

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


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

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

DOI: [10.21105/joss.06969](https://doi.org/10.21105/joss.06969)

## Software

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

Editor: Johanna Bayer ✉ 

## Reviewers:

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

Submitted: 20 June 2024

Published: 05 December 2024

## 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/)).

## Summary

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

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

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

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 available). The  $\mathcal{P}$  (respectively  $\mathcal{N}$ ) denotes a Poisson (respectively Normal) distribution. The matrix  $\mathbf{B}$  is a  $d \times p$  matrix of regression coefficients and  $\Sigma$  is a  $p \times p$  covariance matrix. The variables  $\mathbf{Z}_i$ , known as *latent variables*, are not directly observable. However, from a statistical perspective, they provide more informative insights compared to the observed variables  $\mathbf{Y}_i$ . The unknown parameters  $\mathbf{B}$  and  $\Sigma$  facilitates the analysis of dependencies between variables and the impact of covariates. The primary objective of the package is to estimate these parameters and retrieve the latent variables  $\mathbf{Z}_i$ . Extracting those latent variables may serve as a normalization procedure adequate to count data.

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

## Fields of applications and functionalities

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

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

Additionally, the PLN models seek to explain the impact of covariates (when available), 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

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

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

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

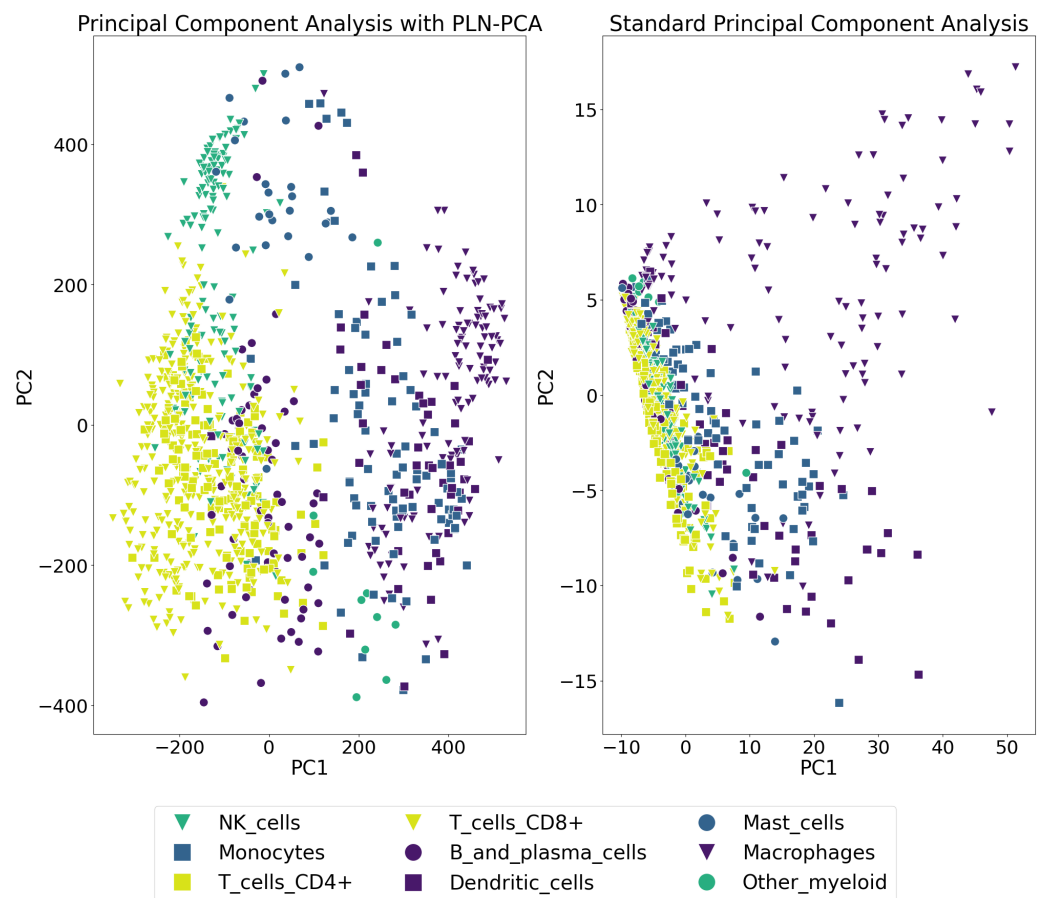
- Normalize count data to obtain more valuable data,
- Analyze the significance of each variable and their correlation,
- Perform regression when covariates are available,
- Reduce the number of features with PLN-PCA.

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

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

## Statement of need

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

The `statsmodels` (Seabold & Perktold, 2010) Python package provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests and statistical data exploration. Notably, it handles count data through the Generalized Linear Model class implementations `PoissonBayesMixedGLM` and `BinomialBayesMixedGLM`. We stand out from this package by allowing covariance between features and performing Principal Component Analysis suitable to count data.

