Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in
Statistica per l'Economia e l'Impresa

Relazione Finale

Developing Underdispersed Discrete Distributions: A New Approach to Poisson Generalization

Relatore Prof. Alessandra Salvan
Dipartimento di Scienze Statistiche

Laureando Panizzutti Giorgio
Matricola 2033258

Anno Accademico 2023/2024

Contents

Introduction

The Poisson distribution is widely used for modeling count data due to its simplicity and well-understood properties. However, it assumes equidispersion, where the mean and variance are equal, an assumption that often does not hold in real-world data. While overdispersion—variance exceeding the mean—has been extensively studied and addressed through various models, underdispersion—variance less than the mean—has received comparatively less attention. However, underdispersion occurs in numerous practical scenarios, highlighting the need for models that can accurately represent it.

Following McCullagh (2002) principle of model extension, Huang (2023) proposes that a valid count model should be capable of representing any feasible combination of mean and variance, including the minimal variance achievable for a given mean. This minimal variance is found in the shifted Bernoulli distribution, serving as the ideal limiting case for maximum underdispersion in discrete models. Huang (2023) shows that most existing models fail to adhere to this principle when modeling underdispersion, limiting their effectiveness. The Conway-Maxwell-Poisson (CMP) distribution is a notable exception, capable of modeling both overdispersion and underdispersion, and converging to the shifted Bernoulli distribution under extreme underdispersion. However, the CMP distribution lacks explicit formulas for its normalization constant and mean, requiring computational methods, which can complicate its application.

The objective of this thesis is to develop an alternative generalization of the Poisson distribution that follows the ideal underdispersion behavior. Chapter 1 introduces some basic concepts and notation, in Chapter 2 we define a general family of discrete distributions whose limiting behavior matches the ideal case of maximum underdispersion. Building on this foundation, in Chapter 3 we construct a mean-parametrized generalization of the Poisson distribution that accommodates arbitrary underdispersion while retaining desirable statistical properties and computational tractability. We implemented the model in an R library, as described in Chapter 4.

Chapter 1

Underdispersion in discrete models for counts

1.1 Introduction

The Poisson distribution is widely utilized for modeling count data and is considered a fundamental probability model due to its simplicity and properties. However, it assumes equidispersion, meaning that the mean and variance are equal. This assumption often does not hold in practice, leading to interest in more flexible models.

1.2 The Poisson distribution

The Poisson distribution models the probability of observing k events in a fixed interval of time or space, given a constant average rate $\lambda > 0$ and the independence of events. Its probability mass function is defined as

$$P(X = k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \in \mathbb{N}_0. \tag{1.1}$$

Where $\mathbb{N}_0$ is the set of natural numbers including 0.

The cumulative distribution function (CDF) of the Poisson distribution, which gives the probability of observing up to $k \in \mathbb{N}_0$ events, is

$$P(X \leq k; \lambda) = \sum_{i=0}^{k} \frac{\lambda^i e^{-\lambda}}{i!} = \frac{\Gamma(k+1, \lambda)}{\Gamma(k+1)} = Q(k+1, \lambda), \tag{1.2}$$

where $\Gamma(a, x)$ is the upper incomplete gamma function defined as

$$\Gamma(a, x) = \int_x^\infty t^{a-1} e^{-t}\, dt,$$

while $\Gamma(a)$ is the complete gamma function

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t}\, dt.$$

The (upper) regularized gamma function is then defined as $Q(a, x) = \Gamma(a, x)/\Gamma(a)$.

For a Poisson-distributed random variable X, the mean $\mathbb{E}(X)$ and the variance $\mathrm{Var}(X)$ are both equal to the rate parameter λ.

1.3 Underdispersion and overdispersion

It is not trivial to extend the Poisson distribution in a natural way to account for cases of overdispersion, where $\mathrm{Var}(X) > \mathbb{E}(X)$, or underdispersion, where $\mathrm{Var}(X) < \mathbb{E}(X)$. Overdispersion is more commonly encountered and has been extensively studied, but underdispersion is also a frequently observed phenomenon and can be more challenging to model correctly.

1.3.1 Underdispersed count models

Many models have been used to model underdispersed counts, such as the double Poisson, seen in Efron (1986), the exponentially weighted Poisson, described in Ridout & Besbeas (2004), and the Conway-Maxwell-Poisson (CMP) seen in Shmueli et al. (2005). An overview of generalizations of the Poisson distribution admitting underdispersion can be found in Huang (2023) and Sellers & Morris (2017). Huang emphasizes that any future generalization of the Poisson distribution should follow an intuitive but rare principle, which McCullagh (2002) outlines in his discussion of the natural extension of statistical models.

According to McCullagh's principle, a valid statistical model should admit a natural extension to include the domain for which inference is required. For a count distribution, this implies that the model should be capable of representing any desired mean and variance that are possible.

Underdispersion can be less intuitive in discrete variables since, unlike many continuous distributions, not all pairs of mean and variance can be achieved. As argued

in Huang (2023), most existing models degenerate into a unit probability mass at the integer closest to the desired mean when maximum underdispersion occurs.

The minimum possible variance for a discrete model with mean μ on $\mathbb{N}_0$ is achieved by the distribution

$$P(Y = k; \mu) = \begin{cases} 1 - (\mu - \lfloor \mu \rfloor), & \text{if } k = \lfloor \mu \rfloor, \\ \mu - \lfloor \mu \rfloor, & \text{if } k = \lfloor \mu \rfloor + 1, \\ 0, & \text{otherwise,} \end{cases} \tag{1.3}$$

where $\mu \geq 0$ represents the mean of the distribution, and $\lfloor \mu \rfloor$ is the greatest integer less than or equal to μ. This form describes a shifted Bernoulli distribution, where the probability mass is split between the two nearest integers surrounding the mean μ. The parameter $\Delta = \mu - \lfloor \mu \rfloor$ corresponds to the fractional part of μ, which determines the exact probabilities of the two outcomes $\lfloor \mu \rfloor$ and $\lfloor \mu \rfloor + 1$. The minimum possible variance achievable, that of the shifted Bernoulli distribution, is then equal to $\Delta(1-\Delta)$. This quantity is equal to 0 only when μ is an integer and has maximum value $1/4$ when $\Delta = 1/2$. We will refer to this distribution as the shifted Bernoulli distribution, including cases where the distribution collapses to a unit mass when the mean is an integer.

Ideally, we seek a generalized count model having the Poisson distribution as a special case and the shifted Bernoulli as the limiting distribution as the dispersion parameter goes to zero. Huang remarks that currently, among Poisson generalizations, only the CMP distribution possesses the ability to account for arbitrary underdispersion without degenerating into a unit probability mass when the mean μ is not an integer.

1.3.2 The Conway-Maxwell-Poisson distribution

The CMP distribution generalizes the Poisson distribution and is defined by the following probability mass function

$$P(Y = y; \lambda, \beta) = \frac{\lambda^y}{(y!)^\beta} \frac{1}{Z(\lambda, \beta)}, \quad y = 0, 1, 2, \ldots$$

where $\lambda > 0$ is the rate parameter, $\beta \geq 0$ is the dispersion parameter, and $Z(\lambda, \beta)$ is the normalizing constant that makes the sum of probabilities equal to one. The parameter β controls the dispersion, with $\beta = 1$ corresponding to the standard Poisson distribution, $\beta > 1$ leading to underdispersion, and $\beta < 1$ to overdispersion.

Huang (2017) proposed a reparametrization of the CMP distribution, replacing the rate parameter λ with the mean μ using an implicit formula. Huang shows that as $\beta \to \infty$, this mean-parametrized CMP has the limiting distribution that is the shifted Bernoulli (1.3).

A downside of this distribution is that the mean parametrization and the normalizing constant $Z(\lambda, \beta)$ in general do not have a closed form and need to be computed numerically.

Chapter 2

A family of count distributions with arbitrary mean and ideal underdispersion behavior

In this chapter, we analyze the shifted Bernoulli distribution to construct a continuous transition to other discrete distributions controlled by a dispersion parameter ν. This approach allows us to define a family of discrete models that exhibit ideal limit behavior under maximum underdispersion and also have the property of mean-parametrization.

2.1 The likelihood of the shifted Bernoulli

The likelihood function $L(\mu; k)$ for a one-parameter statistical model is the probability mass (or probability density) of the observed data k, viewed as a function of the parameter μ. For discrete distributions, it is defined as

$$L(\mu; k) = P(X = k; \mu)$$

where $P(X = k; \mu)$ represents the probability mass function of the discrete variable X.

Here, we refer to the likelihood as the PMF expressed as a function of the parameters, not considering equivalent likelihoods that are all the scalar multiples of this function. Chapters 2 and 3 focus on analyzing the shape of the PMF as a function of the mean, without using it for the typical inferential applications.

For a given k, the likelihood of the shifted Bernoulli defined in (1.3) has the shape of a triangular distribution centered at k with 45-degree angles (as illustrated in Figure 2.1 for $k = 2$).

We can write this function of μ without defining it piecewise, breaking it down as the following sum of 3 maximum functions in the form

$$\max(0, \mu - k + 1) - 2\max(0, \mu - k) + \max(0, \mu - k - 1) \tag{2.1}$$

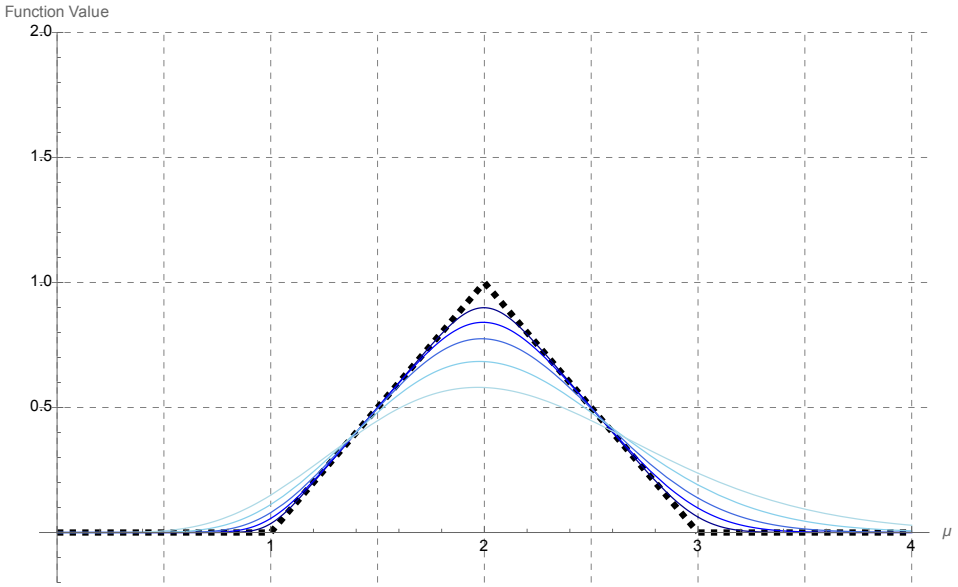


FIGURE 2.1: Likelihood function for a shifted Bernoulli distribution in $k = 2$ (black) and an example of a smooth function converging to this shape (blue)

2.1.1 Expanding the shifted Bernoulli

Since we want the shifted Bernoulli (1.3) to be the limit distribution of a generalized count model, the function (2.1) must be the limit likelihood of this model. To achieve this, we introduce a dispersion parameter ν.

Since we are constructing a likelihood function, it is ideal for it to be smooth in its domain and, in general, at least two times differentiable in the parameters. We want to find such a function that converges to the (2.1) as $\nu \to 0$ and becomes non-differentiable only in this limit case.

To do so, we begin by observing a single term of the sum (2.1)

$$\max(0, \mu - k). \tag{2.2}$$

This function of μ is 0 until $\mu = k$ and then it becomes a line with slope 1 and y-intercept $-k$ when $\mu > k$.

2.1.2 A connection to continuous random variables

It is useful to find a way to construct a smooth function $\mathcal{F}(\mu, k, \nu)$ that converges to (2.2) as $\nu \to 0$. We aim to maintain the behavior of the function as $\mu \to 0$ and as $\mu \to \infty$; specifically, we want the function to be zero with zero first derivative at $\mu = 0$, and to have a 45-degree oblique asymptote as $\mu \to \infty$. An interesting connection to continuous probability distributions emerges if we consider $\mathcal{F}(\mu, k, \nu)$ to be increasing and convex in μ. In fact, the derivative with respect to μ of this function is zero at $\mu = 0$, increases, and approaches 1 as $\mu \to \infty$. This implies that

$$\frac{\partial \mathcal{F}(\mu, k, \nu)}{\partial \mu}$$

is the cumulative distribution function (CDF) of a continuous random variable with support $\mathbb{R}_+$. The derivative of the limit case (2.2) corresponds to the CDF of a degenerate random variable with unit mass at k. It is zero when $\mu < k$ and one when $\mu \geq k$. This means that it is possible to create a smooth function converging to the function (2.1) using some specific continuous distributions.

The constructed function is then of the form

$$\mathcal{F}(\mu, k, \nu) = \int_0^\mu F(t; k, \nu)\, dt, \tag{2.3}$$

where $F(t; k, \nu)$ is the cumulative distribution function (CDF) of a non-negative continuous random variable. As we will see, this random variable must have mean $k \in \mathbb{N}_0$ and a dispersion parameter $\nu > 0$ such that the distribution converges in distribution to a unit point mass in k as $\nu \to 0$. Then, a function converging to (2.1) can be obtained by summing 3 functions of the form in (2.3)

$$\mathcal{F}_A(\mu, k+1, \nu) - 2\mathcal{F}_B(\mu, k, \nu) + \mathcal{F}_C(\mu, k-1, \nu). \tag{2.4}$$

An example is shown in Figure 2.1 for $k = 2$ and varying dispersion parameters. This function can be rewritten using the identity

$$\mathcal{S}(\mu, k, \nu) = \int_0^\mu 1 - F(t; k, \nu)\, dt = \int_0^\mu S(t; k, \nu)\, dt = \mu - \mathcal{F}(\mu, k, \nu) \tag{2.5}$$

That is the integral of the survival function $S(t; k, \nu) = 1 - F(t; k, \nu)$ of the same continuous random variable.

Then

$$-\mathcal{S}_A(\mu, k+1, \nu) + 2\mathcal{S}_B(\mu, k, \nu) - \mathcal{S}_C(\mu, k-1, \nu) \tag{2.6}$$

is equivalent to the function (2.4).

This is useful because of a known property of the survival function, as seen, for example, in Subhabrata Chakraborti & Epprecht (2019), which states that the area below the survival function of a non-negative random variable is equal to the expected value of the random variable (when it exists):

$$\int_0^\infty 1 - F(t)\,dt = \int_0^\infty S(t)\,dt = \mathbb{E}[X] \tag{2.7}$$

where $S(t)$ is the survival function of the random variable X.

This property demonstrates that, as desired, the function in equation (2.3) has an oblique asymptote as $\mu \to \infty$, specifically at $y = \mu - k$, where k is the mean of the distribution whose CDF is $F(t; k, \nu)$. The value of the intercept $-k$ can be obtained by considering the limit

$$\begin{aligned}
\lim_{\mu\to\infty}[\mathcal{F}(\mu, k, \nu) - \mu] &= \lim_{\mu\to\infty}\left[\int_0^\mu F(t; k, \nu) - 1\right] dt \\
&= -\lim_{\mu\to\infty}\int_0^\mu [1 - F(t; k, \nu)]\,dt \\
&= -\int_0^\infty S(t; k, \nu)\,dt \\
&= -\mathbb{E}[X] \\
&= -k.
\end{aligned}$$

Therefore, the asymptote is at $y = \mu - k$.

A visualization of an example of a sequence of functions (2.3) and (2.5) as functions of μ for values of k from 0 to 10 is given in Figure 2.2.

Using the above argument, we can construct count distributions that have some ideal properties.

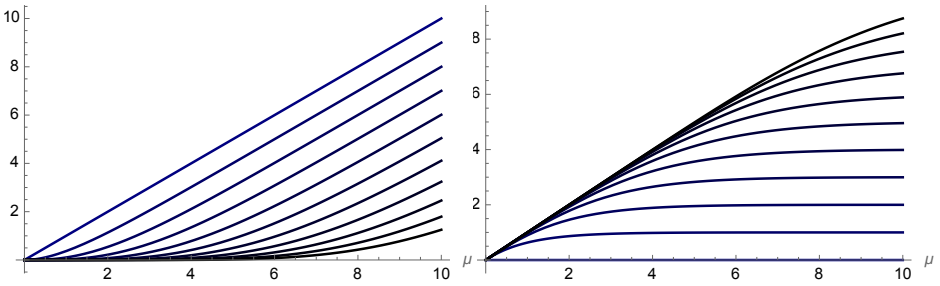


FIGURE 2.2: Examples of a sequence of functions of the form (2.3) (left) and (2.5) (right) for $k = 0, \ldots, 10$ for a fixed dispersion parameter

2.2 Defining a family of arbitrarily underdispersed count distributions

We begin by defining a sequence of functions

$$\{\mathcal{S}_k(\mu,\nu)\}_{k\in\mathbb{Z}} \tag{2.8}$$

with $\mu,\nu \in \mathbb{R}_+$. These functions are constructed as defined in (2.5) using the integral of the survival function of a non-negative continuous random variable with mean k and dispersion parameter ν

$$\mathcal{S}_k(\mu,\nu) = \mathcal{S}(\mu,k,\nu) = \int_0^\mu [1 - F(t;k,\nu)]\,dt = \int_0^\mu S(t;k,\nu)\,dt. \tag{2.9}$$

making sure that as $\nu \to 0$ the random variable converges in distribution to a unit point mass in k.

The survival function of a random variable X with support $\mathbb{N}_0$ belonging to this family is defined as the first difference of this sequence:

$$P(X > k;\mu,\nu) = \mathcal{S}_{k+1}(\mu,\nu) - \mathcal{S}_k(\mu,\nu). \tag{2.10}$$

For values of $k < 0$, the survival function of X has value one as expected, since applying formula (2.9), we get $\mathcal{S}_k(\mu,\nu) = k$ for all k negative integers.

The PMF is

$$\begin{aligned} P(X = k;\mu,\nu) &= P(X > k-1;\mu,\nu) - P(X > k;\mu,\nu) \\ &= \mathcal{S}_k(\mu,\nu) - \mathcal{S}_{k-1}(\mu,\nu) - [\mathcal{S}_{k+1}(\mu,\nu) - \mathcal{S}_k(\mu,\nu)] \\ &= -\mathcal{S}_{k+1}(\mu,\nu) + 2\mathcal{S}_k(\mu,\nu) - \mathcal{S}_{k-1}(\mu,\nu). \end{aligned} \tag{2.11}$$

The likelihood then has the form seen in (2.6).

To make sure that the function defined in equation (2.10) is a valid survival function for a discrete random variable X with support $\mathbb{N}_0$ and mean μ, specific additional requirements must be met.

2.2.1 Additional requirements

The following conditions are necessary to construct a valid model for all $\mu,\nu > 0$:

$$\mathcal{S}_k(\mu,\nu) < \mathcal{S}_{k+1}(\mu,\nu), \quad \forall k \in \mathbb{Z}. \tag{2.12}$$

$$\mathcal{S}_{k+1}(\mu,\nu) - \mathcal{S}_k(\mu,\nu) \text{ is strictly decreasing in } k \geq 0. \tag{2.13}$$

$$\lim_{k\to\infty} \mathcal{S}_k(\mu,\nu) = \mu. \tag{2.14}$$

Requirement (2.12) is necessary; otherwise, we could obtain negative values from equation (2.10). The strict inequality ensures that the survival function remains positive. This property is very close to second-order stochastic dominance between continuous random variables that is implied by the commonly known stronger property of first-order stochastic dominance.

