






# Multivariate Covariance Generalized Linear Models in Python: The mcglm library

Jean Carlos Faoot Maia <sup>1\*</sup> and Wagner Hugo Bonat <sup>1\*</sup>

<sup>1</sup> Paraná Federal University, Brazil \* These authors contributed equally.

DOI: [10.21105/joss.06037](https://doi.org/10.21105/joss.06037)

## Software

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

Editor: [Chris Vernon](#) 

## Reviewers:

- [@Spaak](#)
- [@bkayfield](#)

Submitted: 11 August 2023

Published: 27 June 2024

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Abstract

The `mcglm` library, a newly introduced Python tool, facilitates statistical analyses using Multivariate Covariance Generalized Linear Models (McGLM). This contemporary family of models universalizes the traditional Generalized Linear Models (GLM), empowering them to handle multivariate and non-independent response variables. Due to its flexibility and explicit specification, McGLM supports a lot of statistical analyses across different kinds of data and distinct traits; in this article, we promote `mcglm`.

The `mcglm` library provides a comprehensive platform for data analysis using the McGLM framework. Built upon the established standards of the `statsmodels` library, it provides a comprehensive summary report for fitting assessment, including elementary parameters such as regression and dispersion coefficients, confidence intervals, hypothesis testing results, residual analysis, goodness-of-fit measurements, and correlation coefficients between outcomes. In addition, the library provides a rich set of link and variance functions and tools to define inner-response dependency matrices through the matrix linear predictor. The base code is extendable and reliable, reflecting sound object-oriented programming practices and thorough unit testing.

The library is hosted on PyPI and can be installed with some Python library manager, such as `pip`.

## Introduction

Dated at the beginning of the 19th century and controversial about the actual authorship, the least squares method established an optimization algorithm ([Stigler, 1981](#)). According to the Gauss-Markov theorem ([Hallin, 2014](#)), the resulting estimates are optimal and unbiased under linear conditions. This optimization method forms the basis of linear regression, one of the earliest statistical models ([Galton, 1886](#); [Narula & Wellington., 1982](#)). Linear regression associates a response variable to a group of covariates by employing a linear operation on regression coefficients ([Seal, 1967](#)). Three main assumptions for linear regression are linearity, independent realizations of the response variable, and a Gaussian homoscedastic error with a zero mean. While standing the test of time, linear regression has motivated numerous statistical proposals seeking to generalize its foundational assumptions.

Over time, many statistical proposals have aimed to extend the linear regression. Nelder & Wedderburn ([1972](#)) proposed the Generalized Linear Model (GLM), which relieves the Gaussian assumption accommodating exponential family models ([Müller, 2004](#)). Similarly, the Generalized Additive Model (GAM) ([Hastie & Tibshirani, 1986](#)) eases the linear assumption by using covariates smooth functions. The Generalized Estimating Equations (GEE) ([Liang & Zeger, 1986](#)) apply the quasi-likelihood estimating functions to deal with dependent data. Additional consolidated frameworks for dependent data include Copulas ([Krupskii & Joe, 2013](#);

Masarotto & Varin, 2012), and Mixed Models (Verbeke et al., 2014), among others. One prevalent aspect of the cited frameworks is that they cannot deal with multiple response variables.

The Multivariate Covariance Generalized Linear Model (McGLM) (Wagner H. Bonat & Jørgensen, 2016) extends the GLM by allowing the multivariate analysis of non-independent responses, including longitudinal and spatial data. The model is defined through second-moment assumptions, utilizing five essential components: the linear predictor via design matrix, link function, variance function, covariance link function, and the matrix linear predictor. As a comprehensive statistical model, McGLM facilitates analysis by assessing regression and dispersion coefficients, hypothesis tests, goodness-of-fit measurements, and correlation coefficients between response variables. To the best of our knowledge, the library `mcglm` is the first holistic framework to support statistical analysis in Python with the aid of McGLM.