The R package `GLLVM` package is designed for fitting Generalized Linear Latent Variable Models. It allows for flexible modeling of multivariate response data, accommodating both continuous and discrete responses. Compared to the `pyPLNmodels` package, it offers a broader scope of modeling capabilities, enabling the incorporation of Poisson distribution as well as Binomial or Negative Binomial distributions and an additional zero-inflation component. However, its scalability is notably inferior to our proposed methodology. Our approach, specifically the

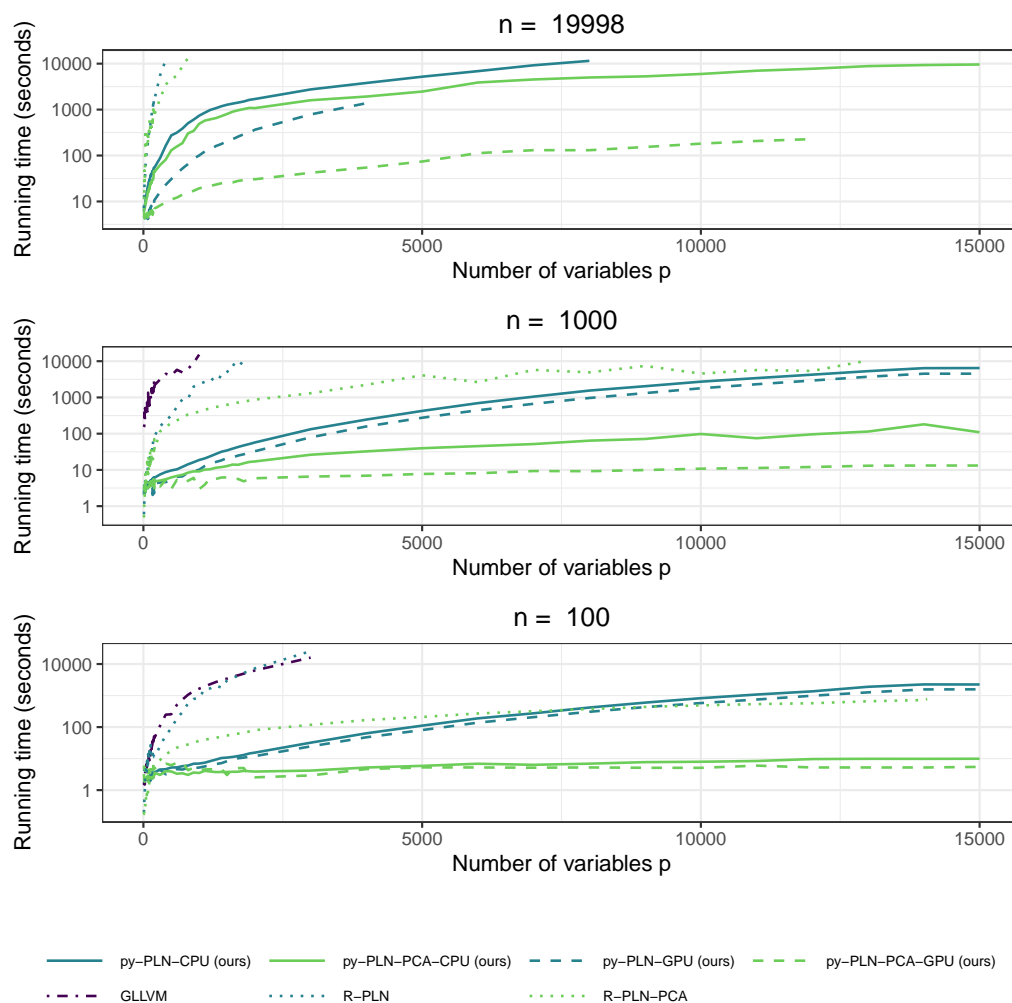
PLN-PCA model, demonstrates superior scalability, effectively accommodating datasets with tens of thousands of variables and the PLN model handles couple thousands of variables within a reasonable computational timeframe. In contrast, GLLVM struggles to scale beyond a few hundred variables within practical computational limits.

## Benchmark

We conducted a comparison using the following configurations:

- PLN and PLN-PCA models fitted with `pyPLNmodels` on CPU, referred to as **py-PLN-CPU** and **py-PLN-PCA-CPU** respectively.
- PLN and PLN-PCA models fitted with `pyPLNmodels` on GPU, referred to as **py-PLN-GPU** and **py-PLN-PCA-GPU** respectively.
- PLN and PLN-PCA models fitted with `PLNmodels` on CPU, referred to as **R-PLN** and **R-PLN-PCA** respectively.
- The GLLVM model with Poisson distributed responses, fitted on CPU, referred to as **GLLVM**.

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



**Figure 2:** Running time analysis on the scMARK benchmark.

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

## Ongoing work

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

## Acknowledgements

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

## Fundings

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

## References

- Aitchison, J., & Ho, C. H. (1989). The multivariate Poisson-log normal distribution. *Biometrika*. <https://doi.org/10.1093/biomet/76.4.643>
- Batardière, B., Chiquet, J., Gindraud, F., & Mariadassou, M. (2024). *Zero-inflation in the multivariate Poisson lognormal family*. <https://doi.org/10.21203/rs.3.rs-4637786/v1>
- Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*. <https://doi.org/10.1080/01621459.2017.1285773>
- Chiquet, J., Mariadassou, M., & Robin, S. (2018). Variational inference for probabilistic Poisson PCA. In *The Annals of Applied Statistics*. <https://doi.org/10.1214/18-aos1177>
- Chiquet, J., Mariadassou, M., & Robin, S. (2021a). *PLNmodels: Poisson lognormal models*. <https://doi.org/10.32614/cran.package.plnmodels>
- Chiquet, J., Mariadassou, M., & Robin, S. (2021b). The Poisson-lognormal model as a versatile framework for the joint analysis of species abundances. *Frontiers in Ecology and Evolution*. <https://doi.org/10.3389/fevo.2021.588292>
- Diaz-Mejia, J. (2021). *ScMARK an 'MNIST' like benchmark to evaluate and optimize models for unifying scRNA data (Version 1.0) [Data set]*. <https://doi.org/10.5281/zenodo.5765804>
- Johnson, S. G. (2007). *The NLOpt nonlinear-optimization package*. <https://github.com/stevengj/nlopt>.
- Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., & Bell, B. M. (2016). TMB: Automatic differentiation and laplace approximation. *Journal of Statistical Software*. <https://doi.org/10.18637/jss.v070.i05>
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32*. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*. <https://doi.org/10.25080/majora-92bf1922-011>