




Multivariate Covariance Generalized Linear Models in Python: The mcglm library

Jean Carlos Faoot Maia ^{1*} and Wagner Hugo Bonat ^{1*}

¹ Paraná Federal University, Brazil * These authors contributed equally.

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

Software

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

Editor: [Andrew Quinn](#) 

Reviewers:

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

Submitted: 11 August 2023

Published: unpublished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](#)).

Abstract

Multivariate Covariance Generalized Linear Models (McGLM) are an extension of classic generalized linear models that allow analysis of multivariate and non-independent response variables. Due to its flexibility and explicit specification, McGLM supports many statistical analyses across a wide variety of data and distinct traits. To promote the use of McGLM for statistical analysis, we introduce the `mcglm` library, a new Python library.

The `mcglm` library provides a comprehensive platform for data analysis using the McGLM framework. It inherits the standards of the `statsmodels` library, providing a comprehensive summary report for fitting assessment that includes regression and dispersion coefficients, confidence intervals, hypothesis testing, 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 both extendable and reliable thanks to good object-oriented programming practices and unit testing.

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

Introduction

Dated at the beginning of the 19th century and controversial about 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. In the subsequent years, many other proposals have aimed to generalize these assumptions; however, the least squares method and its application in linear regression still significantly impact statistical modeling and data analysis.

The statistical modeling literature is concentrate on extending the linear model allowing more realistic assumptions. Nelder & Wedderburn ([1972](#)) proposed the Generalized Linear Model (GLM) which relieves the Gaussian assumption allowing for 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](#)) applies the quasi-likelihood estimating functions to deal with dependent data. Other consolidated frameworks to deal with 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, such as longitudinal and spatial data. This versatility is the main trait of the McGLM framework. The models are specified using only second-moment assumptions through the specification of five components: the linear predictor via design matrix, link function, variance function, covariance link function, and the matrix linear predictor. The model allows the assessment of regression and dispersion coefficients, hypothesis tests, goodness-of-fit measurements, and correlation coefficients between response variables.

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 had not had a standard library until the library `mcglm`. The foremost library statistical analysis in Python is the `statsmodels` (Seabold & Perktold, 2010), it implements classical statistical models, such as: GLM, GAM, GEE, and Copulas, however McGLM is not available. 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 from a frequentist fashion.

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

Implementation

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

McGLMs allow the specification of usual linear predictors from standard linear models including interactions terms and the usual formula notation popular in many statistical software. Similarly, the link function is specified as in the GLM framework usual choices are logit and probit for binary and binomial data, log for count data, and identity for continuous real data.

The variance function is fundamental to the McGLMs, as it is related to the marginal distribution of the response variable. To underscore some common choices, the variance function power specialized in handling continuous data and defines the Tweedie family of distributions. According to Jørgensen (1987) and Jørgensen (1997), this family has its special cases: 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 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:

To describe the covariance structure the user specifies the dependency through the Z matrices in the matrix linear predictor. 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). Finally,

