Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Compound Poisson Normal Regression in R

Laszlo Pecze
University of Fribourg

François Collin
Ironwood Pharma

Abstract

We present an R package implementing Compound Poisson-Normal (CPN) regression. The CPN model assumes that the observed response is generated by a latent Poisson process governing the frequency of contributions, combined with normally distributed increments conditional on the Poisson count. This structure captures overdispersion and excess zeros, making the model suitable for applications in fields such as insurance claims, biology, or healthcare cost modeling. Our implementation provides a formula-based interface familiar to users of generalized linear models in R. Model fitting proceeds via maximum likelihood estimation, using the Nelder-Mead algorithm to optimize the negative log-likelihood. This package extends the applicability of compound distribution models within the R ecosystem, providing a practical tool for analyzing zero-inflated and overdispersed continuous outcomes with interpretable covariate effects.

Keywords: Compound Poisson models, Regression models, R.

1. Introduction: Compound Poisson-Normal regression in R

The Compound Poisson process is a stochastic model widely used to describe phenomena where events occur randomly over time or space, and each event contributes a random, often continuous, amount to the total outcome. This framework naturally models situations where both the frequency and severity of events are random, making it applicable to diverse fields such as insurance claims modeling Delong, Lindholm, and Wüthrich (2021), bioinformatics Hu, Wang, Feng, Xu, Liu, Heidrich-O'Hare, Chen, Yue, Zeng, Rong *et al.* (2024), and ecological studies Foster and Bravington (2013).

Different Compound Poisson processes include CPG (Compound Poisson-Gamma) for modeling count-inflated gamma-like data, CPN (Compound Poisson-Normal) for continuous outcomes with excess zeros or clusters, and CPE (Compound Poisson-Exponential) for skewed, right-tailed data with random event counts.

A common special case is the Compound Poisson-Gamma model, where the event sizes follow a

Type	Distribution	R Package	Description
GLM	Poisson	`stats` (R Core Team 2023)	Classical Poisson regression estimated by maximum likelihood
	Negative Binomial	`MASS` (Venables and Ripley 2002)	Negative Binomial regression with overdispersion parameter
ZIP	Zero-Inflated Poisson	`pscl` (Jackman 2024)	Zero-inflated Poisson model accounting for excess zeros by combining a Poisson and a separate zero-generating process
CPN	Compound Poisson-Normal	`CPN`	Compound Poisson-Normal model estimated by ML using Nelder-Mead; suitable for semi-continuous negative and/or positive data with excess zeros
CPG	Compound Poisson-Gamma	`cplm` (Zhang 2013)	Compound Poisson-Gamma model estimated by ML; accommodates overdispersed positive semi-continuous data with excess zeros

Table 1: Overview of count and semi-continuous regression models. The CPN model accommodates both zero-inflation and continuous positive and/or negative responses by combining Poisson counts with Normally distributed outcomes.

Gamma distribution. This model is particularly suited for positive, skewed, and overdispersed response data, characteristics often observed in insurance claims, biomedical cost data, and ecological counts. In the R environment, the **cplm** package Zhang (2013) provides a flexible framework for fitting Compound Poisson Generalized Linear Models (GLMs), especially the Compound Poisson-Gamma (CPG) model.

However, despite its relevance for continuous data exhibiting excess zeros, no dedicated R package has been developed for the Compound Poisson-Normal (CPN) model. Here, we introduce **CPN**, an R package that implements CPN models within a regression framework, enabling the analysis of semicontinuous or zero-inflated continuous outcomes where both event frequency and magnitude are modeled simultaneously.

2. Models and software

The Compound Poisson-Normal (CPN) regression model extends classical count data models by introducing additional flexibility for semi-continuous outcomes, combining Poisson and Normal components. It is particularly suited for modeling responses with excess zeros and positively skewed continuous values, complementing other commonly used models summarized in Table 1.

The CPN framework assumes that the response variable y_i $(i = 1, \ldots, n)$ arises from the convolution of a Poisson count process with a Normal-distributed outcome:

$$y_i = \sum_{j=1}^{N_i} Z_{ij}, \tag{1}$$

where $N_i \sim \text{Pois}(\lambda_i)$ is a Poisson-distributed latent count, and $Z_{ij} \sim \mathcal{N}(\mu, \sigma^2)$ are independent and identically distributed Normal variables. If $N_i = 0$, then $y_i = 0$ by convention.

The mean of the Poisson component depends on covariates x_i through a log-linear relationship:

$$\log(\lambda_i) \quad = \quad x_i^\top \beta, \tag{2}$$

where β are regression coefficients estimated via maximum likelihood.

The parameters estimated in the model are:

- β – regression coefficients for the Poisson intensity λ_i,
- μ – mean of the Normal component,
- σ – standard deviation of the Normal component.

Estimation is performed via maximum likelihood using the Nelder-Mead optimization method (Nelder and Mead 1965). Standard errors are obtained by numerically approximating the observed information matrix using the Hessian. If the Hessian is singular or not positive definite, a warning is issued, and standard errors are returned as `NA`.