## Statement of need

The McGLM framework is available for R users through the open-source package `mcglm` (W. H. Bonat, 2016); nevertheless, the language Python did not have a standard library until the library `mcglm`. The foremost library for statistical analysis in Python is the `statsmodels` (Seabold & Perktold, 2010). It implements classical statistical models, such as GLM, GAM, GEE, and Copulas. Many other libraries stand out for probabilistic programming in Python (Meent et al., 2018), such as: PyMC (Salvatier et al., 2016), Pyro (Bingham et al., 2018), and PyStan (Carpenter et al., 2017). Those libraries distinguish from `statsmodels` on their Bayesian paradigm of specifying models. The library `mcglm` specifies the McGLM in a frequentist fashion.

The library `mcglm` provides an easy interface for fitting McGLM on the standards of the `statsmodels` (Seabold & Perktold, 2010) library. It provides a comprehensive framework for statistical analysis supported by McGLM, with tools to lead its model specification, fitting, and appropriate report to assess estimates.

## Model Components

McGLM is specified by five components: linear predictors, link functions, variance functions, matrix linear predictors, and covariance link functions. In this section, we discuss the usual choice for each of these components.

McGLM offers the flexibility to specify typical linear predictors, including the usual formula notation popular in many statistical software. In alignment with the GLM framework, the link function encompasses usual choices like logit and probit for binary and binomial data, log for count data, and identity for continuous accurate data. The variance function is fundamental to the McGLM, as it is related to the marginal distribution of the response variable. Noteworthy among common choices is the power of the variance function, which is specialized for handling continuous data and defines the Tweedie family of distributions, as elucidated by Jørgensen (1987) and Jørgensen (1997). This family includes exceptional cases such as Gaussian ( $p = 0$ ), Gamma ( $p = 2$ ), and Inverse Gaussian ( $p = 3$ ). The variance function extended binomial is a common choice for analyzing bounded data. For fitting count data, the dispersion function presented by Jørgensen & Kokonendji (2015), called Poisson-Tweedie, is flexible enough to capture notable models, such as Hermite ( $p = 0$ ), Neyman Type A ( $p = 1$ ), Negative Binomial ( $p = 2$ ) and Poisson inverse Gaussian ( $p = 3$ ). The following table summarizes the mentioned variance functions:

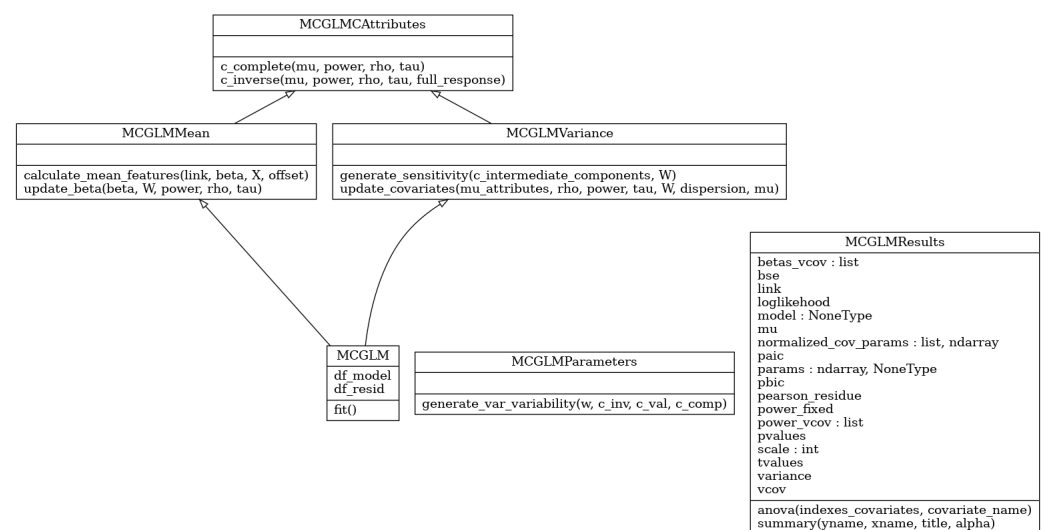
The user specifies the dependency through the Z matrices in the matrix linear predictor to describe the covariance structure. Many of the classical statistical models are replicable by setting tailored Z matrices. To cite a few, mixed models, moving averages, and compound symmetry. For more details, see Wagner H. Bonat & Jørgensen (2016) and W. Bonat (2018).

