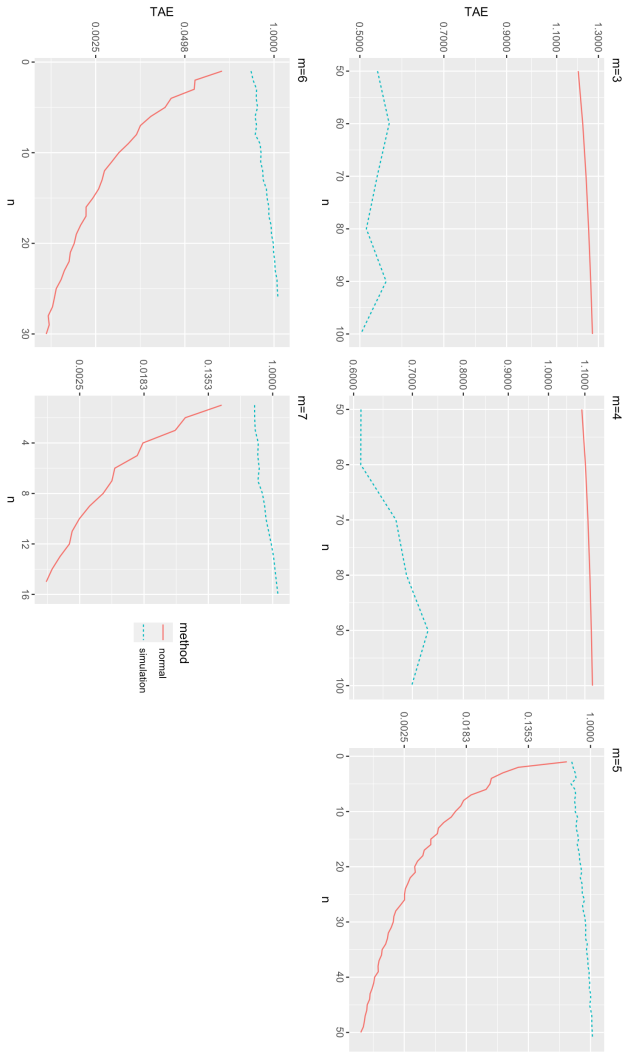

Figure 2: MAE and giving different (n, m) settings

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

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

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

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.

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

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

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 (Deitsch et al. 2019).

The work of Deitsch et al. (2019),Buerhop-Lutz et al. (2018), and Deitsch et al. (2021) 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 3 shows the marginal probability of each cell. For example, in the first row first column panel cell in Figure 3, 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.

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.

