



**HAL**  
open science

## pyPLNmodels: A Python package to analyze multivariate high-dimensional count data

Bastien Batardiere, Joon Kwon, Julien Chiquet

### ► To cite this version:

Bastien Batardiere, Joon Kwon, Julien Chiquet. pyPLNmodels: A Python package to analyze multivariate high-dimensional count data. 2024. hal-04738542

**HAL Id: hal-04738542**

**<https://hal.inrae.fr/hal-04738542v1>**

Preprint submitted on 15 Oct 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License




# 1 pyPLNmodels: A Python package to analyze 2 multivariate high-dimensional count data


3 Bastien Batardiere <sup>1</sup>✉, Joon Kwon<sup>1</sup>, and Julien Chiquet<sup>1</sup>

4 <sup>1</sup> Université Paris-Saclay, AgroParisTech, INRAE, UMR MIA Paris-Saclay ✉ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: Johanna Bayer 

## Reviewers:

- [@LingfengLuo0510](#)
- [@mrazomej](#)

Submitted: 20 June 2024

Published: unpublished

## License

Authors of papers retain copyright  
and release the work under a  
Creative Commons Attribution 4.0  
International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## 5 Summary

6 High dimensional count data are complex to analyze as is, and normalization must be performed,  
7 but standard normalization does not fit the characteristics of count data. The Poisson  
8 LogNormal(PLN) (Aitchison & Ho, 1989) and its Principal Component Analysis variant  
9 PLN-PCA (Chiquet et al., 2018) are two-sided latent variable models allowing both suitable  
10 normalization and analysis of multivariate count data, implemented in this package.

11 Consider  $\mathbf{Y}$  a count matrix consisting of  $n$  rows and  $p$  columns. It is assumed that each  
12 individual  $\mathbf{Y}_i$ , that is the  $i^{\text{th}}$  row of  $\mathbf{Y}$ , is independent of the others and follows a Poisson  
13 lognormal distribution:

$$\mathbf{Y}_i \sim \mathcal{P}(\exp(\mathbf{Z}_i)), \quad \mathbf{Z}_i \sim \mathcal{N}(\mathbf{o}_i + \mathbf{B}^\top \mathbf{x}_i, \Sigma),$$

14 where  $\mathbf{x}_i \in \mathbb{R}^d$  and  $\mathbf{o}_i \in \mathbb{R}^p$  are user-specified covariates and offsets (with default values if not  
15 available). The  $\mathcal{P}$  (resp.  $\mathcal{N}$ ) denotes a Poisson (resp. Normal) distribution. The matrix  $\mathbf{B}$  is  
16 a  $d \times p$  matrix of regression coefficients and  $\Sigma$  is a  $p \times p$  covariance matrix. The variables  $\mathbf{Z}_i$ ,  
17 known as *latent variables*, are not directly observable. However, from a statistical perspective,  
18 they provide more informative insights compared to the observed variables  $\mathbf{Y}_i$ . The unknown  
19 parameters  $\mathbf{B}$  and  $\Sigma$  facilitates the analysis of dependencies between variables and the impact  
20 of covariates. The primary objective of the package is to estimate these parameters and  
21 retrieve the latent variables  $\mathbf{Z}_i$ . Extracting those latent variables may serve as a normalization  
22 procedure adequate to count data.

23 The only difference between the PLN and PLN-PCA models is that the latter assumes a  
24 low-rank structure on the covariance matrix, which is helpful for dimension reduction. Other  
25 variants of the PLN model exist, which are detailed in the work of Chiquet et al. (2021b).

## 26 Fields of applications and functionalities

27 Possible fields of applications include

- Ecology: Joint analysis of species abundances is a common task in ecology, whose goal is to understand the interaction between species to characterize a community, given a matrix of abundances in different sites with abundances given by

$$Y_{ij} = \text{number of species } j \text{ observed in site } i.$$

28 Additionally, the PLN models seek to explain the impact of covariates (when available),  
29 such as temperature, altitude, and other relevant factors on the observed abundances.