A random variable W_1 first-order stochastically dominates another random variable W_2 if their cumulative distribution functions satisfy:

$$F_{W_1}(x) \leq F_{W_2}(x) \quad \text{for all } x \in \mathbb{R},$$

with strict inequality for some x, where $F_{W_1}(x)$ and $F_{W_2}(x)$ are the cumulative distribution functions of W_1 and W_2, respectively.

A random variable W_1 second-order stochastically dominates another random variable W_2 if:

$$\int_{-\infty}^{x} [F_{W_2}(t) - F_{W_1}(t)]\, dt \geq 0 \quad \text{for all } x \in \mathbb{R},$$

with strict inequality for some x. In our case the inequality (2.12) must be strict for every $\mu, \nu > 0$ when applied to the quantity

$$\begin{aligned}\mathcal{S}_{k+1}(\mu,\nu) - \mathcal{S}_k(\mu,\nu) &= \int_0^{\mu} [1 - F(t;k+1,\nu) - (1 - F(t;k,\nu))]\, dt \\ &= \int_{-\infty}^{\mu} [F(t;k,\nu) - F(t;k+1,\nu)]\, dt > 0.\end{aligned}$$

This means that (2.12) requires the distribution linked to $\mathcal{S}_{k+1}(\mu,\nu)$ to second-order stochastically dominate on the one linked to $\mathcal{S}_k(\mu,\nu)$.

Requirement (2.13) ensures that the differences $\mathcal{S}_{k+1}(\mu,\nu) - \mathcal{S}_k(\mu,\nu)$ are strictly decreasing as k increases. If this condition is not met, the survival function of the discrete model would not become strictly decreasing in $k \in \mathbb{N}_0$ and thus would not define a probability distribution with support in $\mathbb{N}_0$. This property does not appear to have a simple direct connection to the properties of the related continuous distributions.

The survival function in zero

$$P(X > 0; \mu, \nu) = 1 - P(X = 0; \mu, \nu) = \mathcal{S}_1(\mu, \nu) \tag{2.15}$$

is positive and strictly less than 1 for all $\mu, \nu > 0$ since $\mathcal{S}_1(\mu, \nu)$ is the integral from 0 to $\mu < \infty$ of the survival function of a random variable with support in $\mathbb{R}_+$ and mean 1. If these conditions apply, (2.10) is a valid survival function for a discrete random variable because it has value 1 in negative integers k, and decreases approaching 0 as $k \to \infty$ since for (2.14)

$$\lim_{k\to\infty} \mathcal{S}_{k+1}(\mu, \nu) - \mathcal{S}_k(\mu, \nu) = \mu - \mu = 0 \tag{2.16}$$

2.2.2 Mean

Requirement (2.14) implies that the mean of the discrete distribution is exactly μ. This holds because $\mathcal{S}_k(\mu, \nu)$ is effectively the sum from zero up to $k-1$ of the survival function of the random variable X:

$$\begin{aligned}\sum_{i=0}^{k-1} P(X > i; \mu, \nu) &= \sum_{i=0}^{k-1} (\mathcal{S}_{i+1}(\mu, \nu) - \mathcal{S}_i(\mu, \nu)) \\ &= (\mathcal{S}_1(\mu, \nu) - \mathcal{S}_0(\mu, \nu)) + (\mathcal{S}_2(\mu, \nu) - \mathcal{S}_1(\mu, \nu)) + \cdots + (\mathcal{S}_k(\mu, \nu) - \mathcal{S}_{k-1}(\mu, \nu)) \\ &= \mathcal{S}_k(\mu, \nu) - \mathcal{S}_0(\mu, \nu).\end{aligned}$$

Since $\mathcal{S}_0(\mu, \nu) = 0$, for $k > 0$

$$\mathcal{S}_k(\mu, \nu) = \sum_{i=0}^{k-1} P(X > i; \mu, \nu) \tag{2.17}$$

This works because the formula (2.7) has an analog for non-negative discrete random variables. As seen in Subhabrata Chakraborti & Epprecht (2019), the expected value of a non-negative discrete distribution, if it exists, can be calculated using the formula

$$\mathbb{E}[X] = \sum_{i=0}^{\infty} P(X > i) \tag{2.18}$$

For a distribution in the family defined earlier, using the survival function defined in equation (2.10), this simplifies to

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i=0}^{\infty} P(X > i; \mu, \nu) = \sum_{i=0}^{\infty} \left[\mathcal{S}_{i+1}(\mu, \nu) - \mathcal{S}_i(\mu, \nu)\right] \\ &= \lim_{k\to\infty} \sum_{i=0}^{k-1} \left[\mathcal{S}_{i+1}(\mu, \nu) - \mathcal{S}_i(\mu, \nu)\right] \\ &= \lim_{k\to\infty} \mathcal{S}_k(\mu, \nu) \\ &= \mu \end{aligned} \tag{2.19}$$

using (2.14) for the last equality.

2.2.3 Variance

The variance can be computed calculating the second moment with the following formula that generalizes (2.18) for the r-th moment of a non-negative discrete random variable seen in Subhabrata Chakraborti & Epprecht (2019):

$$\mathbb{E}[X^r] = \sum_{i=0}^{\infty} \left[((i+1)^r - i^r)P(X > i)\right], \quad r \geq 1.$$

When $r = 2$ is

$$\begin{aligned}\mathbb{E}[X^2] &= \sum_{i=0}^{\infty} (2i+1)(P(X > i)) \\ &= \mathbb{E}[X] + 2\sum_{i=0}^{\infty} i(P(X > i)) \end{aligned} \tag{2.20}$$

With the survival function (2.10), the formula for the second moment becomes:

$$\begin{aligned}\mathbb{E}[X^2] &= \mathbb{E}[X] + 2\sum_{i=0}^{\infty} i(\mathcal{S}_{i+1}(\mu, \nu) - \mathcal{S}_i(\mu, \nu)) \\ &= \mu + 0 + 2\sum_{i=1}^{\infty} i(\mathcal{S}_{i+1}(\mu, \nu) - \mathcal{S}_i(\mu, \nu)) \\ &= \mu + 2\sum_{i=1}^{\infty} i\mathcal{S}_{i+1}(\mu, \nu) - i\mathcal{S}_i(\mu, \nu)\end{aligned}$$

Replacing the infinite series with the limit of the partial sum and using the limit (2.14), we get

$$\begin{aligned}\mathbb{E}[X^2] &= \mu + 2\lim_{n\to\infty}\left[n\mathcal{S}_{n+1}(\mu,\nu) - \sum_{i=1}^{n}\mathcal{S}_i(\mu,\nu)\right] \\ &= \mu + 2\lim_{n\to\infty}\left[n\mu - \sum_{i=1}^{n}\mathcal{S}_i(\mu,\nu)\right] \\ &= \mu + 2\lim_{n\to\infty}\sum_{i=1}^{n}\mu - \mathcal{S}_i(\mu,\nu) \\ &= \mu + 2\sum_{i=1}^{\infty}\left[\mu - \mathcal{S}_i(\mu,\nu)\right]. \end{aligned} \tag{2.21}$$

Then, the variance is obtained as

$$\begin{aligned}\mathrm{Var}(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= \mu + 2\sum_{i=1}^{\infty}\left[\mu - \mathcal{S}_i(\mu,\nu)\right] - \mu^2. \end{aligned} \tag{2.22}$$

2.2.4 Support independent of the parameters

An additional desirable property is that the support of the distribution remains fixed, equal to $\mathbb{N}_0$, regardless of the values of μ and ν when they are greater than 0. Similarly, the parameter space is $\mu > 0$ and $\nu > 0$, independently of the specific values of each parameter. This property clearly does not apply in the limit case when $\nu = 0$, that is the shifted Bernoulli case. In fact the shifted Bernoulli has support equal to only 1 or 2 integers depending on the μ value.

2.3 Applications

Starting with a mean-parametrized one-parameter count distribution, we can obtain a function of the form (2.9) taking the sum from 0 to k of the survival function of this distribution as shown in formula (2.17)

$$\mathcal{S}_{k+1}(\mu) = \sum_{i=0}^{k} P(X > i;\mu). \tag{2.23}$$

This expression may not always have a closed form, but it does in at least two major cases: Poisson and geometric distribution. Now, we can get the survival function of the

associated continuous distribution with mean $k+1$, by taking the derivative in μ of this function

$$S(\mu; k+1) = \frac{\partial}{\partial \mu}\mathcal{S}_{k+1}(\mu) = \frac{\partial}{\partial \mu}\sum_{i=0}^{k} P(X > i; \mu) = \sum_{i=0}^{k}\frac{\partial}{\partial \mu}P(X > i; \mu). \tag{2.24}$$

Once we have this associated continuous distribution, we can try to generalize it by adding a dispersion parameter that introduces arbitrary dispersion, making sure to retain mean parametrization and making sure all other conditions (2.12),(2.13),(2.14) are satisfied.

This associated continuous distribution should exist in most regular cases, but it is simple to think of cases where it does not. An example would be if a count model has the probability of 0, that is equal to $1 - \mathcal{S}_1(\mu, \nu)$, that is not monotonic increasing in the mean, having its derivative, the survival function of the continuous distribution, not strictly positive.

In general, there seem to be some reasonable assumptions that are sufficient to guarantee that a mean-parametrized count model has an associated mean-parametrized non-negative continuous distribution found using (2.24):

1. stochastic ordering by the mean μ of the discrete distributions, using second-order stochastic dominance as the ordering criterion

2. $P(k; \mu, \nu) \to 0$ for all k as $\mu \to \infty$.

As stated in Section 2.2.1, the first property is implied by the usual stochastic ordering by the mean using first-order stochastic dominance (the CDF is decreasing in the mean μ). This is also called stochastic monotonicity and is, for example, a commonly known property of the one-parameter exponential family (as seen in Appendix A of Davis & Liu (2016)). Here, we consider these assumptions reasonable for a generalized count model with a fixed dispersion parameter, but it should be possible to justify them more rigorously. Count distributions like the Poisson and negative binomial follow these requirements. Intuitively, it makes sense that by increasing the mean, the probability mass shifts to greater values. The first assumption seems also to help make sure that the dispersion parameter behaves as intended. More research is needed to verify if these assumptions are sufficient and if these implications are valid. The general idea is that these two properties should make all functions $\mathcal{S}_k(\mu)$ increasing with limit k when $\mu \to \infty$. The mean parametrization of the discrete model should bind the shape of these $\mathcal{S}_k(\mu)$ into a sequence of the form seen in Figure 2.2 (right plot), leading to a continuous distribution. If these conjectures are correct, then we can say that every reasonable

generalization of a count model and more specifically of the Poisson distribution should have an associated arbitrarily underdispersed continuous model found using the formula (2.24). To be more specific, now we are using reasonable to refer to a count model that behaves intuitively, meaning that it has limit underdispersion behavior of a shifted Bernoulli as described in Chapter 1, but also that has stochastic monotonicity in the mean and probabilities going to 0 as the mean parameter goes to infinity.

2.3.1 Examples

A Poisson generalization found with this method will be described in Chapter 3. The Poisson case is also unique since the continuous distribution linked by the sum of the survival function of the Poisson is the same as its likelihood function, which has the shape of a gamma density with mean $k+1$ and scale parameter equal to 1.

The mean-parametrized CMP seen in Chapter 1 likely belongs to this family (if we reparametrize it using the reciprocal of the dispersion parameter $\nu = 1/\beta$). If it does, then it should be linked to a generalization of the gamma distribution with scale $\theta = 1$, but that in general is different from the standard gamma distribution.

Another example is the mean-parametrized geometric distribution with support $\mathbb{N}_0$ that has PMF:

$$P(x = k, \mu) = \frac{\left(1 - \frac{1}{\mu+1}\right)^k}{\mu + 1} \tag{2.25}$$

Calculations have been performed using Wolfram Mathematica and are available in Appendix A. It is possible to verify that the partial sum of its survival function has a simple closed form

$$\mathcal{S}_{k+1}(\mu) = -\frac{\mu\left(\mu\left(1 - \frac{1}{\mu+1}\right)^k - \mu - 1\right)}{\mu + 1} \tag{2.26}$$

Then, if we differentiate twice with respect to μ (and change the sign), we find a probability density function (PDF)

$$\frac{(k^2 + 3k + 2)\left(\frac{\mu}{\mu+1}\right)^k}{(\mu+1)^3} \tag{2.27}$$

that has mean $k+1$ as expected. If we find a way to generalize this continuous distribution, introducing a dispersion parameter while keeping the mean and support constant, we could reach an arbitrarily underdispersed generalization of the geometric distribution. (this may be harder in this case since the (2.27) has infinite variance).

2.4 Conclusions

The method proposed enables the construction of flexible generalized count distributions by introducing a dispersion parameter and ensuring an appropriate underdispersion behavior. It is important to note that this method is not inherently linked to any specific mechanism driving underdispersion. Mechanisms for underdispersion vary significantly, as described in Puig et al. (2024). Identifying them precisely is often challenging, and representing them can be computationally complex. However, having an exact theoretical basis for every model may not be necessary as long as the models we use behave as intended and help to reach better conclusions.

Chapter 3

Generalization of the Poisson distribution

3.1 Introduction

In this chapter, we use the connection between the Poisson and gamma distributions to generalize the Poisson distribution within the family of discrete distributions defined in Chapter 2. Using the integral of the survival function of a mean-parametrized gamma distribution, we construct the sequence of functions $\{\mathcal{S}_k(\lambda, \theta)\}$ that satisfies the properties outlined in Chapter 2. This approach allows us to define a new discrete distribution that retains the mean parametrization of the Poisson distribution while providing flexibility in modeling dispersion.

3.2 The gamma distribution

The gamma distribution is a non-negative continuous probability distribution characterized by a shape parameter $\alpha > 0$ and a scale parameter $\theta > 0$. Its PDF is

$$f(t; \alpha, \theta) = \frac{t^{\alpha-1} e^{-t/\theta}}{\Gamma(\alpha)\theta^{\alpha}}, \quad t \geq 0, \tag{3.1}$$

where $\Gamma(\alpha)$ is the gamma function:

$$\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t}\, dt. \tag{3.2}$$

The CDF is

$$F(t; \alpha, \theta) = \frac{1}{\Gamma(\alpha)} \gamma\left(\alpha, \frac{t}{\theta}\right), \tag{3.3}$$

where $\gamma(\alpha, x)$ is the lower incomplete gamma function:

$$\gamma(\alpha, x) = \int_0^x t^{\alpha-1} e^{-t}\, dt. \tag{3.4}$$

3.3 Finding $\mathcal{S}_k$ for the Poisson distribution

3.3.1 Connection between gamma and Poisson

There is a noteworthy connection between the density of the gamma distribution and the Poisson distribution. Let X be a random variable following a Poisson distribution with mean parameter λ. The likelihood function for an observed value $X = k$ is given by:

$$L(\lambda; k) = P(X = k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}.$$

Now, consider the Gamma distribution with shape parameter α and scale parameter $\theta = 1$, which has PDF

$$f(\lambda; \alpha, 1) = \frac{\lambda^{\alpha-1} e^{-\lambda}}{\Gamma(\alpha)}. \tag{3.5}$$

We can now observe that if we set the shape parameter of the gamma distribution to $\alpha = k + 1$ we obtain

$$L(\lambda; k) = \frac{\lambda^k e^{-\lambda}}{k!} = f(\lambda; k+1, 1) = \frac{\lambda^{(k+1)-1} e^{-\lambda}}{\Gamma(k+1)}, \tag{3.6}$$

where we have used the identity $k! = \Gamma(k + 1)$.

The Poisson likelihood function (without ignoring the constant $1/k!$) for an observation $X = k$ has the same functional form as a gamma PDF for λ with shape $\alpha = k + 1$ and scale $\theta = 1$. This is a known fact and is strictly related to the conjugacy property in Bayesian statistics, where the gamma is the conjugate prior for the Poisson likelihood.

3.3.2 Derivative of the gamma density with $\theta = 1$

An interesting recursive property of the gamma density function arises when $\theta = 1$. The first partial derivative of the gamma density $f(x; \alpha) = f(x; \alpha, 1)$ with respect to x is

$$\begin{aligned}\frac{\partial}{\partial x} f(x; \alpha) &= \frac{\partial}{\partial x}\left(\frac{e^{-x} x^{\alpha-1}}{\Gamma(\alpha)}\right) \\ &= \frac{1}{\Gamma(\alpha)}\left(-e^{-x} x^{\alpha-1} + e^{-x}(\alpha-1)x^{\alpha-2}\right)\end{aligned}$$

If $\alpha > 1$ we can use the property $\Gamma(\alpha) = (\alpha - 1)\Gamma(\alpha - 1)$. Thus, the first derivative simplifies to:

$$\begin{aligned} \frac{\partial}{\partial x} f(x;\alpha) &= \frac{e^{-x}x^{\alpha-2}}{\Gamma(\alpha-1)} - \frac{e^{-x}x^{\alpha-1}}{\Gamma(\alpha)} \\ &= f(x;\alpha-1) - f(x;\alpha). \end{aligned} \tag{3.7}$$

This shows that the partial derivative of the gamma density with parameter α is the difference between the gamma densities with shape $\alpha - 1$ and α. When $\alpha = 1$, the gamma density simplifies to the exponential distribution:

$$f(x;1) = e^{-x}.$$

Its derivative is:

$$\frac{\partial}{\partial x} f(x;1) = -e^{-x} = -f(x;1),$$

This means that formula (3.7) can still be used when $\alpha \leq 1$ using $f(x;\alpha) = 0$ whenever α is a non-positive integer. This is also the limit behavior of the function (3.1) since the reciprocal gamma function $\Gamma(z)^{-1}$ can be continuously extended to have a value of zero at all non-positive integers. By doing this, we ensure that equation (3.7) remains valid for $x > 0$ for all integer values of α.

The second and further partial derivatives can be obtained recursively. The second derivative for $\alpha > 2$ is:

$$\begin{aligned} \frac{\partial^2}{\partial x^2} f(x;\alpha) &= \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x} f(x;\alpha)\right) \\ &= \frac{\partial}{\partial x}\left(f(x;\alpha-1) - f(x;\alpha)\right) \\ &= \left(\frac{\partial}{\partial x} f(x;\alpha-1)\right) - \left(\frac{\partial}{\partial x} f(x;\alpha)\right) \\ &= \left(f(x;\alpha-2) - f(x;\alpha-1)\right) - \left(f(x;\alpha-1) - f(x;\alpha)\right) \\ &= f(x;\alpha-2) - 2f(x;\alpha-1) + f(x;\alpha). \end{aligned} \tag{3.8}$$

This result for the second derivative reveals a surprising connection to the family described in Chapter 2. Integrating both sides of (3.8) twice with respect to x and using the linearity of the integral, we have:

$$\iint \frac{\partial^2}{\partial x^2} f(x;\alpha)\, dx\, dx = \iint \left[f(x;\alpha-2) - 2f(x;\alpha-1) + f(x;\alpha)\right] dx\, dx.$$

The left-hand side simplifies to the original function plus linear terms due to the integration constants:

$$f(x;\alpha)+C_1x+C_2=\iint f(x;\alpha-2)\,dx\,dx-2\iint f(x;\alpha-1)\,dx\,dx+\iint f(x;\alpha)\,dx\,dx,$$

where C_1 and C_2 are constants of integration. The individual functions are integrable in $(0,\infty)$ since they are PDFs of continuous random variables or 0 when $\alpha\leq 0$.

$$f(x;\alpha)=\iint f(x;\alpha-2)\,dx\,dx-2\iint f(x;\alpha-1)\,dx\,dx+\iint f(x;\alpha)\,dx\,dx+C_1x+C_2.$$

This equivalence demonstrates that the gamma PDF with $\theta=1$ can be written as second difference with respect to α of the second antiderivative with respect to x of gamma PDFs. This also means that the Poisson PMF can be written using the same formula since they have the same functional form as explained in (3.6).