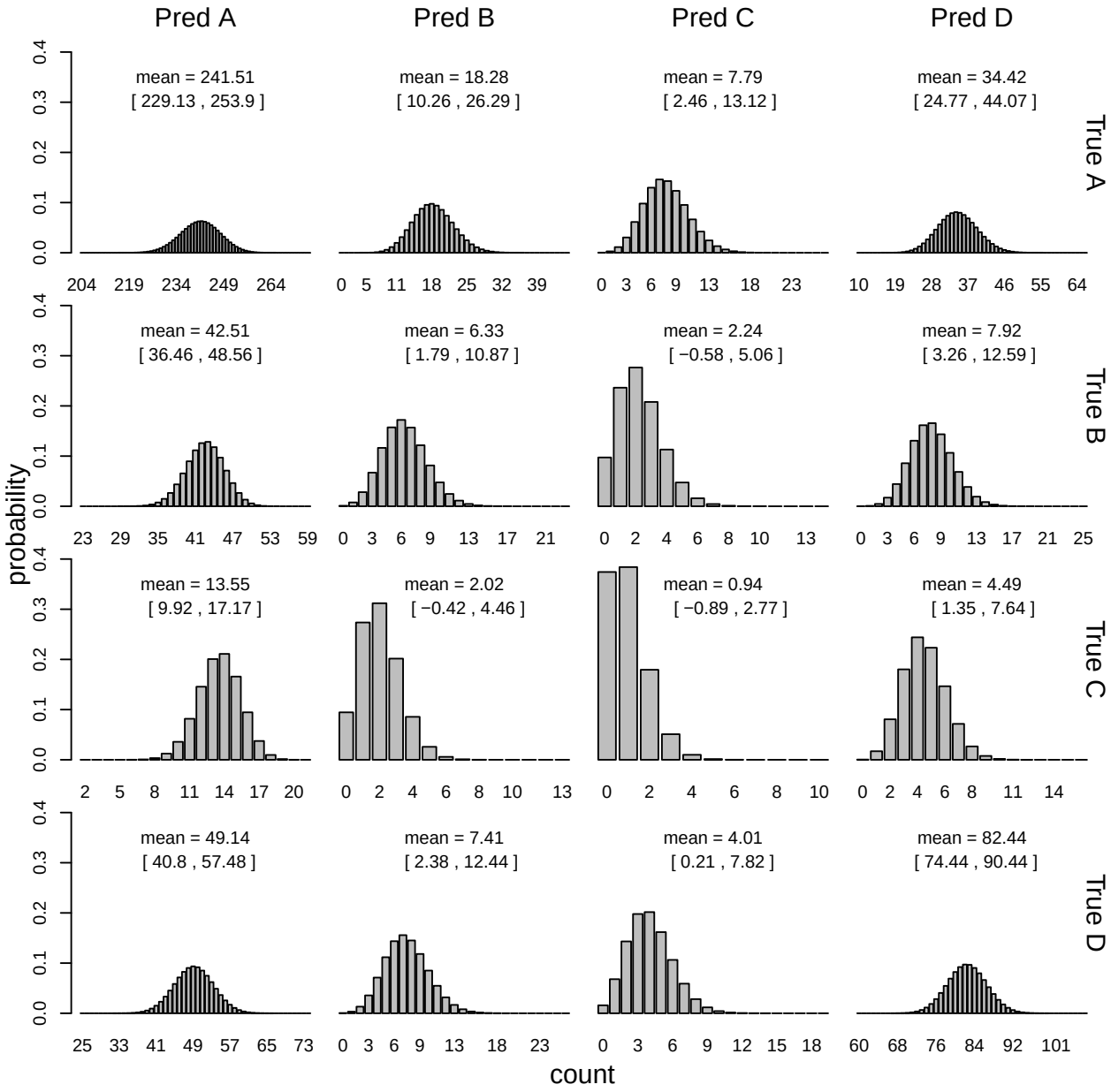

Figure 3: The barplot shows the confusion matrix prediction. For each panel cell, the plot shows the corresponding prediction counts fall in this cell as well as their probability. We also present the mean and 95% confidence interval for prediction.

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.

References

Akter, L. A., I. Moon, and G.-R. Kwon (2019). Double random phase encoding with a poisson-multinomial distribution for efficient colorful image authentication. *Multimedia Tools and Applications 78*(11), 14613–14632.

Buerhop-Lutz, C., S. Deitsch, A. Maier, F. Gallwitz, S. Berger, B. Doll, J. Hauch, C. Camus, and C. J. Brabec (2018). A benchmark for visual identification of defective solar cells in electroluminescence imagery. In *European PV Solar Energy Conference and Exhibition (EU PVSEC)*.

Cheng, Y., I. Diakonikolas, and A. Stewart (2017). Playing anonymous games using simple strategies. In *SODA*.

Daskalakis, C., G. Kamath, and C. Tzamos (2015). On the structure, covering, and learning of poisson multinomial distributions. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, 1203–1217.

Deitsch, S., C. Buerhop-Lutz, E. Sovetkin, A. Steland, A. Maier, F. Gallwitz, and C. Riess (2021). Segmentation of photovoltaic module cells in uncalibrated electroluminescence images. *32*(4).

Deitsch, S., V. Christlein, S. Berger, C. Buerhop-Lutz, A. Maier, F. Gallwitz, and C. Riess (2019, June). Automatic classification of defective photovoltaic module cells in electroluminescence images. *Solar Energy 185*, 455–468.

Diakonikolas, I., D. M. Kane, and A. Stewart (2016). The fourier transform of poisson multinomial distributions and its algorithmic applications. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 1060–1073.

Hong, Y. (2013a). On computing the distribution function for the poisson binomial distribution. *Computational Statistics & Data Analysis 59*, 41–51.

Hong, Y. (2013b). On computing the distribution function for the poisson binomial distribution. *Computational Statistics & Data Analysis 59*, 41–51.

Kamath, G. G. C. (2014). *On learning and covering structured distributions*. Ph. D. thesis, Massachusetts Institute of Technology.

Zhang, M., Y. Hong, and N. Balakrishnan (2018). The generalized poisson-binomial distribution and the computation of its distribution function. *Journal of Statistical Computation and Simulation 88*(8), 1515–1527.