- Genomics: High throughput sequencing technologies now allow quantification, at the level of individual cells, various measures from the genome of humans, animals, and plants. Single-cell Ribonucleic Acid sequencing (scRNA-seq) is one of those and measures

the expression of genes at the level of individual cells. For cell  $i$  and gene  $j$ , the counts  $Y_{ij}$  is given by

$$Y_{ij} = \text{number of times gene } j \text{ is expressed in cell } i.$$

30 One of the challenges with scRNA-seq data is managing the high dimensionality, necessi-  
31 tating dimension reduction techniques suitable to count data.

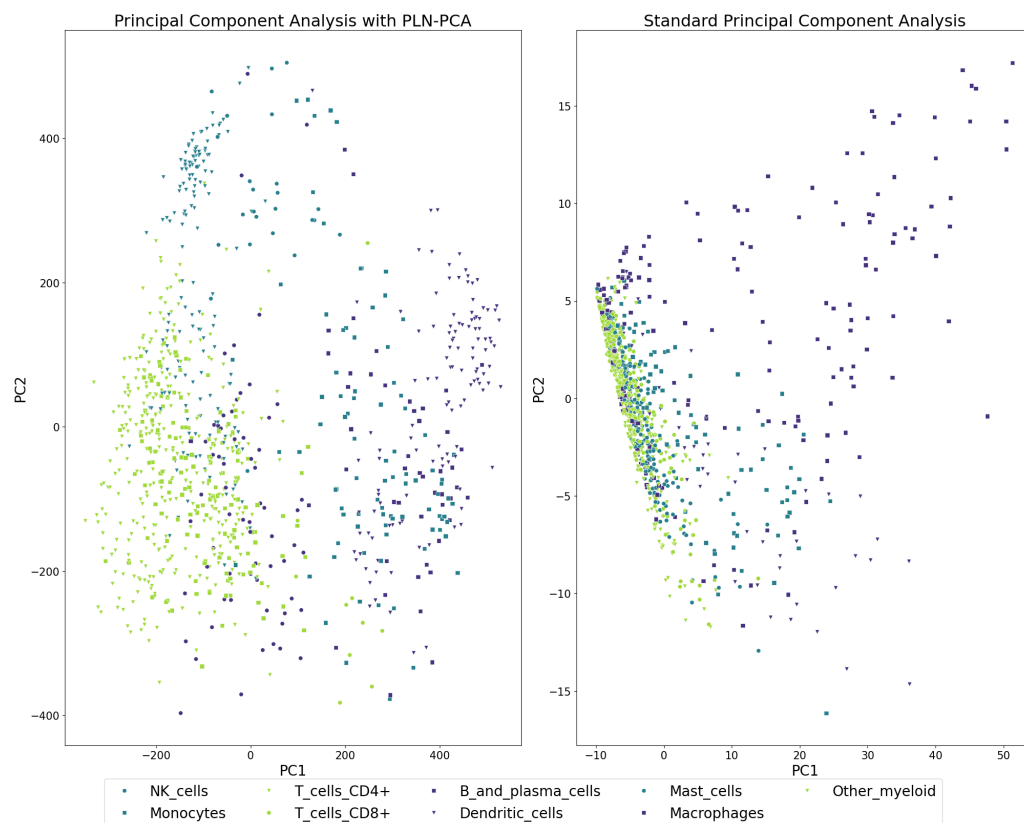
32 The PLN and PLN-PCA variants are implemented in the pyPLNmodels package introduced  
33 here, whose main functionalities are

- 34 ■ Normalize count data to obtain more valuable data,
- 35 ■ Analyze the significance of each variable and their correlation,
- 36 ■ Perform regression when covariates are available,
- 37 ■ Reduce the number of features with PLN-PCA.

38 The pyPLNmodels<sup>1</sup> package has been designed to efficiently process extensive datasets in a  
39 reasonable time and incorporates GPU acceleration for better scalability.