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

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

The Python library mcglm

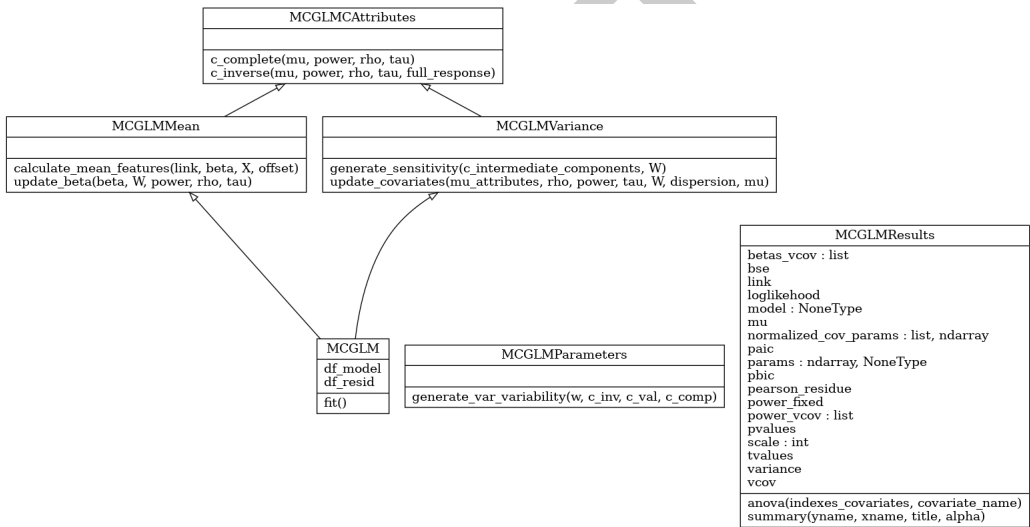


Figure 1: UML for the library

The library mcglm provides the first Python tool for statistical analysis with the aid of McGLMs. 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 with the aid of library pip.

The mcglm library is based on popular libraries of scientific Python programming: The NumPy (Harris, 2020), scipy (Virtanen, 2020), and scipy.sparse. 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 (Madasu et al., 2015) helped to create a readable and extensible code-base. The UML diagram Figure 1 presents mcglm library architecture.

The implementation mcglm lies in six classes: MCGLM, MCGLMMean, MCGLMVariance, MCGLMAttributes, MCGLMParameters and MCGLMResults. Each class has its own scope and responsibilities. For in-depth details, access the code-base <https://github.com/jeancmaia/mcglm>.

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, defined through numpy array structures.

109 Arguments link and variance set the link and variance functions, respectively. For the former,
110 the available options are Identity, Logit, Power, Log, Probit, Cauchy, Cloglog, Log-log,
111 NegativeBinomial, and Inverse Power - all canonical options for GLM. Suitable options for the
112 variance are Constant, Tweedie, BinomialP, BinomialPQ, Geometric-Tweedie, and Poisson-
113 Tweedie. The default values for link and variance functions are identity and constant, suitable
114 picks for a Gaussian models. For multiple outcomes, link and variance must be specified via
115 Python lists.

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

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

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

133 The library can be installed in any Python environment that fulfills the requirements listed on
134 PyPI Webpage.

135 Discussion

136 This article introduces the implementation of the McGLM framework in Python, providing a
137 flexible statistical modeling approach for fitting a wide range of models. The mcglm library,
138 which extends the statsmodels library, offers an accessible interface for accessing estimated
139 parameters and goodness-of-fit measures.

140 As a new alternative for statistical analysis in Python, the mcglm library can be published in
141 the statsmodels library as a new API. While the library is limited in its inability to choose the
142 covariance link function, its exclusive implementation of the identity covariance link function
143 allows the fit of numerous models listed in the statistical literature. The library's object-
144 oriented development philosophy provides a readable, extensible, self-contained, and testable
145 implementation that is ideal for developing statistical models.

146 One significant difference between the mcglm library and existing R language packs is the use
147 of object-oriented development, which proves to be suitable for statistical model production.
148 While mixed models and copulas are similar statistical tools, the mcglm library's ability to
149 analyze multiple responses sets it apart. We suggest conducting a comparative analysis of
150 statistical tools, exploring different model specifications in the future. With statsmodels
151 implementing complementary models, Python is a suitable environment for performing this
152 analysis.