As described in Section 2.3, we can then introduce a dispersion parameter allowing for arbitrary dispersion in the gamma, while keeping mean parametrization. Here, we will reintroduce the standard gamma distribution scale parameter θ.

3.4 Mean-parametrization

In order to find a generalization of the Poisson distribution belonging to the family described in Chapter 2, it is necessary to have a sequence of mean-parametrized non-negative probability density functions. The case where $\theta=1$ already satisfies this property since the shape parameter α is also the mean of the distribution. Changing the value of θ modifies the mean. For this reason, we will change the parametrization for the gamma density, changing the shape parameter into the mean.

The mean and variance of the gamma distribution are

$$k=\alpha\theta,\quad \sigma^2=\alpha\theta^2. \tag{3.9}$$

From this point on, we will use the mean-parametrized gamma distribution by expressing the shape parameter α in terms of the mean k and scale parameter θ

$$\alpha=\frac{k}{\theta}. \tag{3.10}$$

Consequently, the PDF of the mean-parametrized gamma distribution can be defined as follows

$$f^M(x;k,\theta) = f(x;k/\theta,\theta) = \frac{x^{\frac{k}{\theta}-1}e^{-x/\theta}}{\Gamma\left(\frac{k}{\theta}\right)\theta^{\frac{k}{\theta}}}, \quad x \geq 0. \tag{3.11}$$

Here, $f^M(x;k,\theta)$ represents the gamma density function parameterized by the mean $k > 0$ and scale parameter $\theta > 0$.

The CDF of the mean-parametrized gamma is

$$F^M(x;k,\theta) = 1 - \frac{\Gamma\left(\frac{k}{\theta},\frac{x}{\theta}\right)}{\Gamma\left(\frac{k}{\theta}\right)} = 1 - Q\left(\frac{k}{\theta},\frac{x}{\theta}\right).$$

The survival function is:

$$S^M(x;k,\theta) = 1 - F^M(x;k,\theta) = Q\left(\frac{k}{\theta},\frac{x}{\theta}\right). \tag{3.12}$$

3.4.1 Integral of the survival function

With the mean-parametrized gamma density, we calculate the integral of the survival function. The antiderivative of the upper regularized gamma function, as seen in Wolfram Research (2024), is

$$\int Q(a,z)\,dz = z\,Q(a,z) - a\,Q(a+1,z) + C,$$

that for (3.12), it becomes

$$\begin{aligned}\int Q\left(\frac{k}{\theta},\frac{x}{\theta}\right)\,dx &= \theta\int Q\left(\frac{k}{\theta},\frac{x}{\theta}\right)\cdot\frac{1}{\theta}\,dx \\ &= x\,Q\left(\frac{k}{\theta},\frac{x}{\theta}\right) - k\,Q\left(\frac{k}{\theta}+1,\frac{x}{\theta}\right) + C\end{aligned}$$

Then, the integral of the survival function for $\lambda,\theta,k > 0$, is given by:

$$\begin{aligned}\mathcal{G}(\lambda,k,\theta) &= \int_0^\lambda \left[1 - F^M(x;k,\theta)\right]dx \\ &= \int_0^\lambda Q\left(\frac{k}{\theta},\frac{x}{\theta}\right)dx \\ &= \lambda\,Q\left(\frac{k}{\theta},\frac{\lambda}{\theta}\right) - k\,Q\left(\frac{k}{\theta}+1,\frac{\lambda}{\theta}\right) - \left(0\,Q\left(\frac{k}{\theta},\frac{0}{\theta}\right) - k\,Q\left(\frac{k}{\theta}+1,\frac{0}{\theta}\right)\right) \\ &= \lambda\,Q\left(\frac{k}{\theta},\frac{\lambda}{\theta}\right) - k\,Q\left(\frac{k}{\theta}+1,\frac{\lambda}{\theta}\right) + k,\end{aligned} \tag{3.13}$$

using the fact that the regularized gamma function (3.12) is equal to 1 in $x = 0$, since it is the survival function of a non-negative continuous random variable.

3.5 Generalizing the Poisson distribution

We can use the formula (3.13) to find a generalization of the Poisson distribution. We can define a sequence of functions as defined in (2.8), given by

$$\mathcal{S}_k(\lambda, \theta) = \mathcal{G}(\lambda, k, \theta) = \lambda\, Q\left(\frac{k}{\theta}, \frac{\lambda}{\theta}\right) - k\, Q\left(\frac{k}{\theta} + 1, \frac{\lambda}{\theta}\right) + k \tag{3.14}$$

Consistently with the integral of a survival function of a non-negative random variable, this function is k in when k is a non-positive integer. This is because the survival function is 1 below zero, and we are integrating from 0 to a lower value.

Calling X a random variable following the new-found distribution, its survival function as in (2.10), calculated in integer values of k is

$$P(X > k; \lambda, \theta) = \mathcal{S}_{k+1}(\lambda, \theta) - \mathcal{S}_k(\lambda, \theta).$$

The CDF of X, with mean λ and dispersion θ has the following formula

$$\begin{aligned} P(X \leq k; \lambda, \theta) = &\left[(k+1)\, Q\left(\frac{k+1}{\theta} + 1, \frac{\lambda}{\theta}\right) - k\, Q\left(\frac{k}{\theta} + 1, \frac{\lambda}{\theta}\right)\right] \\ &+ \left[\lambda\, Q\left(\frac{k}{\theta}, \frac{\lambda}{\theta}\right) - \lambda\, Q\left(\frac{k+1}{\theta}, \frac{\lambda}{\theta}\right)\right], \end{aligned}$$

Or equivalently:

$$\begin{aligned} P(X \leq k; \lambda, \nu) = &\left[(k+1)\, \frac{\Gamma\left(\frac{k+1}{\theta} + 1, \frac{\lambda}{\theta}\right)}{\Gamma\left(\frac{k+1}{\theta} + 1\right)} - k\, \frac{\Gamma\left(\frac{k}{\theta} + 1, \frac{\lambda}{\theta}\right)}{\Gamma\left(\frac{k}{\theta} + 1\right)}\right] \\ &+ \left[\lambda\, \frac{\Gamma\left(\frac{k}{\theta}, \frac{\lambda}{\theta}\right)}{\Gamma\left(\frac{k}{\theta}\right)} - \lambda\, \frac{\Gamma\left(\frac{k+1}{\theta}, \frac{\lambda}{\theta}\right)}{\Gamma\left(\frac{k+1}{\theta}\right)}\right], \end{aligned} \tag{3.15}$$

with $k \geq 0$ integer. As expected from the support $\mathbb{N}_0$, the CDF is zero if $k < 0$ and $F_X(\lfloor k \rfloor\,; \lambda, \theta)$ if $k > 0$ is not an integer.

3.5.1 Previous work on this distribution

We have to acknowledge that this same model was found at an advanced stage of this work to have already been proposed by Hagmark (2012). In his paper, Hagmark introduces the model through alternative arguments; however, the resulting CDF coincides with the one obtained here. The paper was published in the *Open Journal of Statistics*, which is not indexed in major academic databases such as the Journal Citation Reports (JCR) or Scopus. Consequently, it has received very limited attention in the academic community, with only a few citations in less widely recognized publications and none in prominent journals. It is unclear why this model has not gained much interest and has not been used or mentioned in more recent scientific reviews on underdispersed count models.

In his paper, Hagmark uses a mean-preserving discretization technique to obtain this model. The random variable that Hagmark has discretized is not the gamma distribution but a mixed random variable having probability mass in zero and a probability density in all positive values. For this reason, there seems to be a major difference in the connection to continuous distributions proposed here. The connection to a continuous non-negative random variable with the formula (2.24) should simplify the process of construction, other than also connecting this generalized Poisson distribution directly to to the well-known gamma distribution (3.11) with the standard scale parameter θ. The approach presented in Chapter 2 and the application in Chapter 3 should provide a simpler foundation for constructing this Poisson generalization.

3.5.2 Recovering the Poisson distribution

It can be verified that the CDF (3.15) is equivalent to the Poisson CDF when $\theta = 1$. To do that, we use the recurrence formulas for the gamma functions

$$\Gamma(z+1) = z\,\Gamma(z) \tag{3.16}$$

$$\Gamma(z+1, x) = z\,\Gamma(z, x) + x^z e^{-x}. \tag{3.17}$$

Using (3.16) we can rewrite the CDF (3.15) as

$$P(X \leq k; \lambda, \nu) = \frac{\lambda\Gamma\left(\frac{k}{\theta}, \frac{\lambda}{\theta}\right) - \theta\Gamma\left(\frac{k+\theta}{\theta}, \frac{\lambda}{\theta}\right)}{\Gamma\left(\frac{k}{\theta}\right)} + \frac{\theta\Gamma\left(\frac{k+\theta+1}{\theta}, \frac{\lambda}{\theta}\right) - \lambda\Gamma\left(\frac{k+1}{\theta}, \frac{\lambda}{\theta}\right)}{\Gamma\left(\frac{k+1}{\theta}\right)}. \tag{3.18}$$

Evaluating in $\theta = 1$, we have

$$P(X \leq k; \lambda, \nu) = \frac{\lambda\Gamma(k, \lambda) - \Gamma(k+1, \lambda)}{\Gamma(k)} + \frac{\Gamma(k+2, \lambda) - \lambda\Gamma(k+1, \lambda)}{\Gamma(k+1)}. \tag{3.19}$$

Applying the recurrence relation for the upper incomplete gamma function (3.17), we expand $\Gamma(k+1, \lambda)$ and $\Gamma(k+2, \lambda)$, ultimately resulting in

$$P(X \leq k; \lambda, 1) = \frac{\Gamma(k+1, \lambda)}{\Gamma(k+1)} = Q(k+1, \lambda),$$

that is then equal to the CDF (1.2), thus proving that when $\theta = 1$, the sequence defined in (3.14) leads to the Poisson distribution.

3.5.3 Verifying the requirements

To verify the resulting model is a valid discrete probability distribution, it is necessary to prove the requirements described in Section 2.2.1.

The first one, (2.12) can be proven by showing that the gamma distributions (3.11) are stochastically ordered by the mean for any fixed θ. For this, it is sufficient to see that the regularized gamma function is a strictly increasing function in its first parameter, meaning the survival function (3.12) of the gamma distribution is strictly increasing as the mean increases for values of $x > 0$. This is a known property of one-parameter exponential distributions and the gamma distribution, with a fixed dispersion parameter θ belongs to this family. This guarantees that the CDF of the Poisson generalization has values between 0 and 1. This does not guarantee that the CDF will increase in k.

The second property (2.13) is not trivial to prove because of the complex nature of the derivative of the regularized gamma function with respect to the first parameter. Proof that this is a valid probability distribution has been given by Hagmark (2012).

The third requirement,(2.14) is easily proven: Since $Q(k+1, \lambda)$ is the CDF of the Poisson distribution, its limit as $k \to \infty$ is 1, meaning the limit of $\mathcal{S}_k(\lambda, \theta)$ as $k \to \infty$ is

$$\lim_{k\to\infty} \mathcal{S}_k(\lambda, \theta) = \lim_{k\to\infty} \left(\lambda\, Q\left(\frac{k}{\theta}, \frac{\lambda}{\theta}\right) - k\, Q\left(\frac{k}{\theta} + 1, \frac{\lambda}{\theta}\right) + k \right) = \lambda - k + k = \lambda.$$

3.6 Properties

The proposed model has some useful properties that are not found together in any other currently known Poisson generalization. More properties have been described by Hagmark (2012).

3.6.1 Normalization

Since the cumulative distribution function (3.15) has limit 1 as $k \to \infty$, the resulting PMF is normalized.

3.6.2 Mean

Using formula (2.18), the expected value is

$$\mathbb{E}[X] = \lambda.$$

This can be useful since other generalizations such as the CMP do not have a closed form expression for the mean and require numerical methods or approximations.

3.6.3 Variance

In general, the variance for this model does not have a closed-form expression. Using the formula (2.22), a formula to compute it is

$$\mathrm{Var}(X) = \lambda + 2\sum_{i=1}^{\infty}[\lambda - \mathcal{S}_i(\lambda, \theta)] - \lambda^2 \tag{3.20}$$

The behavior of the variance in function of the mean for a fixed dispersion parameter θ is illustrated in Figure 3.1.

3.6.4 Functional independence of λ and θ

The parameters $\lambda > 0$ and $\theta > 0$ are functionally independent, meaning they can be varied independently without one being a function of the other.

3.6.5 Underdispersion

The main goal of this research was to find a way to generalize the Poisson distribution to allow full dispersion flexibility since, among known generalizations, only the CMP currently can do that. Then, the distribution (3.15) is the only currently known Poisson generalization with arbitrary dispersion and an ideal underdispersion behavior that is explicitly parametrized by the mean. Underdispersed examples of the PMF of this model are shown in Figure 3.2 An interactive plot has been implemented in Mathematica, the code is in Appendix B.

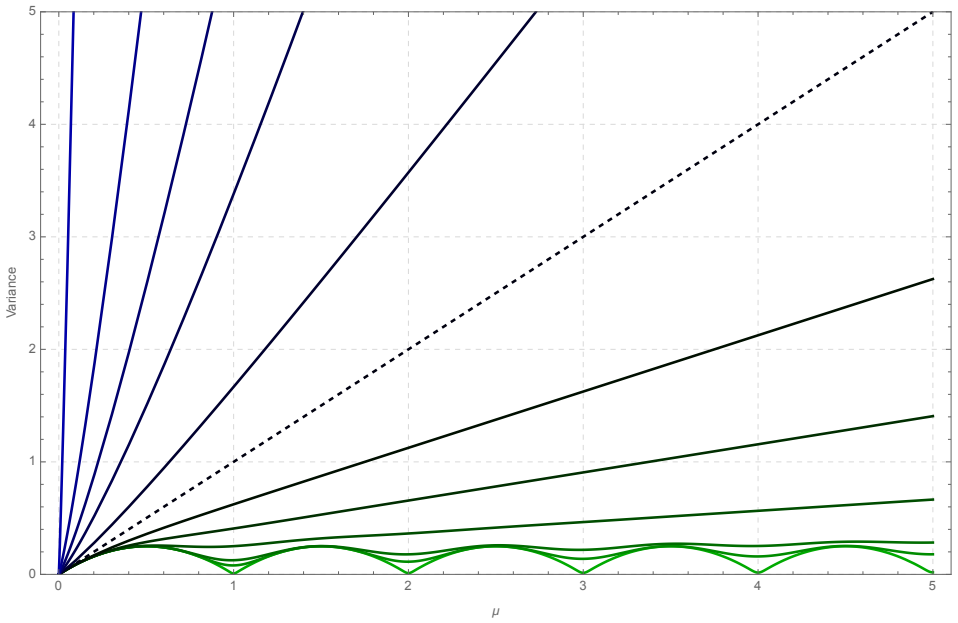


FIGURE 3.1: Variance in function of the mean μ for values of θ from 0.0001 (green) to 250 (blue), the dashed line corresponds to the Poisson case (equidispersion), the values of θ displayed are 0.0001, 0.01, 0.025, 0.1, 0.25, 0.5, 1, 2, 5, 10, 25 and 250.

3.6.6 Overdispersion

This model can still be used in cases of overdispersion when $\theta > 1$, the variance becomes greater than the mean. The distribution looks like a natural (unimodal) overdispersed count distribution, except at $k = 0$. In fact, as θ increases (it becomes more evident when $\theta > \lambda > 1$), the probability of zero increases more than expected, generating an overdispersed zero-inflated model. In many applications, this can actually be a positive thing, it often happens that real world overdispersed count data have zero inflation, and a model like this one can partially account for that using only 2 parameters. If this behavior is not desired, an elegant but limited solution to make the probability of zero look natural was found. This approach is discussed at the end of this chapter. Overdispersed examples of the PMF of this model are shown in Figure 3.3

3.6.7 Other notes

Unlike other generalizations such as the gamma counts distribution described in Sellers & Morris (2017), the generalization described in this chapter does not have the property of closure under the sum, with the exception of the Poisson case.

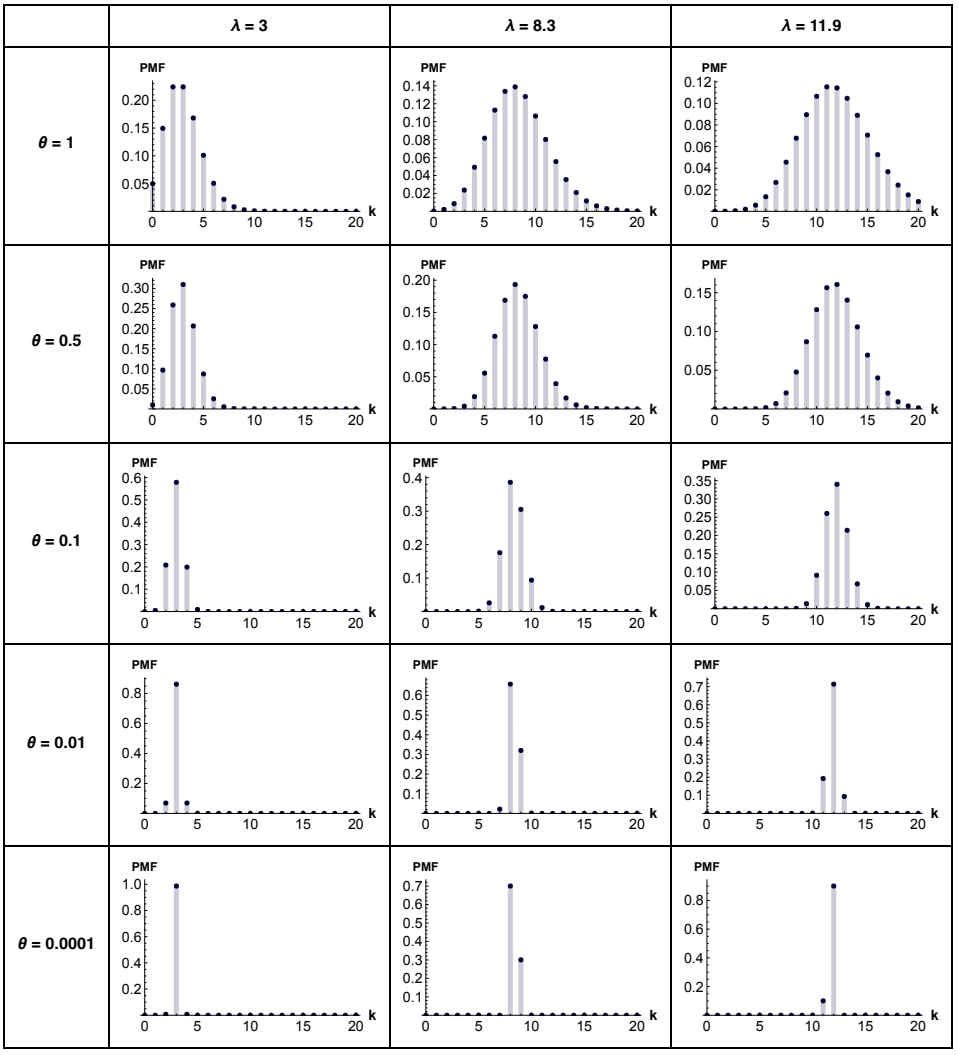


FIGURE 3.2: Underdispersed PMF for some choices of λ and θ

3.7 Regression modeling

The explicit mean parametrization of this distribution simplifies the construction and estimation of regression models. Analogous to the Poisson regression, we can use a logarithmic link function to relate the mean λ to a linear predictor in a regression model:

$$\log(\lambda_i) = \mathbf{z}_i^\top \boldsymbol{\beta}, \tag{3.21}$$

where $\mathbf{z}_i$ is the vector of covariates for observation i, and $\boldsymbol{\beta}$ is the vector of coefficients. This ensures $\lambda_i > 0$ and allows us to model the mean directly, facilitating interpretation and estimation. Software implementation in R will be described in Chapter 4.