40 To illustrate the primary model's interest, we display below a visualization of the first two  
41 principal components when Principal Component Analysis (PCA) is performed with the PLN-  
42 PCA model (left, ours) and standard PCA on the log normalized data (right). The data  
43 considered is the scMARK benchmark (Diaz-Mejia, 2021) described in the benchmark section.  
44 We kept 1000 samples for illustration purposes. The computational time for fitting PLN-PCA  
45 is 23 seconds (on GPU), whereas standard PCA requires 0.7 second.

<sup>1</sup><https://github.com/PLN-team/pyPLNmodels>



**Figure 1:** PLN-PCA (left, ours) and standard PCA on log normalized data (right). Each cell is identified by its respective cell type. This categorization is done solely to demonstrate the method's ability to differentiate between various cell types. Unlike the standard Principal Component Analysis (PCA), which fails to distinguish between different cell types, the PLN-PCA method is capable of doing so.

## 46 Statement of need

47 While the R-package `PLNmodels` (Chiquet et al., 2021a) implements PLN models including  
 48 some variants (Chiquet et al., 2021b), the Python package `pyPLNmodels` based on Pytorch  
 49 (Paszke et al., 2019) has been built to handle large datasets of count data, such as scRNA-seq  
 50 data. Real-world scRNA-seq datasets typically involve thousands of cells ( $n \approx 20000$ ) with  
 51 thousands of genes ( $\approx 20000$ ), resulting in a matrix of size  $\approx 20000 \times 20000$ .

52 The `statsmodels` (Seabold & Perktold, 2010) is a Python library providing classes and functions  
 53 for the estimation of many different statistical models, as well as for conducting statistical tests  
 54 and statistical data exploration. Notably, It handles count data through the Generalized Linear  
 55 Models `PoissonBayesMixedGLM` and `BinomialBayesMixedGLM` classes. We stand out from this  
 56 package by allowing covariance between features and performing Principal Component Analysis  
 57 suitable to count data.

58 The R package `GLLVM` package is designed for fitting Generalized Linear Latent Variable Models.  
 59 It allows for flexible modeling of multivariate response data, accommodating both continuous  
 60 and discrete responses. Compared to the `pyPLNmodels` package, it offers a broader scope of  
 61 modeling capabilities, enabling the incorporation of Poisson distribution as well as Binomial  
 62 or Negative Binomial distributions and an additional zero-inflation component. However, its  
 63 scalability is notably inferior to our proposed methodology. Our approach, specifically the  
 64 PLN-PCA model, demonstrates superior scalability, effectively accommodating datasets with  
 65 tens of thousands of variables and the PLN model handles couple thousands of variables within

66 a reasonable computational timeframe. In contrast, GLLVM struggles to scale beyond a few  
67 hundred variables within practical computational limits.

## 68 **Benchmark**

69 We conducted a comparison using the following configurations:

- 70 ■ PLN and PLN-PCA models fitted with pyPLNmodels on CPU, referred to as **py-PLN-CPU**  
71 and **py-PLN-PCA-CPU** respectively.
- 72 ■ PLN and PLN-PCA models fitted with pyPLNmodels on GPU, referred to as **py-PLN-GPU**  
73 and **py-PLN-PCA-GPU** respectively.
- 74 ■ PLN and PLN-PCA models fitted with PLNmodels on CPU, referred to as **R-PLN** and  
75 **R-PLN-PCA** respectively.
- 76 ■ The GLLVM model with Poisson distributed responses, fitted on CPU, referred to as  
77 **GLLVM**.

78 These models were tested on the scMARK dataset, a benchmark for scRNA data, which contains  
79 19998 cell samples and 14059 gene variables. We plotted the fitting time for these models  
80 against an increasing number of gene variables, ranging from 5 to 14059. Additionally, we varied  
81 the number of cell samples at  $n = 100, 1000, 19998$ . We used  $q = 5$  Principal Components  
82 when fitting each PLN-PCA model and the number of latent variables  $LV=2$  for the GLLVM  
83 model. For each model, the fitting process was halted if the running time exceeded 10,000  
84 seconds. The computational resources utilized for this study include a machine equipped with a  
85 CPU boasting 64 GB of RAM and 32 cores, in addition to a GPU (RTX A5000) furnished with  
86 24 GB of RAM. We were unable to run GLLVM for  $n = 19998$  due to CPU memory limitations.  
87 Similarly, py-PLN-PCA-GPU could not be run when  $n = 19998$  and  $p \geq 13000$  as it exceeded  
88 the GPU memory capacity.