153 References

154 Bingham, E., Chen, J., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh,

- 155 R., Szerlip, P., Horsfall, P., & Goodman, N. (2018). *Pyro: Deep universal probabilistic*
156 *programming*. <https://doi.org/10.48550/arXiv.1810.09538>
- 157 Bonat, W. (2018). Multiple response variables regression models in r : The mcglm package.
158 *Journal of Statistical Software*, 84. <https://doi.org/10.18637/jss.v084.i04>
- 159 Bonat, W. H. (2016). Modelling mixed outcomes in additive genetic models. *ArXiv*. <https://doi.org/10.1515/ijb-2017-0001>
160
- 161 Bonat, Wagner H., & Jørgensen, B. (2016). Multivariate covariance generalized linear models.
162 *Journal of the Royal Statistical Society C*, 65(5), 649–675. <https://doi.org/10.1111/rssc.12145>
163
- 164 Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker,
165 M., Guo, J., Li, P., & Riddell, A. (2017). Stan : A probabilistic programming language.
166 *Journal of Statistical Software*, 76. <https://doi.org/10.18637/jss.v076.i01>
- 167 Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the*
168 *Anthropological Institute of Great Britain and Ireland*, 246–263. <https://doi.org/10.2307/2841583>
169
- 170 Hallin, M. (2014). *Gauss-markov theorem in statistics*. <https://doi.org/10.1002/9781118445112.stat07536>
171
- 172 Harris, H. et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362.
173 <https://doi.org/10.1038/s41586-020-2649-2>
- 174 Hastie, T., & Tibshirani, R. (1986). Generalized additive models (with discussion). *Statistical*
175 *Science* 1, 297–318. <https://doi.org/10.1214/ss/1177013604>
- 176 Jørgensen, B. (1987). Exponential dispersion models. *Journal of the Royal Statistical Society*.
177 <https://doi.org/10.1111/j.2517-6161.1987.tb01685.x>
- 178 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)
179
- 180 Jørgensen, B., & Kokonendji, C. C. (2015). Discrete dispersion models and their tweedie
181 asymptotics. *ASTA Advances in Statistical Analysis*, 100(1), 43–78. <https://doi.org/10.48550/arXiv.1409.7482>
182
- 183 Krupskii, P., & Joe, H. (2013). Factor copula models for multivariate data. *Journal of*
184 *Multivariate Analysis*, 120(1), 85–101. <https://doi.org/10.1016/j.jmva.2013.05.001>
- 185 Liang, K.-Y., & Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models.
186 *Biometrika*, 73(1), 13–22. <https://doi.org/10.1093/biomet/73.1.13>
- 187 Madasu, V., Venna, T., & Eltaieb, T. (2015). SOLID principles in software architecture and
188 introduction to RESM concept in OOP. *Journal of Engineering Science and Technology*, 2,
189 3159–3140.
- 190 Masarotto, G., & Varin, C. (2012). Gaussian copula marginal regression. *Electronic Journal of*
191 *Statistics*, 6, 1517–1549. <https://doi.org/10.1214/12-EJS721>
- 192 Meent, J.-W., Paige, B., Yang, H., & Wood, F. (2018). *An introduction to probabilistic*
193 *programming*. <https://doi.org/10.48550/arXiv.1809.10756>
- 194 Müller, M. (2004). *Generalized linear models*. https://doi.org/10.1007/978-3-642-21551-3_24
- 195 Narula, Subhash C., & Wellington., J. F. (1982). The minimum sum of absolute errors
196 regression: A state of the art survey. *Internationale de Statistique*, 585(3), 317–326.
197 <https://doi.org/10.1038/s41586-020-2649-2>
- 198 Nelder, J. A., & Wedderburn, R. W. M. (1972). *Generalized linear models*. 370–384.
199 <https://doi.org/10.2307/2344614>

- 200 Salvatier, J., Wiecki, T., & Fonnesbeck, C. (2016). *Probabilistic programming in python using*
201 *PyMC3*. <https://doi.org/10.7287/PEERJ.PREPRINTS.1686V1>
- 202 Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with
203 python. *Proceedings of the 9th Python in Science Conference, 2010*. [https://doi.org/10.](https://doi.org/10.25080/Majora-92bf1922-011)
204 [25080/Majora-92bf1922-011](https://doi.org/10.25080/Majora-92bf1922-011)
- 205 Seal, H. L. (1967). Studies in the history of probability and statistics. XV: The historical
206 development of the gauss linear model. *Biometrika*, 54(1/2), 1–24. [https://doi.org/10.](https://doi.org/10.2307/2333849)
207 [2307/2333849](https://doi.org/10.2307/2333849)
- 208 Stigler, S. M. (1981). Gauss and the Invention of Least Squares. *The Annals of Statistics*,
209 9(3), 465–474. <https://doi.org/10.1214/aos/1176345451>
- 210 Verbeke, G., Fieuws, S., Molenberghs, G., & Davidian, M. (2014). The analysis of multivariate
211 longitudinal data: A review. *Statistical Methods in Medical Research*, 23(1), 42–59.
212 <https://doi.org/10.1177/0962280212445834>
- 213 Virtanen, P. et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in
214 Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>

DRAFT