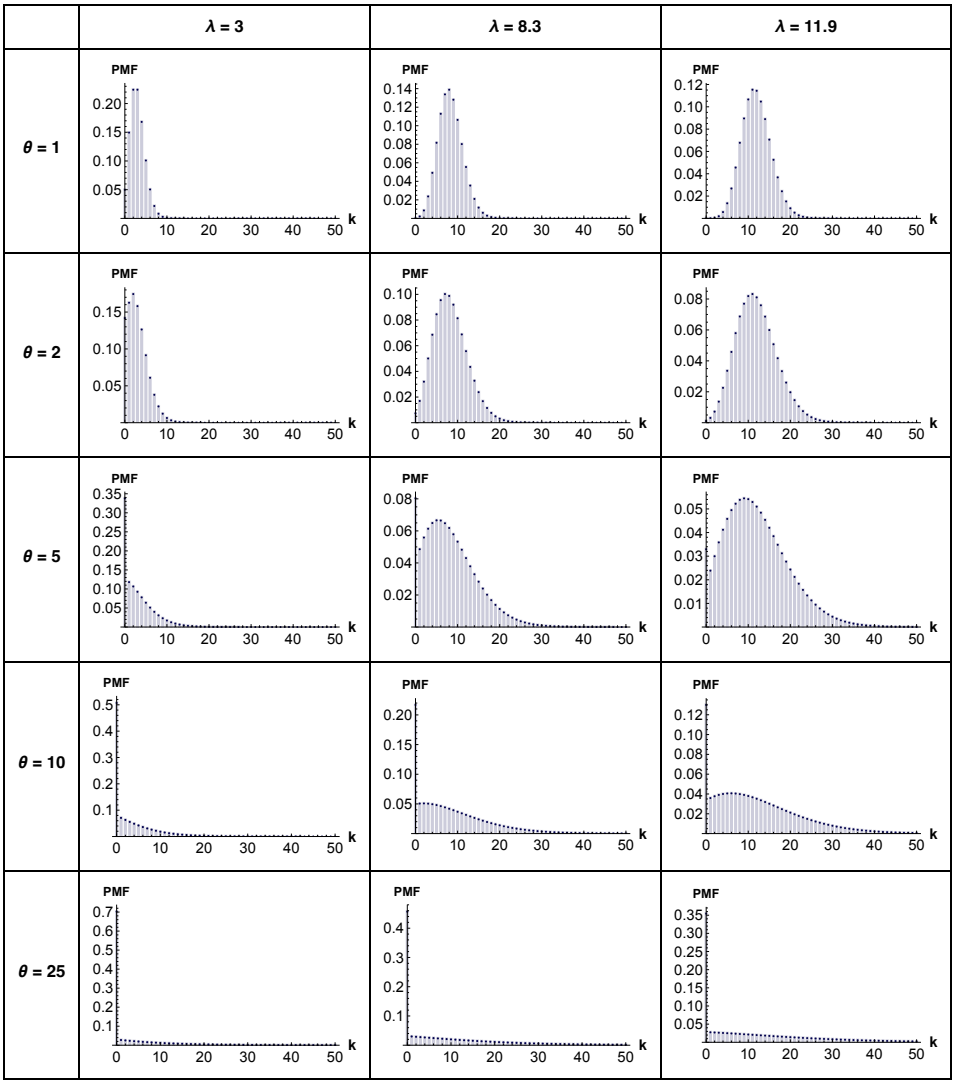


FIGURE 3.3: Overdispersed PMF for some choices of λ and θ. The probability in zero grows as θ grows for a fixed λ

3.8 Alternative possibilities for Poisson generalization

The solution presented is the most immediate, but following the same approach, more generalizations may be possible.

Other possibilities should follow from other generalizations of the Gamma distribution with scale equal to 1 that allow for arbitrary dispersion.

3.9 Correcting the zero-inflation for $\theta > 1$

The distribution found works well in all cases of underdispersion, but without adjustments, becomes a zero-inflated model when $\theta > 1$, and this behavior seems to be more evident when $\theta > \lambda$. The distribution of $X|X \neq 0$ looks like a natural overdispersed generalization of a Poisson distribution. We found an elegant way to correct this behavior for overdispersed cases that only changes the probability at zero, leaving the distribution of $X|X \neq 0$ unchanged. This was originally found by not defining correctly $\mathcal{S}_{-1}(\lambda, \theta)$ for negative k and applying the formula for the PMF (2.11) as usual.

$\mathcal{S}_{-1}(\lambda, \theta)$ was previously defined as equal to the integral of the survival function of a gamma distribution from 0 to -1, that is equal to -1. Instead we redefine the (3.14) when $k = -1$, by simply keeping the same formula used for positive integers

$$\mathcal{S}_k^B(\lambda, \theta) = \lambda\, Q\left(\frac{k}{\theta}, \frac{\lambda}{\theta}\right) - k\, Q\left(\frac{k}{\theta} + 1, \frac{\lambda}{\theta}\right) + k \tag{3.22}$$

valid for all $k \geq -1$, $\theta \geq 1$ $\lambda > 0$.

This formula, computed in $k = -1$ is then

$$\mathcal{S}_{-1}^B(\lambda, \theta) = \lambda\, Q\left(\frac{-1}{\theta}, \frac{\lambda}{\theta}\right) + 1\, Q\left(\frac{-1}{\theta} + 1, \frac{\lambda}{\theta}\right) - 1. \tag{3.23}$$

Applying the PMF formula (2.11) results in making the value of the PMF at zero equal to

$$-\mathcal{S}_{-1}^B(\lambda, \theta) - \mathcal{S}_1^B(\lambda, \theta), \tag{3.24}$$

instead of the usual

$$1 - \mathcal{S}_1(\lambda, \theta) = 1 - \mathcal{S}_1^B(\lambda, \theta). \tag{3.25}$$

This then has the effect of lowering the probability in zero by

$$1 - \mathcal{S}_1^B(\lambda, \theta) - (-\mathcal{S}_{-1}^B(\lambda, \theta) - \mathcal{S}_1^B(\lambda, \theta)) = 1 + \mathcal{S}_{-1}^B(\lambda, \theta). \tag{3.26}$$

Changing only the probability in zero then requires a constant of normalization that is obtained by redistributing the probability subtracted in zero to the other values

$$Z(\lambda, \theta) = 1 - (1 + \mathcal{S}_{-1}^B(\lambda, \theta)) = -\mathcal{S}_{-1}^B(\lambda, \theta). \tag{3.27}$$

We can use

$$\mathcal{S}_k^C(\lambda,\nu) = \frac{\mathcal{S}_k^B(\lambda,\theta)}{Z(\lambda,\theta)} = \frac{\lambda\, Q\left(\frac{k}{\theta},\frac{\lambda}{\theta}\right) - k\, Q\left(\frac{k}{\theta}+1,\frac{\lambda}{\theta}\right) + k}{1-\lambda\, Q\left(\frac{-1}{\theta},\frac{\lambda}{\theta}\right) - Q\left(\frac{-1}{\theta}+1,\frac{\lambda}{\theta}\right)} \tag{3.28}$$

to obtain a new normalized PMF

$$-\mathcal{S}_{k-1}^C(\lambda,\nu) + 2\mathcal{S}_k^C(\lambda,\nu) - \mathcal{S}_{k+1}^C(\lambda,\nu) \tag{3.29}$$

that is defined correctly for all $k \geq 0$, $\theta \geq 1$ $\lambda > 0$ with special case Poisson when $\theta = 1$. More research is needed to verify this point.

The PMF (3.29) is not valid in general when $\theta < 1$ because of the unstable behavior of the (3.23) when $\lambda < 1$, it may be valid when $\lambda \geq 1$.

The distribution (3.29) is not mean parametrized anymore; the mean is known and becomes

$$\mathbb{E}(X) = h(\lambda,\theta) = \frac{\lambda}{Z(\lambda,\theta)} = \frac{\lambda}{1-\lambda\, Q\left(\frac{-1}{\theta},\frac{\lambda}{\theta}\right) - Q\left(\frac{-1}{\theta}+1,\frac{\lambda}{\theta}\right)}. \tag{3.30}$$

Inverting this function is not trivial, but the mean-parametrization can be obtained numerically, noting that $h(\lambda,\theta)$ is strictly increasing in $\theta > 1$.

It is not clear if there is a mechanism that makes this correction work, other than that the quantity (3.24) is always lower than the quantity (3.25) when $\theta > 1$ and they coincide when $\theta = 1$, keeping the special case Poisson. It is unclear if there are deeper reasons why this correction looks natural. In fact, with this correction, graphically, the distribution seems to converge to a geometric distribution with the same mean as $\theta \to \infty$. The (3.29) only works in overdispersed cases; it is possible to define the PMF piecewise, separating the cases when $\theta > 1$ and $\theta < 1$. The likelihood would remain continuous but not be differentiable when $\theta = 1$. Examples of the PMF of this model are shown in figure 3.4. An interactive plot of this PMF has been implemented in Mathematica and the code is in Appendix B.

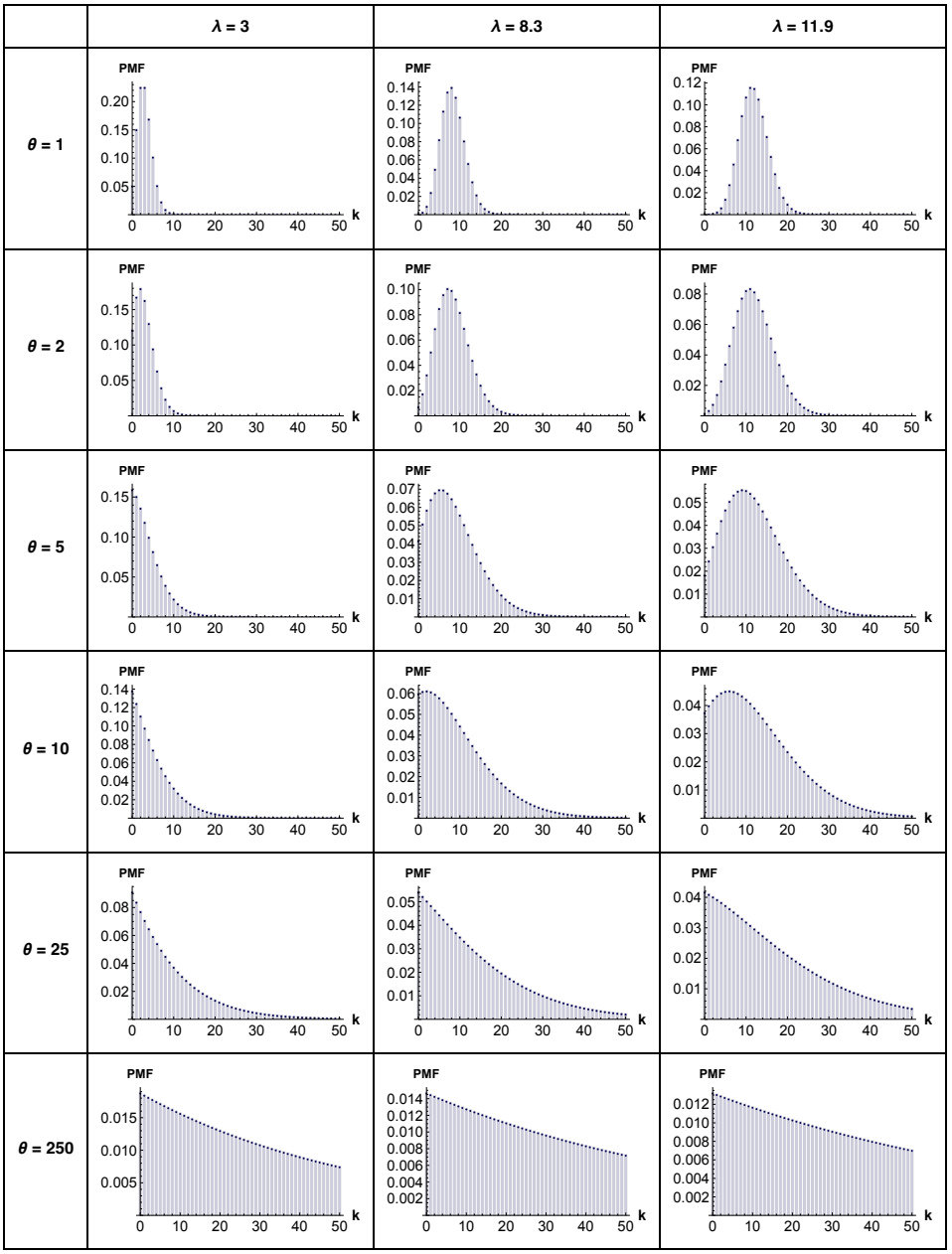


FIGURE 3.4: Zero-deflated, overdispersed PMF for some choices of λ and θ, here λ is not the mean

Chapter 4

Implementation in R

We implemented the model described in Chapter 3 within the R package `gdpoisson`, providing essential functions related to the distribution and the related linear regression model fitting function. This implementation allows users to simulate data, fit a regression model to observed counts, and perform inference using standard statistical techniques. R documentation is available for all functions. The distribution described in Chapter 3 will be called here Gamma-Difference-Poisson (GDPoisson). The package code is provided in Appendix D and some examples are shown in Appendix E.

4.1 Standard Functions

We implemented the four standard R functions for this distribution. In the first three functions, inputs k, λ, and θ can be scalars, vectors, or a mix of both as long as all vector inputs are of the same length.

4.1.1 Distribution function

The CDF `pgdpois` is calculated using this formula that is connected to the (3.13) and that can be obtained by applying the formulas (3.16) and (3.17).

$$\mathcal{S}_k(\lambda, \theta) - k = (k - \lambda)\left(1 - F\left(\lambda; \frac{k}{\theta},\ \theta\right)\right) + kf\left(\frac{\lambda}{\theta}; 1 + \frac{k}{\theta},\ 1\right) \tag{4.1}$$

Where the F and f are the CDF and PDF of the gamma distribution parametrized by shape (first parameter) and scale (second parameter), as defined in (3.3) and (3.1). This formula significantly reduces numerical instability due to catastrophic cancellation.

Catastrophic cancellation occurs when subtracting two nearly equal floating point values, leading to a loss of significant digits and thus reducing the accuracy of the result. However, numerical issues still arise in cases where the probability is very small.

It is possible to increase the precision of this calculation arbitrarily. A good way to do this in R would be to use the Python library `mpmath`. This library implements the function `gammainc` for the Regularized gamma function with arbitrary precision. There does not seem to be an efficient implementation in R. The main R library for arbitrary precision Rmpfr does not currently implement this function. Arbitrary precision could be useful to compute very small probability values since numerical stability cannot be improved using the logarithm of the probability mass function, as it is done in the standard Poisson case. We made an arbitrarily precise PMF function using Wolfram Mathematica, the code details are left in Appendix C.

4.1.2 Density function

The PMF `dgdpois` is not calculated as the difference of the `pgdpois` function, and instead is still calculated with (4.1), in order to avoid calculating twice the middle term of the second difference.

4.1.3 Quantile function

The quantile function `qgdpois` is simply calculated by iterating through the CDF until the required probability is found.

4.1.4 Simulating underdispersed counts

As Huang (2023) remarks, the ability to generate arbitrarily underdispersed counts is particularly useful for simulation studies. Underdispersed counts can be simulated using inverse transform sampling based on the cumulative distribution function (CDF) of the model. By generating uniform random numbers and applying the inverse CDF (quantile function), we obtain random variates that adhere to the specified distribution characteristics. Since the quantile function is not known, in `rgdpois` we iterate through the CDF until the required probability is found. This is not the most efficient implementation and can be improved using approximations and heuristic methods. An example would be using a binary search algorithm beginning from a reasonable guess of the quantile that we are looking for.

4.2 Regression Model

In this section, we introduce a regression model based on the proposed distribution from Chapter 3. Estimation is performed using standard maximum likelihood estimation techniques.

4.2.1 Assumptions

The GDPoisson regression, similar to those of generalized linear models:

1. **Independence of Observations:** The observations $Y_1, Y_2, \ldots, Y_n$ are independent random variables.

2. **Distribution of the Response Variable:** Each response variable Y_i follows the generalized Poisson distribution introduced in Chapter 3, with mean λ_i and dispersion parameter θ, that is,

$$Y_i \sim \text{GDPoisson}(\lambda_i, \theta), \quad i = 1, \ldots, n.$$

   The mean λ_i is related to the covariates $\mathbf{x}_i$ through a logarithmic link function:

$$\log(\lambda_i) = \mathbf{x}_i^\top \boldsymbol{\beta},$$

   where $\mathbf{x}_i \in \mathbb{R}^p$ is the vector of explanatory variables for observation i, and $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector of regression coefficients.

3. **Dispersion Parameter:** The dispersion parameter θ is assumed to be constant across observations, although it could be extended to vary with covariates, as discussed later.

4.2.2 Maximum Likelihood Estimation

The parameters of the model, $\boldsymbol{\beta}$ and θ, are estimated using maximum likelihood estimation (MLE). The likelihood function is continuously differentiable with respect to the parameters, which allows the use of gradient-based optimization methods such as the Newton-Raphson algorithm.

The log-likelihood function for the observed data $(k_1, k_2, \ldots, k_n)^\top$ is given by:

$$\ell(\boldsymbol{\beta}, \theta) = \sum_{i=1}^{n} \log P(Y_i = k_i; \lambda_i, \theta),$$

where $P(Y_i = k_i; \lambda_i, \theta)$ is the PMF of the GDPoisson distribution evaluated at k_i, the observed count for observation i. In practice, we minimize the negative log-likelihood function:

$$-\ell(\boldsymbol{\beta}, \theta) = -\sum_{i=1}^{n} \log P(Y_i = k_i; \lambda_i, \theta).$$

In our implementation, initial estimates for $\boldsymbol{\beta}$ are obtained from a standard Poisson regression using the `glm` function with a log-link, which provides efficient and numerically stable starting values. For the dispersion parameter θ, we use an initial guess equal to the Fisher dispersion index estimated from the Poisson regression residuals. This index is calculated as

$$\hat{\theta} = \frac{1}{n-p} \sum_{i=1}^{n} \frac{(k_i - \hat{\lambda}_i)^2}{\hat{\lambda}_i},$$

where:

- k_i are the observed counts,
- $\hat{\lambda}_i$ are the fitted values from the initial Poisson regression,
- n is the number of observations and
- p is the number of estimated parameters in the model.