Function name	Formula
Tweedie/Power	$V(.;p) = \mu^p$
Binomial	$V(.;p) = \mu^p(1 - \mu)^p$
Poisson-Tweedie	$V(.;p) = \mu + \mu^p$

**Table 1:** Table with variance functions implemented

Finally, W. Bonat (2018) proposed three covariance link functions: identity, inverse, and exponential-matrix.

## The Python library mcglm



**Figure 1:** UML for the library

The library `mcglm` provides the first Python tool for statistical analysis with the aid of McGLM. Heavily influenced by its twin R version (W. Bonat, 2018), the library has ninety-one percent of unit-testing coverage. URLs of source-code and PyPI, the official repository for Python libraries, are <https://github.com/jeancmaia/mcglm> and <https://pypi.org/project/mcglm>. The library `mcglm` can easily be installed using the library `pip`.

The `mcglm` library is based on popular libraries of scientific Python programming: NumPy (Harris, 2020) and SciPy (Virtanen, 2020). We inherit `statsmodels`'s interface and deliver a code library akin to their standards API. Object-oriented programming is another cornerstone for the library `mcglm`; the SOLID principles (Rana & Khonica, 2021) helped to create a readable and extensible code base. The UML diagram Figure 1 presents the `mcglm` library architecture.

The implementation `mcglm` lies in six classes: `MCGLM`, `MCGLMMean`, `MCGLMVarianc`, `MCGLMAttributes`, `MCGLMParameters` and `MCGLMResults`. Each class has its scope and responsibilities. For in-depth details, access our documentation at <https://mcglm.readthedocs.io>.

We adopted the `statsmodels` standards of attribute names; the `endog` argument is a vector, or a matrix, with the realizations of the response variable; the `exog` statement defines the covariates through design matrices. For multiple outcomes, `endog` and `exog` must be specified via Python lists. The `z` argument establishes dependency arrays through numpy array structures.

Arguments `link` and `variance` set the link and variance functions, respectively. For the former, the available options are Identity, Logit, Power, Log, Probit, Cauchy, Cloglog, Log-log,

NegativeBinomial, and Inverse Power - all canonical options for GLM. Suitable options for the variance are Constant, Tweedie, BinomialP, BinomialPQ, Geometric-Tweedie, and Poisson-Tweedie. The default values for the link and variance functions are identity and constant, suitable picks for Gaussian models. For multiple outcomes, link and variance must be specified via Python lists.

The offset argument is suitable for either continuous or count outcomes. In addition, parameter `ntrial` is the canonical number of trials for binomial data. Finally, parameter `power_fixed` activates searching for the power parameter for Tweedie models. For multiple outcomes, parameters must be specified via Python lists.

An instantiated object can fit a model with the `fit()` method, which returns an object of the `MCGLMResults` class. This object can trigger two methods: `summary()`, a comprehensive report of estimates on the statsmodels fashion, and `anova()`, to an ANOVA test for categorical covariates. Some other attributes may be helpful, such as `mu`, which returns a vector with expected values; `pearson_residuals` for the Pearson normalized residuals; `aic`, `bic`, and `loglikelihood` for model comparison.

Moreover, library `mcglm` provides methods to assist in specifying the matrix linear predictor through dependence matrices `Z`. There are three available methods: `mc_id()`, which crafts a matrix for independent realizations of outcome; `mc_mixed()`, which builds matrices for mixed models, and `mc_ma()` that build matrices for moving average fitting, popular models in time series analysis. The package `mcglm` of R language implements similar methods to aid in the matrix linear predictor specification. For in-depth details about those matrices, see Wagner H. Bonat & Jørgensen (2016).

