




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

Bastien Batardiere ¹, Joon Kwon¹, and Julien Chiquet¹

¹ 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: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished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](#)).

Summary

High dimensional count data are hard to analyse as is, and normalization must be performed but standard normalization does not fit to the characteristics of count data. The Poisson LogNormal(PLN) ([Aitchison & Ho, 1989](#)) and its Principal Component Analysis counterpart PLN-PCA ([Chiquet et al., 2018](#)) are two-sided models allowing both suitable normalization and analysis of multivariate count data. Each model is implemented in the pyPLNmodels package introduced here. Possible fields of applications include

- Ecology: for n sites and p species, the counts represents the number of individuals of each species in each site. The PLN models aims to understand the correlation between species, specifically to establish potential dependencies, competitive interactions, and predatory dynamics. 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: for n cells and p genes, the counts represents the number of times a gene is expressed in each cell. The objective is to estimate the correlation between genes and reduce the number of features.

The models can deal with offsets when needed. The main functionalities of the pyPLNmodels package are

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

The package has been designed to efficiently process extensive datasets in a reasonable time. It incorporates GPU acceleration and employs stochastic optimization techniques to enhance computational efficiency and speed.

To illustrate the main 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) 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 and 9 cell types for illustration purposes.

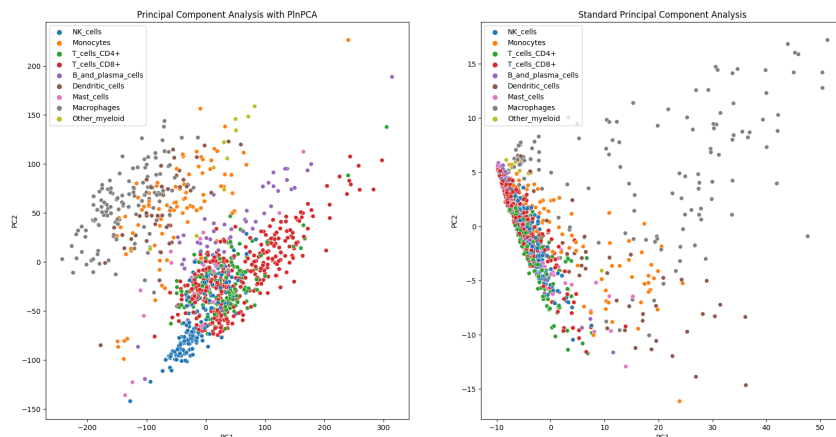


Figure 1: PLN-PCA (left) and standard PCA on log normalized data (right).

Statement of need

While the R-package `PLNmodels` (Chiquet et al., 2021) implements PLN models, the python package `pyPLNmodels` based on Pytorch (Paszke et al., 2019) has been built to handle large datasets of count data, such as scRNA (single-cell Ribonucleic acid) data. Real-world scRNA datasets typically involves thousands of cells ($n \approx 20000$) with thousands of genes (≈ 20000), resulting in a matrix of size $\approx 20000 \times 20000$. The package has GPU support for a better scalability.

The `statsmodels` (Seabold & Perktold, 2010) python package allows to deal with count data thanks to the Generalized Linear Models `PoissonBayesMixedGLM` and `BinomialBayesMixedGLM` classes. We stand out from this package by allowing covariance between features and performing Principal Component Analysis adequate to count data.

The `gllvm` package (Niku et al., 2019) offers a broader scope of modeling capabilities, enabling the incorporation of not only Poisson distribution but also Binomial or negative Binomial distributions, along with 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, while the PLN model handles 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 fit PLN and PLN-PCA models with the `pyPLNmodels` package on the scMARK dataset, a benchmark for scRNA (single-cell Ribonucleic acid) data with $n = 19998$ samples (cells) and 14059 features (gene expression). The cell type is given for each cell and can take 28 different values. We plot below the running times required to fit such models when the number of features (i.e. genes) grows. We used 60 Principal Components when fitting the PLN-PCA model. A tolerance must be set as stopping criterion when fitting each model. The running time required with the default tolerance is plot in solid line and a dotted line is plot with a relaxed tolerance. Note that the default tolerance ensures the model parameters have reached convergence but the relaxed one gives satisfying model parameters, while being much faster.