This initial estimate corresponds to the dispersion parameter found in quasi-Poisson modeling. Using it as the initial guess is appropriate because it is generally close to the dispersion parameter of the GDPoisson, or at least their logarithms have the same sign, which aids in the convergence of the optimization algorithm.

Optimization is carried out using the `optim` function in R, which minimizes the negative log-likelihood function. Various optimization methods are available and can be specified via the `method` argument.

4.2.3 Dispersion Parameter

To ensure $\theta > 0$ during optimization, we model it on a logarithmic scale, defining $\theta = e^{\gamma}$ where $\gamma \in \mathbb{R}$. This transformation enables unconstrained optimization of γ in $\mathbb{R}$.

4.2.4 Standard Errors

Standard errors of the parameter estimates can be obtained from the observed information matrix or from its expected value, the Fisher information matrix, calculated in the

maximum likelihood estimate. These quantities are asymptotically equivalent. Here, we use the observed information matrix. Specifically, the observed information matrix is given by the Hessian matrix:

$$\mathbf{J}_{ij}(\boldsymbol{\phi}) = \frac{\partial^2(-\ell(\boldsymbol{\phi}))}{\partial\phi_i\partial\phi_j},$$

where ϕ_i and ϕ_j are elements of the parameter vector $\boldsymbol{\phi} = (\boldsymbol{\beta}, \gamma)^\top$. The estimated covariance matrix of the parameter estimates is then obtained by inverting the observed Fisher information matrix evaluated at the MLE $\hat{\boldsymbol{\phi}}$

$$\mathbf{V} = \mathbf{J}^{-1}(\hat{\boldsymbol{\phi}}).$$

The standard errors are the square roots of the diagonal elements of the covariance matrix evaluated at the MLE

$$\mathrm{SE}(\phi_i) = \sqrt{\mathbf{V}_{ii}}.$$

These standard errors provide measures of the precision of the parameter estimates. They are used for hypothesis testing, usually to check if regression parameters differ significantly from zero.

4.2.5 Other quantities

The Akaike Information Criterion (AIC) is calculated using the standard formula:

$$\mathrm{AIC} = 2\,p - 2\ell(\hat{\boldsymbol{\phi}}),$$

where p is the number of parameters estimated, and $\ell(\hat{\boldsymbol{\phi}})$ is log-likelihood value computed at the MLE.

The residual deviance and null deviance are computed as twice the difference between the log-likelihood of the saturated model and, respectively, the current model and the null model. Specifically,

$$\text{Residual Deviance} = 2(\ell_{\text{saturated}} - \ell_{\text{current}}),$$

$$\text{Null Deviance} = 2(\ell_{\text{saturated}} - \ell_{\text{null}}).$$

The saturated model assumes that the dispersion is 0 since all means are integers equal to the observed values. For this reason, the log-likelihood of the saturated model is always 0, since the likelihood of the shifted Bernoulli is equal to 1 when the mean is

equal to k and $\log(1) = 0$. The null model likelihood is calculated at the maximum likelihood estimate (MLE) of a model with only an intercept, which also increases the estimated dispersion parameter. The residual deviance is typically far from zero even if the fit is good because of the discrete nature of the model. This value can still be used to test if two nested models are significantly different by using the difference in their residual deviances to perform a likelihood-ratio test.

4.2.6 Possible further developments

Future improvements to the model could include:

- **Double Generalized Linear Model:** As seen in Huang (2023), we can allow the dispersion parameter θ_i to vary with covariates by introducing a separate linear predictor. A logarithmic link function for the dispersion parameter ensures positivity

$$\log(\theta_i) = \mathbf{z}_i^\top \gamma,$$

  where $\mathbf{z}_i$ is a vector of covariates for observation i related to dispersion, and γ is the vector of coefficients for the dispersion model.

- **Zero-Inflation or Zero-Deflation Modeling:** Introducing a parameter to control zero-inflation or zero-deflation, accommodating datasets with an excess or deficiency of zero counts. This allows for the modification of the probability mass function to adjust the probability in zero.

4.3 Conclusions

Examples comparing the fit of over and underdispersed count datasets with other regression models in R are provided in Appendix E. These examples show that the proposed model consistently achieves a lower (AIC) than the standard Poisson model for both overdispersed and underdispersed data, indicating a better fit. This outcome aligns with expectations and is common to many generalized Poisson models.

A primary advantage of the proposed model is its ability to provide more accurate standard error estimates by using a probability distribution that is also capable of achieving arbitrary dispersion for any mean. In contrast, methods like quasi-Poisson regression lack an underlying probability distribution, limiting their interpretability.

Appendix A

Geometric distribution linked function

```
p[μ_] := 1 / (μ + 1)
```

```
GeometricCDF[k_, μ_] := 1 - (1 - p[μ]) ^ (k + 1)
GeometricPDF[k_, μ_] := (1 - p[μ]) ^ (k) * p[μ]
GeometricPDF[k, μ]
```

$$\frac{\left(1 - \frac{1}{1+\mu}\right)^{k}}{1 + \mu}$$

```
GeometricSURV[k_, μ_] := 1 - GeometricCDF[k, μ]
```

```
SkIniziale[k_, μ_] := Sum[GeometricSURV[i, μ], {i, 0, k}]
```

```
SkIniziale[k, μ]
```

$$-\frac{\mu\left(-1 - \mu + \mu\left(1 - \frac{1}{1+\mu}\right)^{k}\right)}{1 + \mu}$$

```
N[SkIniziale[40, 4]]
```

```
3.99957
```

```
(*change the sign since we are deriving a survival function*)
```

```
DDSSgeom[k_, μ_] = -Simplify[D[D[SkIniziale[k, μ], μ], μ]]
```

$$\frac{\left(2 + 3\,k + k^{2}\right)\left(\frac{\mu}{1+\mu}\right)^{k}}{(1 + \mu)^{3}}$$

```
DDSSgeom[μ - 1, x]
```

$$\frac{\left(\frac{x}{1+x}\right)^{-1+\mu}\left(2 + 3\,(-1 + \mu) + (-1 + \mu)^{2}\right)}{(1 + x)^{3}}$$

```
Integrate[DDSSgeom[k, μ], {μ, 0, Infinity}, Assumptions → {μ > 0, k > 0}]
```

```
1
```

```
Integrate[μ * DDSSgeom[k, μ], {μ, 0, Infinity}, Assumptions → {μ > 0, k > 0}]
```

$$1 + k$$

Appendix B

Interactive plot in Mathematica

2]:=

$$-\left(\begin{cases} -1+k-(-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\mu}{\theta}\right] & -1+k>0 \\ -1+k & \text{True} \end{cases}\right)$$

$$\left(\begin{cases} 1+k-(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right] & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

62]=

$$-\left(\begin{cases} -1+k-(-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\mu}{\theta}\right] & -1+k>0 \\ -1+k & \text{True} \end{cases}\right)$$

$$\left(\begin{cases} 1+k-(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right] & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

3]:=

```
(*steps for derivation of the PMF*)
```

4]:=

```
gammaPDFalpha[x_, α_, θ_] =
 Piecewise[{{(x^(α - 1) Exp[-x / θ]) / (θ^α Gamma[α]), x > 0}, {0, x ≤ 0}}]
```

64]=

$$\begin{cases} \frac{e^{-\frac{x}{\theta}}\,x^{-1+\alpha}\,\theta^{-\alpha}}{\text{Gamma}[\alpha]} & x>0 \\ 0 & \text{True} \end{cases}$$

5]:=

```
gammaCDFalpha[x_, α_, θ_] =
 Piecewise[{{ 1 - Gamma[α, x / θ] / Gamma[α], x > 0}, {0, x ≤ 0}}]
```

65]=

$$\begin{cases} 1-\frac{\text{Gamma}\left[\alpha,\frac{x}{\theta}\right]}{\text{Gamma}[\alpha]} & x>0 \\ 0 & \text{True} \end{cases}$$

5]:=

```
(*mean parametrized gamma*)
(*mean k*)
```

7]:=

```
gammaPDF[x_, k_, θ_] = gammaPDFalpha[x, k / θ, θ]
```

67]=

$$\begin{cases} \frac{e^{-\frac{x}{\theta}}\,x^{-1+\frac{k}{\theta}}\,\theta^{-\frac{k}{\theta}}}{\text{Gamma}\left[\frac{k}{\theta}\right]} & x>0 \\ 0 & \text{True} \end{cases}$$

3]:=

```
gammaCDFmu[x_, k_, θ_] = gammaCDFalpha[x, k / θ, θ]
```

68]=

$$\begin{cases} 1-\frac{\text{Gamma}\left[\frac{k}{\theta},\frac{x}{\theta}\right]}{\text{Gamma}\left[\frac{k}{\theta}\right]} & x>0 \\ 0 & \text{True} \end{cases}$$

```
gammaSURVmu[x_, k_, θ_] = Simplify[1 - gammaCDFmu[x, k, θ]]
```

$$\begin{cases} 1 & x \le 0 \\ \dfrac{\text{Gamma}\left[\frac{k}{\theta}, \frac{x}{\theta}\right]}{\text{Gamma}\left[\frac{k}{\theta}\right]} & \text{True} \end{cases}$$

```
SkNOTSimplified[μ_, k_, θ_] =
 Piecewise[{{Simplify[Integrate[gammaSURVmu[z, k, θ], {z, 0, μ},
       Assumptions → {μ > 0, θ > 0, k ≥ 0}]], k > 0}, {k, k ≤ 0}}]
```

$$\begin{cases} k + \dfrac{\theta\left(-e^{-\frac{\mu}{\theta}}\theta^{-\frac{k}{\theta}}\mu^{\frac{k+\theta}{\theta}} + (-k+\mu)\,\text{Gamma}\left[\frac{k+\theta}{\theta}, \frac{\mu}{\theta}\right]\right)}{k\,\text{Gamma}\left[\frac{k}{\theta}\right]} & k > 0 \\ k & k \le 0 \\ 0 & \text{True} \end{cases}$$

```
(*manually applying gamma property*)
```

```
Sk[μ_, k_, θ_] =
```

$$\begin{cases} k + \dfrac{\left(-e^{-\frac{\mu}{\theta}}\theta^{-\frac{k}{\theta}}\mu^{\frac{k+\theta}{\theta}} + (-k+\mu)\,\text{Gamma}\left[\frac{k+\theta}{\theta}, \frac{\mu}{\theta}\right]\right)}{\text{Gamma}\left[\frac{k}{\theta}+1\right]} & k > 0 \\ k & k \le 0 \\ 0 & \text{True} \end{cases}$$

$$\begin{cases} k + \dfrac{-e^{-\frac{\mu}{\theta}}\theta^{-\frac{k}{\theta}}\mu^{\frac{k+\theta}{\theta}} + (-k+\mu)\,\text{Gamma}\left[\frac{k+\theta}{\theta}, \frac{\mu}{\theta}\right]}{\text{Gamma}\left[1+\frac{k}{\theta}\right]} & k > 0 \\ k & k \le 0 \\ 0 & \text{True} \end{cases}$$

```
(*CDF OF THE MODEL*)
```

```
SURVcomplex[k_, μ_, θ_] = Sk[μ, k + 1, θ] - Sk[μ, k, θ]
```

$$-\left(\begin{cases} k + \dfrac{-e^{-\frac{\mu}{\theta}}\theta^{-\frac{k}{\theta}}\mu^{\frac{k+\theta}{\theta}} + (-k+\mu)\,\text{Gamma}\left[\frac{k+\theta}{\theta}, \frac{\mu}{\theta}\right]}{\text{Gamma}\left[1+\frac{k}{\theta}\right]} & k > 0 \\ k & k \le 0 \\ 0 & \text{True} \end{cases}\right) +$$

$$\left(\begin{cases} 1 + k + \dfrac{-e^{-\frac{\mu}{\theta}}\theta^{-\frac{1+k}{\theta}}\mu^{\frac{1+k+\theta}{\theta}} + (-1-k+\mu)\,\text{Gamma}\left[\frac{1+k+\theta}{\theta}, \frac{\mu}{\theta}\right]}{\text{Gamma}\left[1+\frac{1+k}{\theta}\right]} & 1 + k > 0 \\ 1 + k & 1 + k \le 0 \\ 0 & \text{True} \end{cases}\right)$$

```
PMFcomplex[k_, μ_, θ_] =
 FullSimplify[SURVcomplex[k - 1, μ, θ] - SURVcomplex[k, μ, θ],
  Assumptions → {k ≥ -1, μ > 0, θ > 0}]
```

$$-\left(\begin{cases} -1+k+\dfrac{-e^{-\frac{\mu}{\theta}}\theta^{\frac{1-k}{\theta}}\mu^{\frac{-1+k+\theta}{\theta}}+(1-k+\mu)\,\text{Gamma}\left[\frac{-1+k+\theta}{\theta},\frac{\mu}{\theta}\right]}{\text{Gamma}\left[\frac{-1+k+\theta}{\theta}\right]} & k>1 \\ -1+k & \text{True} \end{cases}\right)+$$

$$2\left(\begin{cases} k+\dfrac{-e^{-\frac{\mu}{\theta}}\theta^{-\frac{k}{\theta}}\mu^{\frac{k+\theta}{\theta}}+(-k+\mu)\,\text{Gamma}\left[\frac{k+\theta}{\theta},\frac{\mu}{\theta}\right]}{\text{Gamma}\left[\frac{k+\theta}{\theta}\right]} & k>0 \\ k & \text{True} \end{cases}\right)-$$

$$\left(\begin{cases} 1+k+\dfrac{-e^{-\frac{\mu}{\theta}}\theta^{-\frac{1+k}{\theta}}\mu^{\frac{1+k+\theta}{\theta}}+(-1-k+\mu)\,\text{Gamma}\left[\frac{1+k+\theta}{\theta},\frac{\mu}{\theta}\right]}{\text{Gamma}\left[\frac{1+k+\theta}{\theta}\right]} & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

```
(*Seems complicated, it can be simplified more, manually*)
```

```
SkSimple[μ_, k_, θ_] = Piecewise[
   {{k - k GammaRegularized[1 + k/θ, μ/θ] + μ GammaRegularized[k/θ, μ/θ], k > 0}, {k, True}}]
```

$$\begin{cases} k-k\,\text{GammaRegularized}\left[1+\frac{k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{k}{\theta},\frac{\mu}{\theta}\right] & k>0 \\ k & \text{True} \end{cases}$$

```
SURVSimple[k_, μ_, θ_] = SkSimple[μ, k + 1, θ] - SkSimple[μ, k, θ]
```

$$-\left(\begin{cases} k-k\,\text{GammaRegularized}\left[1+\frac{k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{k}{\theta},\frac{\mu}{\theta}\right] & k>0 \\ k & \text{True} \end{cases}\right)+$$

$$\left(\begin{cases} 1+k-(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right] & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

```
PMFSimple[k_, μ_, θ_] = SURVSimple[k - 1, μ, θ] - SURVSimple[k, μ, θ]
```

$$-\left(\begin{cases} -1+k-(-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+ & -1+k> \\ \mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\mu}{\theta}\right] & \\ -1+k & \text{True} \end{cases}\right)$$

$$\left(\begin{cases} 1+k-(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right] & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

2]:=

3]:=

```
PMFsimple[k_, μ_, θ_] =
```

$$-\left(\begin{cases} -1+k-(-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\mu}{\theta}\right] & -1+k \\ -1+k & \text{True} \end{cases}\right)$$

$$\left(\begin{cases} 1+k-(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right] & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

```
(*PLOT*)
Manipulate[Module[{genPoisValues, sumValues, mean, variance},

  genPoisValues = Quiet[Table[PMFsimple[x, μ, θ], {x, 0, 4 * (μ + θ)}]];
  mean = Total[Table[x * genPoisValues⟦x + 1⟧, {x, 0, 4 * (μ + θ)}]];
  variance = Total[Table[(x - mean) ^ 2 * genPoisValues⟦x + 1⟧, {x, 0, 4 * (μ + θ)}]];
  Column[
   {DiscretePlot[{genPoisValues⟦x + 1⟧}, {x, 0, 2 * (μ + θ)}, PlotRange → {0, 1.01},
     PlotStyle → PointSize[Large], Filling → Axis, ImageSize → Large],
    Row[{"μ = ", N[μ, {5, 3}], "    θ = ", NumberForm[θ, {2, 2}]}],
    Row[{"Mean (E[x]): ", N[mean, {2, 2}]}],
    Row[{"Variance: ", N[variance, {2, 2}]}],
    Row[{"GenPois values for x = 0 to 4*(μ+θ): ",
      N[genPoisValues⟦1 ;; Min[{Floor[4 * (μ + θ)], 15}]⟧, {5, 3}]}]}]],
 {{μ, 1}, 0.1, 25}, {{θ, 1}, 0.01, 9}]
```

33]=

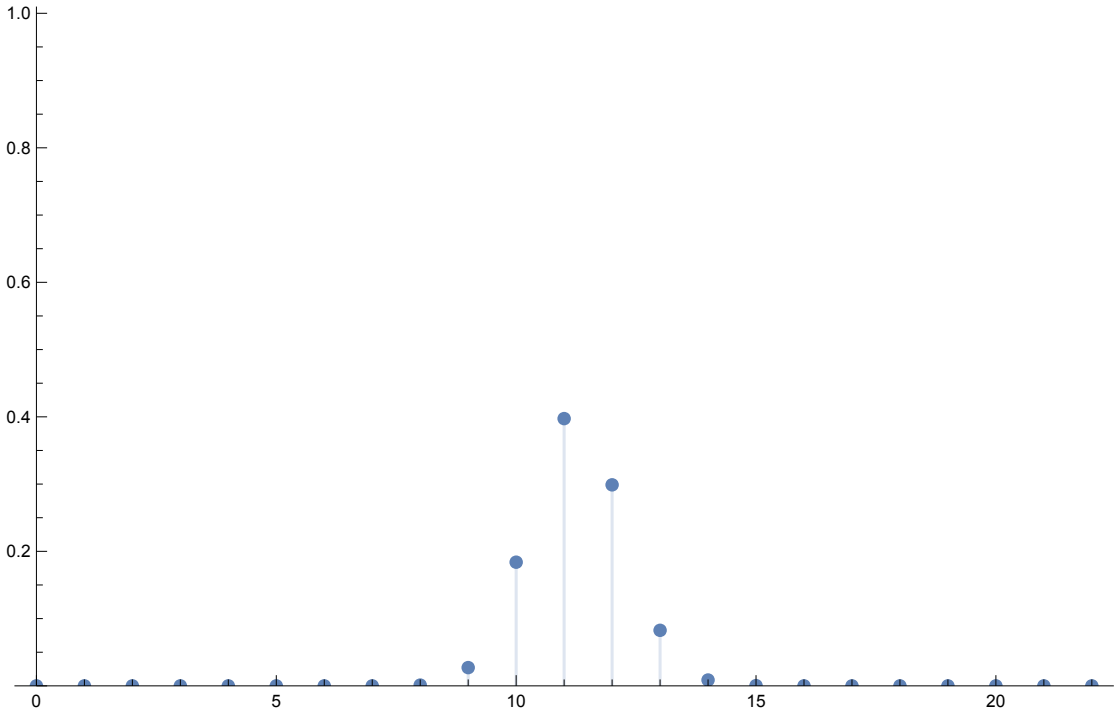
$$-\left(\begin{cases} -1+k-(-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\mu}{\theta}\right] & -1+k> \\ -1+k & \text{True} \end{cases}\right)$$

$$\left(\begin{cases} 1+k-(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right] & 1+k>0 \\ 1+k & \text{True} \end{cases}\right)$$