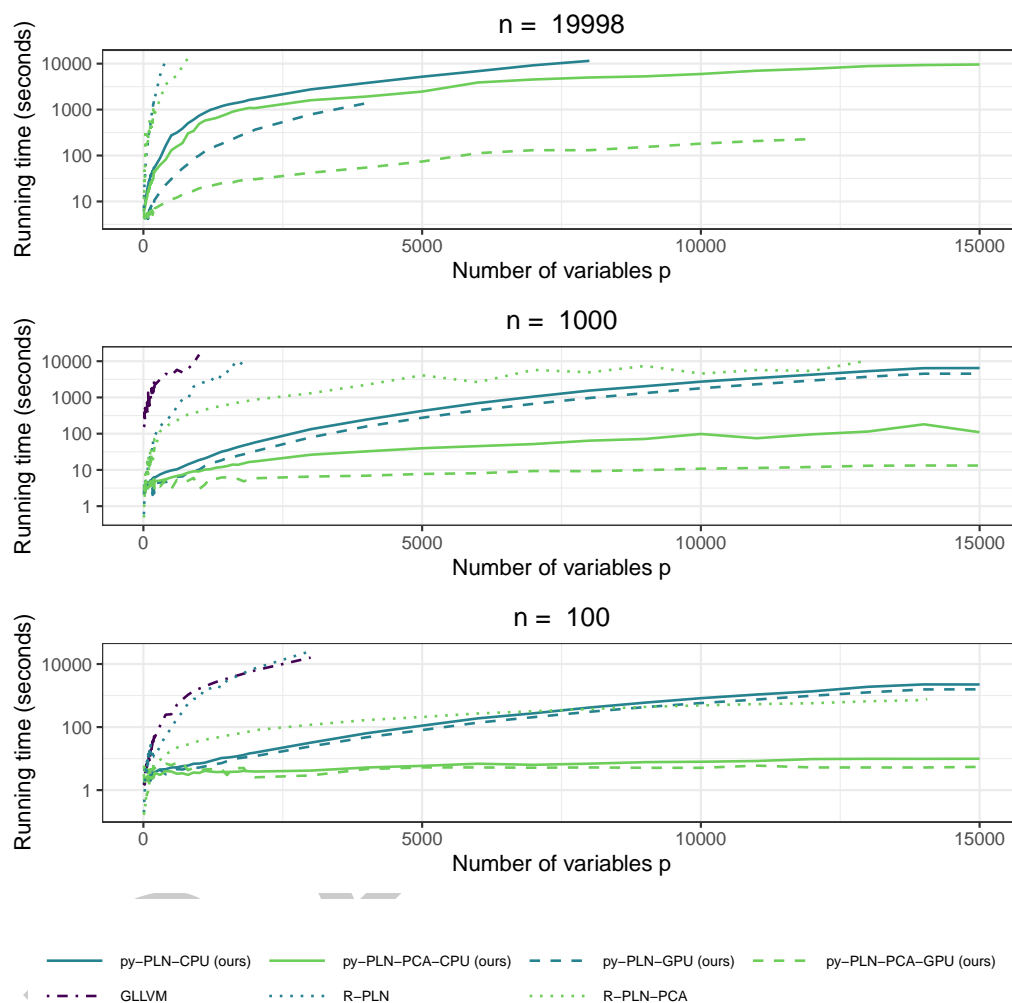


Figure 2: Running time analysis on the scMARK benchmark.