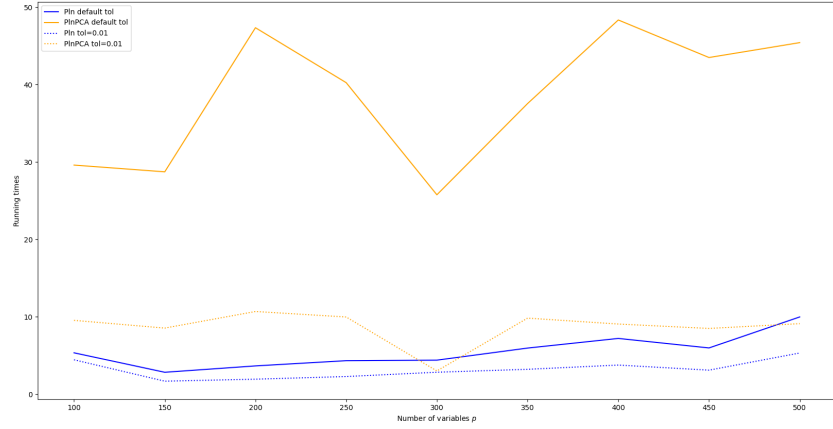


Figure 2: Running time analysis on the scMARK benchmark.

Mathematical description

Models

We introduce formally the PLN (Aitchison & Ho, 1989) and PLN-PCA (Chiquet et al., 2018) models. Let $n, p, d, q \in \mathbb{N}_*^4$. We consider:

- n samples ($i = 1, \dots, n$)
- p features ($j = 1, \dots, p$)
- n measures $X_i = (x_{ih})_{1 \leq h \leq d}$: X_{ih} = given covariate for sample i
- n counts $Y_i = (Y_{ij})_{1 \leq j \leq p}$
- n offsets $O_i = (o_{ij})_{1 \leq j \leq p}$

We assume that for all $1 \leq i \leq n$, the observed abundances $(Y_{ij})_{1 \leq j \leq p}$ are independent conditionally on a latent variable $Z_i \in \mathbb{R}^p$ such that:

$$\begin{aligned} Z_i &\sim \mathcal{N}(\beta^\top X_i, CC^\top) \\ (Y_{ij} | Z_{ij}) &\sim \mathcal{P}(\exp(o_{ij} + Z_{ij})), \end{aligned} \quad (1)$$

where $\beta \in \mathbb{R}^{d \times p}$ represents the unknown regression coefficients, and $C \in \mathbb{R}^{p \times q}$ denotes an unknown matrix, with $q \leq p$ a hyperparameter. When $q < p$, the model corresponds to PLN-PCA. Conversely, when $q = p$, the model reverts to the standard PLN. The unknown (and identifiable) parameter is $\theta = (\Sigma, \beta)$, where $\Sigma = CC^\top$ corresponds to the covariance matrix of the gaussian component.

Inference

We infer the parameter θ by maximizing the bi-concave Evidence Lower BOund(ELBO):

$$J_Y(\theta, q) = \mathbb{E}_q[\log p_\theta(Y, Z)] - \mathbb{E}_q[\log q(Z)] \leq \log p_\theta(Y),$$

where p_θ is the model likelihood and $q = (q_i)_{1 \leq i \leq n}$ is a variational parameter approximating the (unknown) law $Z | Y$.

Acknowledgements

The authors would like to thank Jean-Benoist Léger for the time spent on giving precious advices to build a proper python package. This work was supported by the French ANR SingleStatOmics.

References

- Aitchison, J., & Ho, C. H. (1989). The multivariate Poisson-log normal distribution. *Biometrika*, 76(4), 643–653. <https://doi.org/10.1093/biomet/76.4.643>
- Chiquet, J., Mariadassou, M., & Robin, S. (2018). *Variational inference for probabilistic poisson PCA*. <https://arxiv.org/abs/1703.06633>
- Chiquet, J., Mariadassou, M., & Robin, S. (2021). 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]. Zenodo. <https://doi.org/10.5281/zenodo.5765804>
- Niku, J., Hui, F. K. C., Taskinen, S., & Warton, D. I. (2019). Gllvm - fast analysis of multivariate abundance data with generalized linear latent variable models in r. *Methods in Ecology and Evolution*, 10, 2173–2182.
- 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 (pp. 8024–8035). Curran Associates, Inc. <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*.