34]=

μ 11.25

θ 0.07



μ = 11.25 θ = 0.07

Mean (E[x]): 11.25

Variance: 0.95335

GenPois values for x = 0 to 4*(μ+θ):

{0., 0., 0., 0., 0., 4.64428×10^{-12}, 1.89734×10^{-8}, 0.0000107245, 0.00111842, 0.0271359, 0.183867, 0.397366, 0.298885, 0.0825683, 0.00868386}

5]:=

5]:=

```
(*ZERO INFLATION CORRECTION FOR OVERDISPERSED CASES*)
```

```
SkPURE[μ_, k_, θ_] = k - k GammaRegularized[1 + k/θ, μ/θ] + μ GammaRegularized[k/θ, μ/θ]
```

```
SkPURE[μ, k, θ]
```

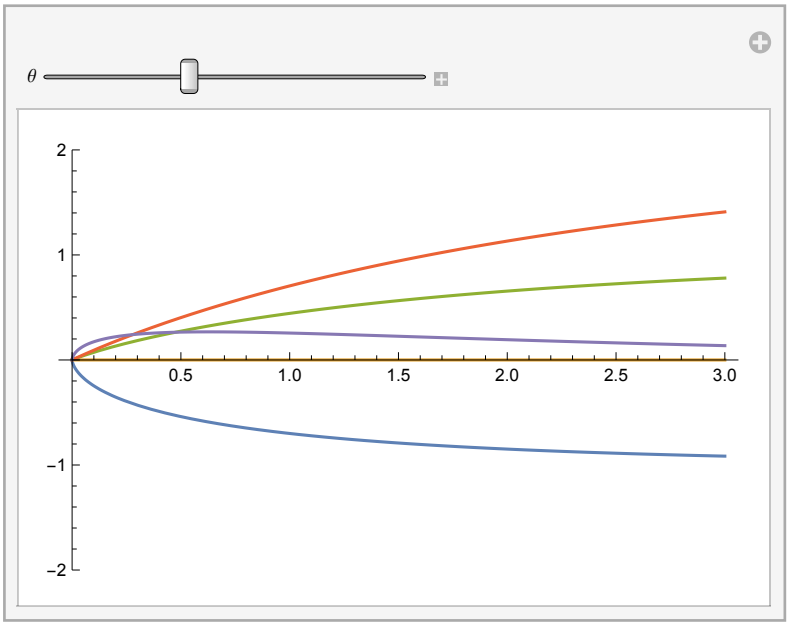
$$k - k\,\text{GammaRegularized}\left[1 + \frac{k}{\theta}, \frac{\mu}{\theta}\right] + \mu\,\text{GammaRegularized}\left[\frac{k}{\theta}, \frac{\mu}{\theta}\right]$$

$$k - k\,\text{GammaRegularized}\left[1 + \frac{k}{\theta}, \frac{\mu}{\theta}\right] + \mu\,\text{GammaRegularized}\left[\frac{k}{\theta}, \frac{\mu}{\theta}\right]$$

```
CORRECTEDPROBABILITY[μ_, θ_] = -SkPURE[μ, -1, θ] - SkPURE[μ, 1, θ]
```

$$-\text{GammaRegularized}\left[1 - \frac{1}{\theta}, \frac{\mu}{\theta}\right] + \text{GammaRegularized}\left[1 + \frac{1}{\theta}, \frac{\mu}{\theta}\right] - \mu\,\text{GammaRegularized}\left[-\frac{1}{\theta}, \frac{\mu}{\theta}\right] - \mu\,\text{GammaRegularized}\left[\frac{1}{\theta}, \frac{\mu}{\theta}\right]$$

```
Manipulate[
 Plot[{SkPURE[μ, -1, θ], SkPURE[μ, 0, θ], SkPURE[μ, 1, θ], SkPURE[μ, 2, θ],
   CORRECTEDPROBABILITY[μ, θ]}, {μ, 0, 3}, PlotRange → {-2, 2}], {{θ, 3}, 0.1, 8}]
```



```
Deflated[k_, μ_, θ_] = -SkPURE[μ, k - 1, θ] + 2 SkPURE[μ, k, θ] - SkPURE[μ, k + 1, θ]
```

$$-2\,k + (-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta}, \frac{\mu}{\theta}\right] +$$
$$(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta}, \frac{\mu}{\theta}\right] - \mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta}, \frac{\mu}{\theta}\right] +$$
$$2\left(k - k\,\text{GammaRegularized}\left[1+\frac{k}{\theta}, \frac{\mu}{\theta}\right] + \mu\,\text{GammaRegularized}\left[\frac{k}{\theta}, \frac{\mu}{\theta}\right]\right) -$$
$$\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta}, \frac{\mu}{\theta}\right]$$

```
(*cannot be used for underdispersed cases since
 for non integer values of 1/theta it becomes unstable *)

Deflated[0, 0.1, 0.23]
```

$$-0.519027$$

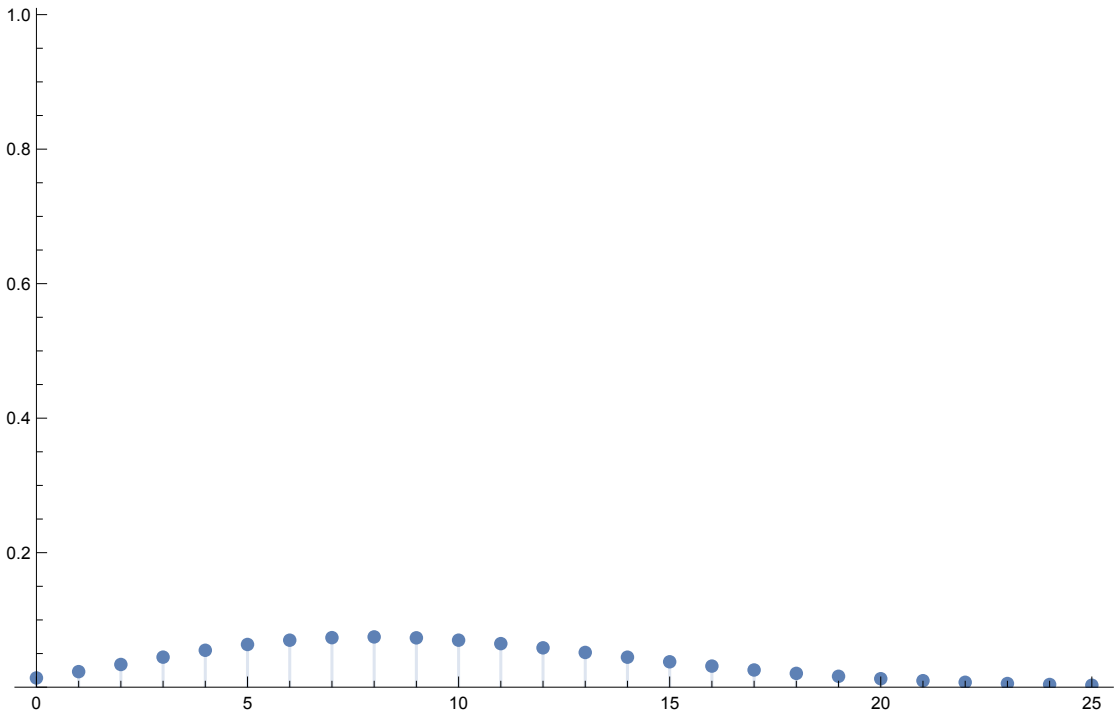
```
(*for this reason this works only on overdispersed cases*)

Manipulate[Module[{genPoisValues, sumValues, mean, variance},
  genPoisValues = Quiet[Table[Deflated[x, μ, θ], {x, 0, 4 * (μ + θ) + 3}]];
  mean = Total[Table[x * genPoisValues[[x + 1]], {x, 0, 4 * (μ + θ)}]];
  variance =
   Total[Table[(x - mean)^2 * genPoisValues[[x + 1]], {x, 0, Floor[4 * (μ + θ) + 1]}]];
  Column[
   {DiscretePlot[{genPoisValues[[x + 1]]}, {x, 0, 2 * (μ + θ)}, PlotRange → {0, 1.01},
     PlotStyle → PointSize[Large], Filling → Axis, ImageSize → Large],
    Row[{"μ = ", N[μ, {5, 3}], "    θ = ", NumberForm[θ, {5, 3}]}],
    Row[{"Mean (E[x]): ", N[mean, {5, 5}]}],
    Row[{"Variance: ", N[variance, {5, 3}]}],
    Row[{"GenPois values for x = 0 to 4*(μ+θ): ",
      N[genPoisValues[[1 ;; Min[{Floor[4 * (μ + θ) + 1], 15}]]], {5, 3}]}]}]],
 {{μ, 1}, 0.1, 25}, {{θ, 1}, 0.01, 9}]
```

00]=

μ 9.65

θ 3.19



μ = 9.65 Θ = 3.190

Mean (E[x]): 9.65

Variance: 28.814

GenPois values for x = 0 to 4*(μ+Θ):

{0.0137354, 0.0231229, 0.0337604, 0.0446817, 0.0548883,
0.0635029, 0.0698793, 0.0736579, 0.0747689, 0.0733932,
0.0698987, 0.0647655, 0.0585147, 0.0516499, 0.0446141}

1]:=

2]:=

3]:=

```
(*we create a correct PMF normalizing the pmf*)
```

```
DeflatedPMF[k_, μ_, θ_] = Deflated[k, μ, θ] / (-SkPURE[μ, -1, θ])
```

$$\left(-2\,k+(-1+k)\,\text{GammaRegularized}\left[1+\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+(1+k)\,\text{GammaRegularized}\left[1+\frac{1+k}{\theta},\frac{\mu}{\theta}\right]-\mu\,\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\mu}{\theta}\right]+2\left(k-k\,\text{GammaRegularized}\left[1+\frac{k}{\theta},\frac{\mu}{\theta}\right]+\mu\,\text{GammaRegularized}\left[\frac{k}{\theta},\frac{\mu}{\theta}\right]\right)-\mu\,\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\mu}{\theta}\right]\right)\Big/\left(1-\text{GammaRegularized}\left[1-\frac{1}{\theta},\frac{\mu}{\theta}\right]-\mu\,\text{GammaRegularized}\left[-\frac{1}{\theta},\frac{\mu}{\theta}\right]\right)$$

```
FullSimplify[DeflatedPMF[k, λ, θ]]
```

$$\left(\lambda\left(\text{GammaRegularized}\left[\frac{-1+k}{\theta},\frac{\lambda}{\theta}\right]-2\,\text{GammaRegularized}\left[\frac{k}{\theta},\frac{\lambda}{\theta}\right]+\text{GammaRegularized}\left[\frac{1+k}{\theta},\frac{\lambda}{\theta}\right]\right)-(-1+k)\,\text{GammaRegularized}\left[\frac{-1+k+\theta}{\theta},\frac{\lambda}{\theta}\right]+2\,k\,\text{GammaRegularized}\left[\frac{k+\theta}{\theta},\frac{\lambda}{\theta}\right]-(1+k)\,\text{GammaRegularized}\left[\frac{1+k+\theta}{\theta},\frac{\lambda}{\theta}\right]\right)\Big/\left(-1+\lambda\,\text{GammaRegularized}\left[-\frac{1}{\theta},\frac{\lambda}{\theta}\right]+\text{GammaRegularized}\left[\frac{-1+\theta}{\theta},\frac{\lambda}{\theta}\right]\right)$$

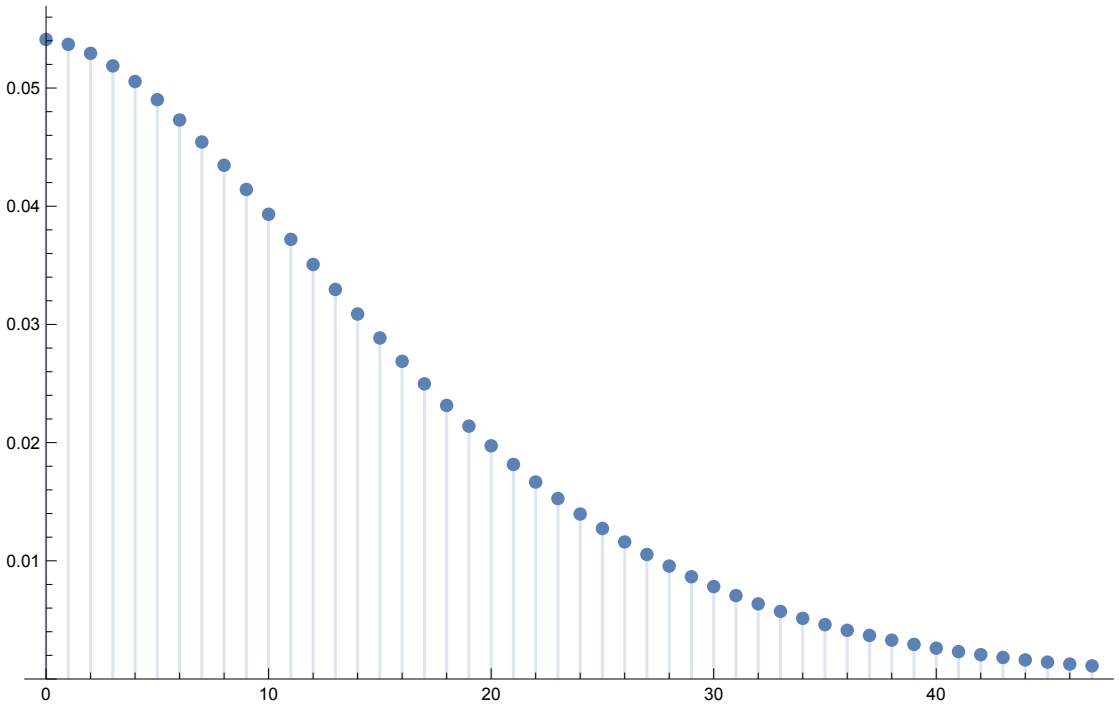
```
(*PLOT OF THE ZERO INFLATION ADJUSTED DISTRIBUTION,
normalized, this PMF is stable only for theta>1 or lambda>1*)

Manipulate[Module[{genPoisValues, sumValues, mean, variance, sum},
  genPoisValues = Quiet[Table[DeflatedPMF[x, μ, θ], {x, 0, 4 * (μ + θ) + 3}]];
  mean = Total[Table[x * genPoisValues[[x + 1]], {x, 0, 4 * (μ + θ)}]];
  sum = Total[Table[genPoisValues[[x + 1]], {x, 0, 4 * (μ + θ)}]];
  variance =
   Total[Table[(x - mean) ^ 2 * genPoisValues[[x + 1]], {x, 0, Floor[4 * (μ + θ) + 1]}]];
  Column[
   {DiscretePlot[{genPoisValues[[x + 1]]}, {x, 0, 2 * (μ + θ)}, PlotRange → {0, All},
     PlotStyle → PointSize[Large], Filling → Axis, ImageSize → Large],
    Row[{"μ = ", N[μ, {5, 3}], "    θ = ", NumberForm[θ, {5, 3}]}],
    Row[{"sum : ", N[sum, {5, 5}]}],
    Row[{"Mean (E[x]): ", N[mean, {5, 5}]}],
    Row[{"Variance: ", N[variance, {5, 3}]}],
    Row[{"GenPois values for x = 0 to 4*(μ+θ): ",
      N[genPoisValues[[1 ;; Min[{Floor[4 * (μ + θ) + 1], 15}]]], {5, 3}]}]}]],
 {{μ, 1}, 0.1, 25}, {{θ, 1}, 0.01, 90}]
```

07]=

μ 9.35

θ 14.6



μ = 9.35 Θ = 14.600
sum : 0.999995
Mean (E[x]): 12.2682
Variance: 108.941
GenPois values for x = 0 to 4*(μ+Θ):
{0.0541112, 0.0536961, 0.0529359, 0.0518742, 0.0505535,
0.0490148, 0.0472968, 0.0454361, 0.0434663, 0.0414184,
0.0393202, 0.0371968, 0.0350702, 0.0329598, 0.030882}

8]:=

9]:=

0]:=

1]:=

```
(*function that computes the variance numerically*)
```

2]:=

```
VAR[μ_, θ_] := μ + 2 * NSum[μ - SkSimple[μ, k, θ], {k, 1, Infinity}] - μ^2
```

Appendix C

Computing the PMF with arbitrary precision in Wolfram Mathematica

```
GammaPDFBuiltin[x_, α_, θ_, prec_ : 50] := N[PDF[GammaDistribution[α, θ], x], prec]

GammaCDFBuiltin[x_, α_, θ_, prec_ : 50] := N[CDF[GammaDistribution[α, θ], x], prec]

(*high-precision S function*)
S[k_, m_, v_, prec_ : 50] :=
 Piecewise[{{N[(k - m) * GammaRegularized[k / v, m / v], prec] +
     N[k * GammaPDFBuiltin[m / v, 1 + (k / v), 1, prec], prec], k > 0}}]
(*high-precision CDF*)
PreciseCDF[k_, m_, v_, prec_ : 50] := N[S[k + 1, m, v, prec] - S[k, m, v, prec], prec]
(*high-precision PMF*)
PrecisePMF[k_, m_, v_, prec_ : 50] :=
 N[PreciseCDF[k, m, v, prec] - PreciseCDF[k - 1, m, v, prec], prec]
```

```
kValue = 899;
mValue = 20;
vValue = 0.4;
(*choose precision*)
PREC = 3000;

(*.........*)
kValue = SetPrecision[kValue, PREC];
mValue = SetPrecision[mValue, PREC];
vValue = SetPrecision[vValue, PREC];

(*printing the correspondent pmf value*)
PMFvalue = PrecisePMF[kValue, mValue, vValue, PREC]
```

$$7.546400866382255885347713194536693711492162882316345284702446085186893685870041201470112724299482666955034012629588290841273292438966119731696016751349181859759638115886199288165231018040835274580616110653830264331546314028350271298841 \times 10^{-2761}$$

```
(*Check the precision of the result*)
Precision[PMFvalue]
```

```
235.848
```