89 Each package uses variational inference (Blei et al., 2017) to maximize an Evidence Lower  
 90 Bound(ELBO), which serves as an approximation to the model's log-likelihood. Variational  
 91 inference aims to approximate the posterior distribution of the latent variables by minimizing the  
 92 divergence between the posterior and a variational distribution. To maximize the ELBO, all the  
 93 methods uses gradient ascent. The GLLVM uses the automatic differentiation of Template Model  
 94 Builder (TMB) library (Kristensen et al., 2016) with a C++ backend. PLNmodels uses C++  
 95 backend along with nlopt(Johnson, 2007) optimization library, while pyPLNmodels leverages  
 96 the automatic differentiation from Pytorch to compute the gradients of the ELBO. Each  
 97 PLN-PCA model is estimated using comparable variational inference methods. However, the  
 98 variational approximation for the PLN model in the pyPLNmodels version is more efficient than  
 99 its counterpart in PLNmodels.

## 100 Ongoing work

101 A zero-inflated version of the PLN model is currently under development, with a preprint  
 102 (Batardière et al., 2024) expected to be published shortly.

103 **Acknowledgements**

104 The authors would like to thank Jean-Benoist Léger for the time spent giving precious advice  
105 on how to build a proper Python package.

106 **Fundings**

107 Bastien Bartardière and Julien Chiquet are supported by the French ANR grant ANR-18-CE45-  
108 0023 Statistics and Machine Learning for Single Cell Genomics (SingleStatOmics).

109 **References**

- 110 Aitchison, J., & Ho, C. H. (1989). The multivariate Poisson-log normal distribution. *Biometrika*.  
111 <https://doi.org/10.1093/biomet/76.4.643>
- 112 Batardière, B., Chiquet, J., Gindraud, F., & Mariadassou, M. (2024). *Zero-inflation in the*  
113 *multivariate poisson lognormal family*. <https://arxiv.org/abs/2405.14711>
- 114 Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for  
115 statisticians. *Journal of the American Statistical Association*. [http://dx.doi.org/10.1080/](http://dx.doi.org/10.1080/01621459.2017.1285773)  
116 [01621459.2017.1285773](http://dx.doi.org/10.1080/01621459.2017.1285773)
- 117 Chiquet, J., Mariadassou, M., & Robin, S. (2018). Variational inference for probabilistic  
118 poisson PCA. In *The Annals of Applied Statistics*. <https://doi.org/10.1214/18-aos1177>
- 119 Chiquet, J., Mariadassou, M., & Robin, S. (2021a). *PLNmodels: Poisson lognormal models*.  
120 <https://cran.r-project.org/web/packages/PLNmodels/index.html>
- 121 Chiquet, J., Mariadassou, M., & Robin, S. (2021b). The poisson-lognormal model as a versatile  
122 framework for the joint analysis of species abundances. *Frontiers in Ecology and Evolution*.  
123 <https://doi.org/10.3389/fevo.2021.588292>
- 124 Diaz-Mejia, J. (2021). *scMARK an 'MNIST' like benchmark to evaluate and optimize models*  
125 *for unifying scRNA data* (Version 1.0) [Data set]. <https://doi.org/10.5281/zenodo.5765804>
- 126 Johnson, S. G. (2007). *The NLOpt nonlinear-optimization package*. [https://github.com/](https://github.com/stevengj/nlopt)  
127 [stevengj/nlopt](https://github.com/stevengj/nlopt).
- 128 Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., & Bell, B. M. (2016). TMB: Automatic  
129 differentiation and laplace approximation. *Journal of Statistical Software*. [https://doi.org/](https://doi.org/10.18637/jss.v070.i05)  
130 [10.18637/jss.v070.i05](https://doi.org/10.18637/jss.v070.i05)
- 131 Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T.,  
132 Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito,  
133 Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S.  
134 (2019). PyTorch: An imperative style, high-performance deep learning library. In  
135 *Advances in neural information processing systems 32*. [http://papers.neurips.cc/paper/](http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf)  
136 [9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf](http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf)
- 137 Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical mod-  
138 eling with python. *9th Python in Science Conference*. [https://doi.org/10.25080/](https://doi.org/10.25080/majora-92bf1922-011)  
139 [majora-92bf1922-011](https://doi.org/10.25080/majora-92bf1922-011)