## References

- Bingham, E., Chen, J., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., & Goodman, N. (2018). *Pyro: Deep universal probabilistic programming*. <https://doi.org/10.48550/arXiv.1810.09538>
- Bonat, W. (2018). Multiple response variables regression models in R : The `mcglm` package. *Journal of Statistical Software*, 84. <https://doi.org/10.18637/jss.v084.i04>
- Bonat, W. H. (2016). Modeling Mixed outcomes in Additive Genetic Models. *ArXiv*. <https://doi.org/10.1515/ijb-2017-0001>
- Bonat, Wagner H., & Jørgensen, B. (2016). Multivariate Covariance Generalized Linear Models. *Journal of the Royal Statistical Society C*, 65(5), 649–675. <https://doi.org/10.1111/rssc.12145>
- Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan : A probabilistic programming language. *Journal of Statistical Software*, 76. <https://doi.org/10.18637/jss.v076.i01>
- Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute of Great Britain and Ireland*, 246–263. <https://doi.org/10.2307/2841583>
- Hallin, M. (2014). *Gauss-Markov theorem in Statistics*. <https://doi.org/10.1002/9781118445112.stat07536>
- Harris, H. et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hastie, T., & Tibshirani, R. (1986). Generalized Additive Models (with discussion). *Statistical Science* 1, 297–318. <https://doi.org/10.1214/ss/1177013604>
- Jørgensen, B. (1987). Exponential dispersion models. *Journal of the Royal Statistical Society*.

<https://doi.org/10.1111/j.2517-6161.1987.tb01685.x>

- Jørgensen, B. (1997). The theory of dispersion models. *CRC Press*. [https://doi.org/10.1002/1097-0258\(20000730\)19:14%3C1952::AID-SIM474%3E3.0.CO;2-K](https://doi.org/10.1002/1097-0258(20000730)19:14%3C1952::AID-SIM474%3E3.0.CO;2-K)
- Jørgensen, B., & Kokonendji, C. C. (2015). Discrete dispersion models and their Tweedie asymptotics. *AStA Advances in Statistical Analysis*, 100(1), 43–78. <https://doi.org/10.48550/arXiv.1409.7482>
- Krupskii, P., & Joe, H. (2013). Factor Copula Models for multivariate data. *Journal of Multivariate Analysis*, 120(1), 85–101. <https://doi.org/10.1016/j.jmva.2013.05.001>
- Liang, K.-Y., & Zeger, S. L. (1986). Longitudinal data analysis using Generalized Linear Models. *Biometrika*, 73(1), 13–22. <https://doi.org/10.1093/biomet/73.1.13>
- Masarotto, G., & Varin, C. (2012). Gaussian Copula marginal regression. *Electronic Journal of Statistics*, 6, 1517–1549. <https://doi.org/10.1214/12-EJS721>
- Meent, J.-W., Paige, B., Yang, H., & Wood, F. (2018). *An introduction to probabilistic programming*. <https://doi.org/10.48550/arXiv.1809.10756>
- Müller, M. (2004). *Generalized Linear Models*. [https://doi.org/10.1007/978-3-642-21551-3\\_24](https://doi.org/10.1007/978-3-642-21551-3_24)
- Narula, Subhash C., & Wellington, J. F. (1982). The minimum sum of absolute errors regression: A state of the art survey. *Internationale de Statistique*, 585(3), 317–326. <https://doi.org/10.1038/s41586-020-2649-2>
- Nelder, J. A., & Wedderburn, R. W. M. (1972). *Generalized Linear Models*. 370–384. <https://doi.org/10.2307/2344614>
- Rana, M. E., & Khonica, E. (2021). Impact of design principles and patterns on software flexibility: An experimental evaluation using flexible point (FXP). *Journal of Computer Science*, 17(7), 624–638. <https://doi.org/10.3844/jcssp.2021.624.638>
- Salvatier, J., Wiecki, T., & Fonnesbeck, C. (2016). *Probabilistic programming in Python using PyMC3*. <https://doi.org/10.7287/PEERJ.PREPRINTS.1686V1>
- Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. *Proceedings of the 9th Python in Science Conference, 2010*. <https://doi.org/10.25080/Majors-92bf1922-011>
- Seal, H. L. (1967). Studies in the history of probability and statistics. XV: The historical development of the Gauss linear model. *Biometrika*, 54(1/2), 1–24. <https://doi.org/10.2307/2333849>
- Stigler, S. M. (1981). Gauss and the invention of Least Squares. *The Annals of Statistics*, 9(3), 465–474. <https://doi.org/10.1214/aos/1176345451>
- Verbeke, G., Fieuws, S., Molenberghs, G., & Davidian, M. (2014). The analysis of Multivariate Longitudinal data: A review. *Statistical Methods in Medical Research*, 23(1), 42–59. <https://doi.org/10.1177/0962280212445834>
- Virtanen, P. et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>