Appendix D

R package code

```r
#' Cumulative Distribution Function of the GD-Poisson Distribution
#'
#' Computes the cumulative distribution function (CDF) of the GD-Poisson
#' distribution for a vector of integer values. The parameters `lambda` and
#' `theta` can also be vectors of the same length as `q`, or any parameter can
#' be a single value to be applied across all evaluations.
#'
#' @param q A vector of non-negative integers at which to evaluate the CDF.
#' @param lambda A positive parameter representing the mean of the distribution.
#' @param theta A positive parameter representing the dispersion of the distribution.
#'
#' @return A numeric vector of the same length as \code{q}, containing the CDF
#' evaluated at each element of \code{q}.
#' @details The GD-Poisson distribution is a generalization of the Poisson
#' distribution that introduces arbitrary underdispersion.
#' @examples
#' # Evaluate the CDF at q = 0:10 with lambda = 5 and theta = 4
#' pgdpois(0:10, lambda = 5, theta = 6)
#'
#' @export
pgdpois <- function(q, lambda, theta) {
  # Get the lengths of each input
  lengths <- c(length(q), length(lambda), length(theta))

  # Check if all lengths are the same or if any length is 1
```

```
  if (!all(lengths == max(lengths) | lengths == 1)) {
    stop("All inputs must either be of the same length or length 1.")
  }

  # Determine the length to use for vectorization
  max_length <- max(lengths)

  # Repeat values if necessary to match the maximum length
  q <- if (length(q) == 1) rep(q, max_length) else q
  lambda <- if (length(lambda) == 1) rep(lambda, max_length) else lambda
  theta <- if (length(theta) == 1) rep(theta, max_length) else theta

  # Apply the scalar function element-wise
  sapply(seq_along(q), function(i) {
    pgdpois_scalar(q[i], lambda[i], theta[i])
  })

}


# Internal function: Scalar CDF of the GD-Poisson distribution
pgdpois_scalar <- function(k, lambda, theta) {
  if (k < 0 ) {
    return(0)
  }
  if ( k != floor(k)){
    k <- floor(k)
  }
  if (lambda <= 0 || theta <= 0) {
    stop("Parameters 'lambda' and 'theta' must be positive.")
  }

  kp1 <- k + 1
```

```
  #cdf in zero
  if (k == 0) {
    return((kp1 - lambda) * (pgamma(lambda, shape = kp1 / theta, scale = theta,
                                    lower.tail = FALSE)) +
             kp1 * (dgamma(lambda / theta, shape = 1 + kp1 / theta, scale = 1)))
  }
  #cdf in k
  return(
    (kp1 - lambda) * (pgamma(lambda, shape = kp1 / theta, scale = theta,
                             lower.tail = FALSE)) +
      kp1 * (dgamma(lambda / theta, shape = 1 + kp1 / theta, scale = 1)) -
      ((k - lambda) * (pgamma(lambda, shape = k / theta, scale = theta,
                              lower.tail = FALSE)) +
         k * (dgamma(lambda / theta, shape = 1 + k / theta, scale = 1)))
  )


}

# R/dgdpois.R

#' Probability Mass Function of the GD-Poisson Distribution
#'
#' Computes the probability mass function (PMF) of the GD-Poisson distribution
#' for a vector of integer values. The parameters \code{lambda} and \code{theta}
#' can also be vectors of the same length as \code{k}, or any parameter can be
#' a single value to be applied across all evaluations.
#'
#' @param k A vector of non-negative integers at which to evaluate the PMF.
#' @param lambda A positive parameter representing the mean of the distribution.
#' @param theta A positive parameter representing the dispersion of the distribution.
#'
#' @return A numeric vector of the same length as \code{k}, containing the PMF
```

```r
#' evaluated at each element of \code{k}.
#' @details The GD-Poisson distribution allows for greater flexibility in
#' modeling count data with over-dispersion compared to the standard Poisson distribution.
#'
#' The function is vectorized, meaning that \code{k}, \code{lambda}, and
#' \code{theta} can be vectors. If any of these parameters are single values
#' (length 1), they will be used for all evaluations. All vector inputs must
#' either be of the same length or have a length of 1.
#'
#' @examples
#' # Evaluate the PMF at k = 0:10 with lambda = 2 and theta = 1
#' dgdpois(0:10, lambda = 2, theta = 1)
#'
#' # Vectorized parameters: different values for theta
#' dgdpois(0:5, lambda = 2, theta = c(1, 1.1, 1.2, 1.3, 1.4, 1.5))
#'
#' # Scalar lambda and theta with vector k
#' dgdpois(0:5, lambda = 2, theta = 1)
#'
#' @export
dgdpois <- function(k, lambda, theta) {
  # Get the lengths of each input
  lengths <- c(length(k), length(lambda), length(theta))

  # Check if all lengths are the same or if any length is 1
  if (!all(lengths == max(lengths) | lengths == 1)) {
    stop("All inputs must either be of the same length or length 1.")
  }

  # Determine the length to use for vectorization
  max_length <- max(lengths)

  # Repeat values if necessary to match the maximum length
```

```
  k <- if (length(k) == 1) rep(k, max_length) else k
  lambda <- if (length(lambda) == 1) rep(lambda, max_length) else lambda
  theta <- if (length(theta) == 1) rep(theta, max_length) else theta

  # Apply the scalar function element-wise
  sapply(seq_along(k), function(i) {
    dgdpois_scalar(k[i], lambda[i], theta[i])
  })
}


# Internal function: Scalar PMF of the GD-Poisson distribution
dgdpois_scalar <- function(k, lambda, theta) {
  if (k != floor(k)){
    stop("k must be an integer")
  }
  if (k < 0) {
    return(0)
  }
  #return(pgdpois_scalar(k, lambda, theta) - pgdpois_scalar(k - 1, lambda, theta))
  kp1 <- k + 1
  if (k == 0) {
    return((kp1 - lambda) * (pgamma(lambda, shape = kp1 / theta, scale = theta,
                                    lower.tail = FALSE)) +
             kp1 * (dgamma(lambda / theta, shape = 1 + kp1 / theta, scale = 1)))
  }
  if (k == 1) {
    return(     (kp1 - lambda) * (pgamma(lambda, shape = kp1 / theta, scale = theta,
                                         lower.tail = FALSE)) +
                  kp1 * (dgamma(lambda / theta, shape = 1 + kp1 / theta, scale = 1))

                -2*((k - lambda) * (pgamma(lambda, shape = k / theta, scale = theta,
                                           lower.tail = FALSE)) +
```

```
                    k * (dgamma(lambda / theta, shape = 1 + k / theta, scale = 1)) )

    )
  }

  km1 <- k - 1
  return(
    (kp1 - lambda) * (pgamma(lambda, shape = kp1 / theta, scale = theta,
                             lower.tail = FALSE)) +
      kp1 * (dgamma(lambda / theta, shape = 1 + kp1 / theta, scale = 1))

    -2*((k - lambda) * (pgamma(lambda, shape = k / theta, scale = theta,
                               lower.tail = FALSE)) +
          k * (dgamma(lambda / theta, shape = 1 + k / theta, scale = 1)))
    +
      (km1 - lambda) * (pgamma(lambda, shape = km1 / theta, scale = theta,
                               lower.tail = FALSE)) +
      km1 * (dgamma(lambda / theta, shape = 1 + km1 / theta, scale = 1))
  )




}



#' Random Generation from the GD-Poisson Distribution
#'
#' Generates random samples from the GD-Poisson distribution.
#'
#' @param n Integer specifying the number of random samples to generate.
```

```r
#' @param lambda A positive parameter representing the mean of the distribution.
#' @param theta A positive parameter representing the dispersion of the distribution.
#'
#' @return An integer vector of length \code{n}, containing random samples from
#' the GD-Poisson distribution.
#' @details The generated random sample follow the GD-Poisson distribution.
#' @examples
#' # Generate 10 random samples with lambda = 2 and theta = 1
#' rgdpois(10, lambda = 2, theta = 1)
#' @export
rgdpois <- function(n, lambda, theta) {
  # n: number of samples to generate
  # lambda, theta: parameters of the distribution
  sample <- integer(n)

  for (i in seq_len(n)) {
    u <- runif(1)  # Uniform random number between 0 and 1
    k <- -1        # Start from k = 0 after increment
    repeat {
      k <- k + 1
      if (pgdpois_scalar(k, lambda, theta) >= u) {
        sample[i] <- k
        break
      }
    }
  }
  return(sample)
}

# R/qgdpois.R

#' Quantile Function of the GD-Poisson Distribution
#'
```

```
#' Computes the quantile function of the GD-Poisson distribution for a vector of
#' probabilities. The parameters \code{lambda} and \code{theta} can also be
#' vectors of the same length as \code{p}, or any parameter can be a single
#' value to be applied across all evaluations.
#'
#' @param p A vector of probabilities (each between 0 and 1) at which to evaluate
#' the quantiles.
#' @param lambda A positive parameter representing the mean of the distribution.
#' @param theta A positive parameter representing the dispersion of the distribution.
#' @param lower.tail Logical; if \code{TRUE} (default), probabilities are
#' \eqn{P(X \leq x)}, otherwise, \eqn{P(X > x)}.
#' @param log.p Logical; if \code{TRUE}, probabilities \code{p} are given as
#' \code{log(p)}.
#'
#' @return An integer vector of the same length as \code{p}, containing the
#' quantiles corresponding to each probability in \code{p}.
#' @details The quantile function finds the smallest integer \eqn{k} such that
#' the CDF at \eqn{k} is greater than or equal to the specified probability \eqn{p}.
#'
#' The function is vectorized, meaning that \code{p}, \code{lambda}, and \code{theta}
#' can be vectors. If any of these parameters are single values (length 1),
#' they will be used for all evaluations. All vector inputs must either be of
#' the same length or have a length of 1.
#'
#' @examples
#' # Find the 25th, 50th, and 75th percentiles with lambda = 2 and theta = 1
#' qgdpois(c(0.25, 0.5, 0.75), lambda = 10, theta = 1)
#'
#' # Vectorized parameters: different values for theta
#' qgdpois(c(0.25, 0.5, 0.75), lambda = c(1, 5.3, 6.5), theta = 8)
#'
#' # Scalar values
#' qgdpois(0.99, lambda = 24, theta = 1)
```

```r
#'
#' @export
qgdpois <- function(p, lambda, theta, lower.tail = TRUE, log.p = FALSE) {
  # Handle log.p transformation
  if (log.p) {
    p <- exp(p)
  }

  # Handle lower.tail transformation
  if (!lower.tail) {
    p <- 1 - p
  }

  # Validate probabilities
  if (any(p < 0 | p > 1)) {
    stop("All probabilities 'p' must be between 0 and 1.")
  }

  # Get the lengths of each input
  lengths <- c(length(p), length(lambda), length(theta))

  # Check if all lengths are the same or if any length is 1
  if (!all(lengths == max(lengths) | lengths == 1)) {
    stop("All inputs must either be of the same length or length 1.")
  }

  # Determine the length to use for vectorization
  max_length <- max(lengths)

  # Repeat values if necessary to match the maximum length
  p <- if (length(p) == 1) rep(p, max_length) else p
  lambda <- if (length(lambda) == 1) rep(lambda, max_length) else lambda
  theta <- if (length(theta) == 1) rep(theta, max_length) else theta
```

```
  # Apply the scalar function element-wise
  sapply(seq_along(p), function(i) {
    current_p <- p[i]
    current_lambda <- lambda[i]
    current_theta <- theta[i]

    # Handle edge cases
    if (is.na(current_p)) {
      return(NA)
    }

    if (current_p == 0) {
      return(0)
    }

    if (current_p == 1) {
      return(Inf)
    }

    k <- 0
    while (pgdpois(k, current_lambda, current_theta) < current_p) {
      k <- k + 1
      if (k > 1e6) {
        stop("Unable to find quantile. Please check the input parameters.")
      }
    }
    return(k)
  })
}



#compute a single loglikelihood value, handling edge cases
```

```r
logLikgd_scalar <- function(k, lambda, theta) {
  # Handle zero or negative PMF values to avoid log(0) or log
  #of negative numbers
  if (lambda == 0) {
    0
  } else {
    p = dgdpois_scalar(k, lambda, theta)
    #handles catastrophic cancellation cases
    if (p <= 0 | is.na(p)) {
      return(-147)
    }
    log(p)
  }
}

#computes total loglikelihood for a set of parameters params, a model matrix
#X and the response y
neg_log_likelihood_loglink <- function(params, X, y) {
  # Extract beta coefficients and theta
  p <- length(params) - 1
  betas <- params[1:p]
  logtheta <- params[p + 1]
  # Compute linear predictor and lambda
  eta <- X %*% betas
  lambda <- exp(eta)

  theta <- exp(logtheta)

  # Use sapply to compute the loglikelihood for each observation
  LLi <- sapply(seq_along(y), function(i) {
    logLikgd_scalar(y[i], lambda[i], theta)
  })
  negLL <- -sum(LLi)
```

```
  return(negLL)
}


# main optimization function, using the optim function to find the argmin of
#the negative loglikelihood
MLE_find <-
  function(X,
           y,
           start_params = NULL,
           method = "L-BFGS-B",
           max_retries = 5) {
    n <- nrow(X)
    p <- ncol(X)

    # Generate initial starting values if not provided
    generate_starting_values <- function() {
      #use Poisson as initial estimates
      glm_fit <- glm(y ~ X - 1, family = poisson(link = "log"))

      start_betas <- coef(glm_fit)
      #initial estimate of theta based on the dispersion parameter of
      #quasipoisson
      pearson_residuals <- residuals(glm_fit, type = "pearson")
      start_logtheta <- log(sum(pearson_residuals ^ 2) / (n - p))
      c(start_betas, start_logtheta)
    }

    # Get initial estimates from the Poisson model
    initial_estimates <- if (is.null(start_params)) {
      generate_starting_values()
    }
```

```r
# optim function, with retry mechanism
attempt <- 0
while (attempt <= max_retries - 1) {
  # Initialize var_cov_matrix and std_errors at the beginning of each
  #attempt
  var_cov_matrix <- NULL
  std_errors <- rep(NA, p + 1)

  #randomize the initial estimates for each attempt after the 1st one
  start_params <- initial_estimates + 0.1 * attempt * rnorm(p + 1)

  #find minima of the loglikelihood function
  fit <- tryCatch({
    optim(
      par = start_params,
      fn = neg_log_likelihood_loglink,
      X = X,
      y = y,
      method = method,
      #lower = lower_bounds,
      #upper = upper_bounds,
      control = list(
        maxit = 1000,
        parscale = abs(start_params),
        pgtol = 1e-8,
        REPORT = 500
      ),
      hessian = TRUE
    )
  }, error = function(e) {
    NULL
  })
```

```
    # Check if optimization was successful
    if (!is.null(fit) &&
        fit$convergence == 0 && is.finite(fit$value)) {
      # Compute the variance-covariance matrix
      if (!is.null(fit$hessian)) {
        var_cov_matrix <- solve(fit$hessian)
        if (!is.null(var_cov_matrix)) {
          # Hessian inversion successful
          std_errors <- sqrt(diag(var_cov_matrix))
          break
        } else {
          # Singular Hessian, retry optimization
          if (attempt == max_retries - 1)
            warning("Hessian is singular. Final attempt failed.")
        }
      } else {
        # Hessian not available, retry optimization
        if (attempt == max_retries - 1)
          warning("Hessian not available. Final attempt failed.")
      }
    } else {
      # Optimization failed
      if (attempt == max_retries - 1)
        warning("Optimization failed after final attempt. Review data or
                model specification.")
    }

    attempt <- attempt + 1
  }


  # If optimization still failed after max_retries
  if (attempt > max_retries ||
```

```
        is.null(fit) ||
        fit$convergence != 0 ||
        !is.finite(fit$value) || is.null(var_cov_matrix)) {
      stop(
        "Optimization failed after multiple attempts. Consider checking the
        data or model specification."
      )
    }


    fit$std_errors = std_errors
    fit$var_cov_matrix = var_cov_matrix

    return(fit)

  }


#this function find the fit of the current model, the null model and
#all values related to the glm
mle_gdpois_loglink <-
  function(X, y, start_params = NULL, method = "L-BFGS-B", max_retries = 5) {
    n <- nrow(X)
    p <- ncol(X)

    fit= MLE_find(X, y,start_params=start_params, method = method,
                  max_retries = max_retries)

    NULLfit= MLE_find( as.matrix(rep(1,n),ncol=1), y,start_params=start_params,
                       method = method, max_retries = max_retries)

    # Extract estimated parameters
    est_params <- fit$par
```

```
  betas_est <- est_params[1:p]
  theta_est <- exp(est_params[p + 1])

  logLikelihood = -fit$value



  saturatedLL=0

  resdeviance= 2*(saturatedLL - logLikelihood)
  nullLL=-NULLfit$value



  nulldeviance=2*(saturatedLL-nullLL)

  result <- list(
    coefficients = betas_est,
    theta = theta_est,
    std_errors = fit$std_errors,
    logLik = logLikelihood,
    aic = 2 * (p+1) - 2 * logLikelihood,
    bic = log(n) * (p+1) - 2 * logLikelihood,
    null.deviance = nulldeviance,
    deviance = resdeviance,
    df.null=n-2,
    df.residual=n-p-1,
    converged = fit$convergence,
    var_cov_matrix = fit$var_cov_matrix,
    method= method,
    X = X,
    y = y,
    n = n,
    p = p+1,
    fitted.values = exp(X%*%betas_est),
```

```
      residuals = y-exp(X%*%betas_est),
      iter = fit$counts["function"]
    )

    return(result)
  }


# R/glm_gdp.R

#' Fit a GD-Poisson Generalized Linear Model using a logarithmic link function.
#'
#' Fits a generalized linear model using the GD-Poisson distribution for the
#' response variable.
#'
#' @param formula An object of class \code{\link{formula}}. The model formula
#' specifying the response and predictors.
#' @param data A data frame containing the variables specified in the formula.
#' @param method Character string specifying the optimization method to be used
#' in the function. Default is \code{"L-BFGS-B"}.
#' @param max_retries Integer specifying the maximum number of retries for the
#' optimization algorithm.
#' @return An object of class \code{glm.gdp} containing the fitted model
#' results, including coefficients, fitted values, and other relevant information.
#'
#' @details
#' The \code{glm.gdp} function fits a GD-Poisson generalized linear model to
#' the provided data. By specifying \code{method} users can control the
#' optimization process used in maximum likelihood estimation.
#'
#' @examples
#' # Fit a GD-Poisson model with default settings
#' fit <- glm.gdp(y ~ x1 + x2, data = my_data)
```

```r
#'
#' # Fit a GD-Poisson model with custom optimization method and maximum retries
#' fit <- glm.gdp(y ~ x1 + x2, data = my_data, method = "BFGS", max_retries = 10)
#'
#' @export
glm.gdp <- function(formula, data, method = "L-BFGS-B", max_retries = 5) {
  # Extract response and design matrix
  mf <- model.frame(formula, data)
  y <- model.response(mf)
  X <- model.matrix(formula, data)
  start_params = NULL
  # Fit the model using mle_gdpois_loglink with specified parameters
  fit_result <- mle_gdpois_loglink(X, y, start_params = start_params,
                                   method = method, max_retries = max_retries)

  # Store additional information
  fit_result$call <- match.call()
  fit_result$formula <- formula
  fit_result$data <- data

  # Set the class of the object
  class(fit_result) <- "glm.gdp"

  return(fit_result)
}



#' Summary Method for GD-Poisson GLM Objects
#'
#' Provides a summary of a fitted GD-Poisson generalized linear model, including coefficien
#' standard errors, z-values, p-values, residual summaries, dispersion parameter, log-likel
#' AIC, BIC, deviance measures, degrees of freedom, number of iterations, and convergence s
```

```r
#'
#' @param object An object of class \code{gdpois_glm} produced by \code{\link{glm.gdp}}.
#' @param ... Additional arguments (currently not used).
#'
#' @return An object of class \code{summary.gdp_glm} containing the summary information.
#'
#' @export
summary.glm.gdp <- function(object, ...) {

  # Create a coefficients table
  coefficients_table <- data.frame(
    Estimate     = object$coefficients,
    `Std. Error` = object$std_errors[1:length(object$coefficients)],
    `z value`    = object$coefficients / object$std_errors[1:length(object$coefficients)],
    `Pr(>|z|)`   = 2 * (1 - pnorm(abs(object$coefficients
                                      / object$std_errors[1:length(object$coefficients)]))),
    check.names  = FALSE
  )

  # Assign parameter names as row names
  rownames(coefficients_table) <- colnames(object$X)

  # Summary as a list referencing object components
  summary_output <- list(
    call            = object$call,
    terms           = object$formula,
    coefficients     = coefficients_table,
    dispersion       = object$theta,
    null.deviance    = object$null.deviance,
    residual.deviance= object$deviance,
    aic              = object$aic,
    bic              = object$bic,
    df.null          = object$df.null,
```

```
    df.residual      = object$df.residual,
    logLik           = object$logLik,
    iter             = object$iter,
    fitted.values    = object$fitted.values,
    data             = object$data,
    n                = object$n,
    p                = object$p,  # Including theta
    converged        = object$converged
  )

  # Assign class to the summary object
  class(summary_output) <- "summary.glm.gdp"

  return(summary_output)
}




#' Print Method for Summary GD-Poisson GLM Objects
#'
#' Prints a detailed summary of a fitted GD-Poisson generalized linear model to the consol
#' including call, residual summaries, coefficients table, dispersion parameter, deviance
#' information criteria, degrees of freedom, and number of iterations.
#'
#' @param x An object of class \code{summary.glm.gdp}.
#' @param ... Additional arguments (currently not used).
#'
#' @return Prints the summary to the console.
#'
#' @export
print.summary.glm.gdp <- function(x, ...) {
```

```
  # Print the Call
  cat("\nCall:\n")
  print(x$call)

  # Print Coefficients
  cat("\nCoefficients:\n")
  printCoefmat(x$coefficients, digits = 4, signif.stars = TRUE)

  # Print Dispersion Parameter
  cat("\nDispersion parameter (theta):", formatC(x$dispersion, digits = 4), "\n")

  # Print Null and Residual Deviance
  cat("\nNull deviance:", formatC(x$null.deviance, digits = 4),
      "on", x$df.null, "degrees of freedom\n")
  cat("Residual deviance:", formatC(x$residual.deviance, digits = 4),
      "on", x$df.residual, "degrees of freedom\n")

  # Print AIC
  cat("AIC:", formatC(x$aic, digits = 4), "\n")

  # Print Number of Iterations
  cat("Number of Fisher Scoring iterations:", x$iter, "\n")

  # Check for Convergence Status
  if (!is.null(x$converged)) {
    if (x$converged == 0) {
      cat("Optimization converged successfully.\n")
    } else {
      cat("Optimization did not converge.\n")
    }
  }
}
```

Appendix E

R examples

framed

```
rm(list = ls())
options(digits = 4)
# Get the current R file location
#current_file_location <- dirname(rstudioapi::getActiveDocumentContext()$path)
# Set the working directory to the current file location
#setwd(current_file_location)

#install the package
install.packages("gdpoisson_0.2.0.tar.gz", repos = NULL, type = "source")



library(gdpoisson)
#CDF
pgdpois(0:10,lambda=5, theta=3)

##  [1] 0.09541 0.18379 0.28743 0.39763 0.50629 0.60708 0.69588 0.77072 0.83138 0.87887 0.914

#PMF
dgdpois(0:10,lambda=5, theta=3)

##  [1] 0.09541 0.08838 0.10364 0.11020 0.10866 0.10079 0.08880 0.07484 0.06066 0.04749 0.036

dgdpois(1000:1010,lambda=5, theta=3)

##  [1] 0 0 0 0 0 0 0 0 0 0 0

#underdispersion
dgdpois(0:10,lambda=5.3, theta=0.2)

##  [1] 1.730e-08 2.346e-05 2.182e-03 3.786e-02 1.906e-01 3.545e-01 2.831e-01 1.080e-01 2.132
## [11] 1.482e-04

#poisson case
dgdpois(0:10,lambda=5, theta=1)
```

```
##  [1] 0.006738 0.033690 0.084224 0.140374 0.175467 0.175467 0.146223 0.104445 0.065278 0.03
```

```
dpois(0:10,lambda=5)
```

```
##  [1] 0.006738 0.033690 0.084224 0.140374 0.175467 0.175467 0.146223 0.104445 0.065278 0.03
```

```
#ideal underdispersion behavior for any mean as theta goes to 0
round(dgdpois(20:25, lambda=22.8, theta=0.0001),4)
```

```
## [1] 0.0 0.0 0.2 0.8 0.0 0.0
```

```
round(dgdpois(1000:1006, lambda=1002.3, theta=0.0001),4)
```

```
## [1] 0.0000 0.0291 0.6433 0.3261 0.0015 0.0000 0.0000
```

```
round(dgdpois(1000:1006, lambda=1002.3, theta=0.00001),4)
```

```
## [1] 0.0000 0.0000 0.6999 0.3000 0.0000 0.0000 0.0000
```

```
#quantile
?qgdpois
qgdpois(p=0.99,lambda=5,theta=6)
```

```
## [1] 21
```

```
#sample
?rgdpois
dgdpois(0:10,lambda=10, theta=0.4)
```

```
##  [1] 5.869e-10 1.240e-07 6.059e-06 1.186e-04 1.187e-03 6.984e-03 2.647e-02 6.892e-02 1.292
## [11] 1.955e-01
```

```
#underdispersed sample
(sample1=rgdpois(100,10,0.4))
```

```
##   [1]  8 13  9  8 10 10 11 11  8 13 10 11  9 15 10 10  9  7  8 10  7 11  9 12 11 10 13 12
##  [36] 14  7  9 10 10  9 11  8  9  8 10 15  5  7 13  9 12  8 12  6 12 11  7 11 13 11  9 11
##  [71] 11 11 10  6 11 10 14 10  9  7 10 10 13  7 12  7  9 10  9  7 11 11 13  9 10 10 11 11
```

```r
mean(sample1)
```

```
## [1] 9.98
```

```r
(sample3=rgdpois(100,1000,0.01))
```

```
##   [1]  998  999 1000 1005  997  999 1003  997  992 1002 1002 1000 1007  996  999 1006 1003
##  [22] 1003 1003 1000  996 1008 1000  999  995 1007 1001 1000  996  995 1002 1000  998  996
##  [43]  999 1001 1001 1001 1001 1002  995 1003  996  997  998  999 1001 1003  997  999 1004
##  [64] 1001  997  999 1004 1003  995 1000  998 1002 1001 1004 1000 1003  997  998 1003 1003
##  [85] 1005 1000 1000 1001  998 1003 1001 1004 1003 1002 1005 1002  997 1002  997 1000
```

```r
mean(sample3)
```

```
## [1] 1000
```

```r
#poisson
(sample2=rgdpois(100,10,1))
```

```
##   [1] 15  9  8 12 15  9 14 13 11  8  7 13 11 13 10 10 12  7 12  6  7  9 11  4  7  7 11 13
##  [36] 14  9 10  6  5 13  4  9  9 10 17 14  7 13 10 14 15  7  8  8  6  4  9 14 12 10 14 11
##  [71]  9  7  9 14 10  5  9  5 14 11 14 15  7  8  7  8  7  9  8  8  6  9 15  5  9 14  8  6
```

```r
mean(sample2)
```

```
## [1] 10.09
```

```r
#souradisperso
(sample3=rgdpois(100,10,8))
```

```
##   [1] 11 33 14  7 15  1 13 16 11  2 26 19 24  3 24 18 28 10 14  1 29  5  0 11 14  0  4  6
##  [36]  7  9  3 10  6 18 16  2  8 17 19 10 15 21  4  4  0  7  7 15 21 12  5 17  8  9  6  0
##  [71]  9 12  5  4  0 23  4  0 19  7 16  5  6 10 27  4  9 16  0 13  6 25  3  3 25 28  7 26
```

```r
mean(sample3)
```

```
## [1] 10.9
```

```r
#cloth dataset
library(boot)
str(cloth)

## 'data.frame': 32 obs. of  2 variables:
##  $ x: num  1.22 1.7 2.71 3.71 3.72 3.75 4.17 4.41 4.58 4.91 ...
##  $ y: num  1 4 5 14 7 9 2 8 4 7 ...

poisglm = glm(y ~ x, data=cloth, family=poisson(link = "log"))
summary(poisglm)

##
## Call:
## glm(formula = y ~ x, family = poisson(link = "log"), data = cloth)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.9718     0.2125    4.57  4.8e-06 ***
## x             0.1930     0.0306    6.30  3.0e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 103.714  on 31  degrees of freedom
## Residual deviance:  61.758  on 30  degrees of freedom
## AIC: 189.1
##
## Number of Fisher Scoring iterations: 4

n=nrow(cloth)

sum((residuals(poisglm, type="pearson")^2))/ (n - 2)

## [1] 2.122
```

```r
gdpoismodel <- glm.gdp(y ~ x, data=cloth, max_retries = 10)

summary(gdpoismodel)

##
## Call:
## glm.gdp(formula = y ~ x, data = cloth, max_retries = 10)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.97929    0.28660   3.417 0.000633 ***
## x            0.19197    0.04144   4.632 3.62e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Dispersion parameter (theta): 1.899
##
## Null deviance: 194.2 on 30 degrees of freedom
## Residual deviance: 176.9 on 29 degrees of freedom
## AIC: 182.9
## Number of Fisher Scoring iterations: 11
## Optimization converged successfully.

####################
####################


#### underdispersed simulation
n <- 200
x1sim <- rnorm(n,14,3)
x2sim <- runif(n)
beta_true <- c(0.8, 0.3, -0.5)
theta_true <- 0.05
```

```
data <- data.frame(x1 = x1sim, x2 = x2sim)
X <- model.matrix(~ x1 + x2, data)
eta <- X %*% beta_true
lambda <- exp(eta)
# Simulate y
y <- sapply(lambda, function(l) rgdpois(1, l , theta_true) )
data$y <- y
y
```

```
##   [1]  59  90  93  84  23 108  34 117 154 116 150 151 441 735  16 269  82 115 391 188  79
##  [27]  73 320 214 231 149 126  92 198  73 100  68  65  93 164  97 159  19 348  39  25 104
##  [53] 471  39  29 114 200  58 201 183 190  56 510 398  45 195 157 295 341  76 274 178  92
##  [79] 729  83 173 110  68  99  53  50  49  53  38 136  71 435 126  72  78  74 175  52 411
## [105] 111  46  48  63  59  29 145 127  69 325 350  37  82  57  25  38 153  54 124  22   8
## [131]  36  96 200  16  32 191  87 192  58  48 120  18 266 121 668  91 284  25 321 157 122
## [157] 174 309  17 136  26 108 211 328  39  95 307  66 464  86  34 174 218 178 452 153 104
## [183] 365  48  57  79  94 104 127 300 449  85 140  87  80 228 166  69  92  57
```

```
poisglm = glm(y ~ x1 + x2, data=data, family=poisson(link = "log"))
summary(poisglm)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = poisson(link = "log"), data = data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.80923    0.03638    22.2   <2e-16 ***
## x1           0.29953    0.00218   137.5   <2e-16 ***
## x2          -0.50277    0.02081   -24.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
```

```
##     Null deviance: 20658.1738  on 199  degrees of freedom
## Residual deviance:     9.1453  on 197  degrees of freedom
## AIC: 1322
##
## Number of Fisher Scoring iterations: 3

sum((residuals(poisglm, type="pearson")^2))/ (n - 3)

## [1] 0.04649

gdpoismodel <- glm.gdp(y ~ x1 + x2, data = data, max_retries = 10)
summary(gdpoismodel)

##
## Call:
## glm.gdp(formula = y ~ x1 + x2, data = data, max_retries = 10)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.8092733  0.0076787   105.4   <2e-16 ***
## x1           0.2995312  0.0004596   651.8   <2e-16 ***
## x2          -0.5028633  0.0043549  -115.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Dispersion parameter (theta): 0.04262
##
## Null deviance:  2461 on 198 degrees of freedom
## Residual deviance: 886.5 on 196 degrees of freedom
## AIC: 894.5
## Number of Fisher Scoring iterations: 52
## Optimization converged successfully.

#quasi poisson
quasimodel <- glm(y ~ x1 + x2, data = data, family = quasipoisson(link = "log"))
summary(quasimodel)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = quasipoisson(link = "log"),
##     data = data)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.80923    0.00784     103   <2e-16 ***
## x1           0.29953    0.00047     638   <2e-16 ***
## x2          -0.50277    0.00449    -112   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 0.04649)
##
##     Null deviance: 20658.1738  on 199  degrees of freedom
## Residual deviance:     9.1453  on 197  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 3

library(MASS)
#Negative Binomial
nbmodel <- glm.nb(y ~ x1 + x2, data = data)

## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace = control$trace
> :  iteration limit reached
## Warning in glm.nb(y ~ x1 + x2, data = data):  alternation limit reached

summary(nbmodel)

##
## Call:
## glm.nb(formula = y ~ x1 + x2, data = data, init.theta = 22319580.08,
##     link = log)
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.80923    0.03638    22.2   <2e-16 ***
## x1           0.29953    0.00218   137.5   <2e-16 ***
## x2          -0.50277    0.02081   -24.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(22347269) family taken to be 1)
##
##     Null deviance: 20657.9918  on 199  degrees of freedom
## Residual deviance:     9.1452  on 197  degrees of freedom
## AIC: 1324
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  22319580
##           Std. Err.:  333321514
## Warning while fitting theta: alternation limit reached
##
##  2 x log-likelihood:  -1316

#issues with dispersion parameter, cannot model severe underdispersion.

#compois
library(mpcmp)
COMPoisglm <- glm.cmp(y ~ x1 + x2, data = data)
summary(COMPoisglm)

##
## Call: glm.cmp(formula = y ~ x1 + x2, data = data)
##
```

```
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.753  -0.700   0.026   0.722   3.676
##
## Linear Model Coefficients:
##              Estimate   Std.Err Z value Pr(>|z|)
## (Intercept)  0.809206  0.007768     104   <2e-16 ***
## x1           0.299533  0.000465     644   <2e-16 ***
## x2          -0.502786  0.004438    -113   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Mean-CMP estimated to be 22.1)
##
##
##     Null deviance: 454224.1  on 199 degrees of freedom
## Residual deviance:    200.1  on 197 degrees of freedom
##
## AIC: 896.6
```

```
####################
####################
```

```
#######  overdispersed, zero-inflated real case
#example with a real overdispersed, zero-inflated dataset (affairs) in package "COUNT"
#install.packages("COUNT")
library(COUNT)
data("affairs")
#str(affairs)

poisglm0 = glm(naffairs ~ .-vryhap-vryrel-yrsmarr6, data=affairs, family=poisson(link = "log"
summary(poisglm0)

##
```

```
## Call:
## glm(formula = naffairs ~ . - vryhap - vryrel - yrsmarr6, family = poisson(link = "log"),
##     data = affairs)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.14150    0.17799   -0.79  0.42662
## kids        -0.24170    0.10716   -2.26  0.02411 *
## vryunhap     1.23914    0.15421    8.04  9.3e-16 ***
## unhap        1.42357    0.10486   13.58  < 2e-16 ***
## avgmarr      0.49329    0.11813    4.18  3.0e-05 ***
## hapavg       0.35681    0.10211    3.49  0.00047 ***
## antirel      1.36328    0.15886    8.58  < 2e-16 ***
## notrel       0.70559    0.14426    4.89  1.0e-06 ***
## slghtrel     0.84305    0.14327    5.88  4.0e-09 ***
## smerel      -0.00849    0.15003   -0.06  0.95488
## yrsmarr1    -1.34344    0.21745   -6.18  6.5e-10 ***
## yrsmarr2    -1.60154    0.18533   -8.64  < 2e-16 ***
## yrsmarr3    -0.72720    0.11411   -6.37  1.9e-10 ***
## yrsmarr4    -0.38999    0.10184   -3.83  0.00013 ***
## yrsmarr5    -0.01432    0.10352   -0.14  0.88998
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2925.5  on 600  degrees of freedom
## Residual deviance: 2303.2  on 586  degrees of freedom
## AIC: 2827
##
## Number of Fisher Scoring iterations: 7

gdpglm0 = glm.gdp(naffairs ~ .-vryhap-vryrel-yrsmarr6, data=affairs)
summary(gdpglm0)
```

```
##
## Call:
## glm.gdp(formula = naffairs ~ . - vryhap - vryrel - yrsmarr6,
##     data = affairs)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.44514    0.45001  -0.989 0.322580
## kids        -0.05897    0.28171  -0.209 0.834190
## vryunhap     1.37163    0.42468   3.230 0.001239 **
## unhap        1.52792    0.27492   5.558 2.74e-08 ***
## avgmarr      0.63023    0.29453   2.140 0.032371 *
## hapavg       0.43154    0.25477   1.694 0.090297 .
## antirel      1.40692    0.40656   3.461 0.000539 ***
## notrel       0.63966    0.36387   1.758 0.078758 .
## slghtrel     0.86720    0.36061   2.405 0.016181 *
## smerel      -0.02061    0.37114  -0.056 0.955720
## yrsmarr1    -1.30525    0.54091  -2.413 0.015820 *
## yrsmarr2    -1.28714    0.42544  -3.025 0.002483 **
## yrsmarr3    -0.51400    0.28424  -1.808 0.070553 .
## yrsmarr4    -0.30417    0.26912  -1.130 0.258374
## yrsmarr5     0.06282    0.27781   0.226 0.821119
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Dispersion parameter (theta): 11.88
##
## Null deviance:  1474 on 599 degrees of freedom
## Residual deviance:  1387 on 585 degrees of freedom
## AIC:  1419
## Number of Fisher Scoring iterations: 150
## Optimization converged successfully.
```

Bibliography

Davis, R. A. & Liu, H. (2016). Theory and inference for a class of nonlinear models with application to time series of counts. *Statistica Sinica* , 1673–1707.

Efron, B. (1986). Double exponential families and their use in generalized linear regression. *Journal of the American Statistical Association* **81**, 709–721.

Hagmark, P.-E. (2012). An exceptional generalization of the Poisson distribution. *Open Journal of Statistics* **2**, 313–318.

Huang, A. (2017). Mean-parametrized conway–maxwell–poisson regression models for dispersed counts. *Statistical Modelling* **17**, 359–380.

Huang, A. (2023). On arbitrarily underdispersed discrete distributions. *American Statistician* **77**, 29 – 34.

McCullagh, P. (2002). What is a statistical model? *The Annals of Statistics* **30**, 1225–1310.

Puig, P., Valero, J. & Fernández-Fontelo, A. (2024). Some mechanisms leading to underdispersion: Old and new proposals. *Scandinavian Journal of Statistics* **51**, 245 – 267.

Ridout, M. S. & Besbeas, P. (2004). An empirical model for underdispersed count data. *Statistical Modelling* **4**, 77–89.

Sellers, K. F. & Morris, D. S. (2017). Underdispersion models: Models that are "under the radar". *Communications in Statistics - Theory and Methods* **46**, 12075 – 12086.

Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S. & Boatwright, P. (2005). A useful distribution for fitting discrete data: revival of the conway–maxwell–poisson distribution. *Journal of the Royal Statistical Society Series C: Applied Statistics* **54**, 127–142.

Subhabrata Chakraborti, F. J. & Epprecht, E. (2019). Higher-order moments using the survival function: The alternative expectation formula. *The American Statistician* **73**, 191–194.

Wolfram Research, I. (2024). Mathematica, Version 14.1. Champaign, IL, 2024. Available at: http://functions.wolfram.com/06.08.21.0001.01.