R provides an implementation of the CPN model through the function `cpn()`, structured as follows:

```
cpn(formula, data = NULL, mu_init = NULL, sigma_init = NULL, k_max = 10)
```

The most important arguments are:

- `formula`: Model specification in formula form (e.g., `y ~ x1 + x2`),
- `data`: Optional data frame containing model variables,
- `mu_init`, `sigma_init`: Optional initial values for Normal parameters,
- `k_max`: Upper summation limit for approximating the Poisson convolution, recommended between 10 and 100.

The fitted model returns an object of class '`cpn`' containing estimated coefficients, fitted values, deviance residuals, and additional diagnostics.

3. Illustrations

For a basic illustration of the Compound Poisson-Normal (CPN) regression model, a simulated dataset is used, designed to mimic typical data arising from a latent Poisson event process with normally distributed contributions. The dataset includes both categorical and continuous predictors, along with the response variable `y`, representing the total observed outcome.

The data can be generated and loaded by:

```
R> library("CPN")
R> set.seed(123)
R> data <- simulate_cpn_data()
R> head(data)
```

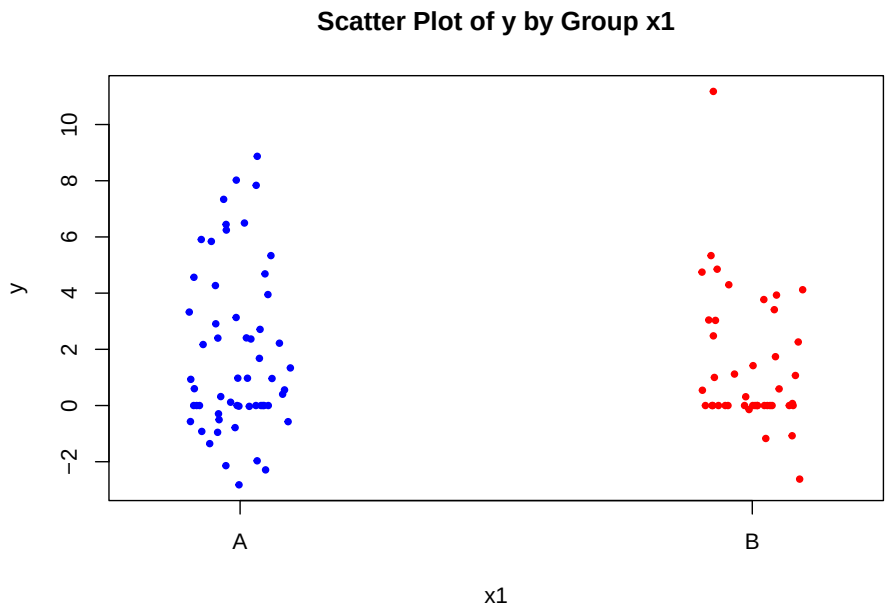

Figure 1: Scatter plot of the response `y` grouped by `x1` in the simulated CPN dataset.

```
           y x1          x2
1 -0.5703738  A  0.25331851
2  0.0000000  A -0.02854676
3  0.9613621  A -0.04287046
4  5.3350483  B  1.36860228
5  3.1328607  A -0.22577099
6 -2.6186645  B  1.51647060
```

A basic visualization of the response variable `y` grouped by the categorical predictor `x1` is shown in Figure 1.

As a first step, we fit a Compound Poisson-Normal regression model using the `cpn()()` function, modeling `y` as a function of both predictors:

```
R> fit <- cpn(y ~ x1 + x2, data = data)
```

A summary of the fitted model, including coefficient estimates, auxiliary parameters (`mu`, `sigma`), and fit statistics is obtained by:

```
R> summary(fit)

Call:
cpn(formula = y ~ x1 + x2, data = data)

Deviance Residuals:
   Min     1Q Median     3Q    Max
```

```
-2.871 -2.059 -1.269  2.181  3.745

Coefficients:
            Estimate Std.Error z.value      Pr.z
(Intercept)  0.63754   0.15190  4.1969 2.705e-05 ***
x1B         -0.60713   0.22934 -2.6473  0.008114 **
x2           0.53609   0.10791  4.9679 6.767e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Estimated mu parameter: 0.9600
Estimated sigma parameter: 1.7062

Null deviance: 478.20 on 99 degrees of freedom
Residual deviance: 449.74 on 95 degrees of freedom
AIC: 459.74
```

Estimated coefficients can also be accessed directly:

```
R> coef(fit)

(Intercept)         x1B          x2          mu       sigma
  0.6375359  -0.6071284   0.5360923   0.9599623   1.7062428
```

By default, `coef()()` returns both the linear predictor coefficients and auxiliary parameters. Setting `full = FALSE` extracts only the regression coefficients:

```
R> coef(fit, full = FALSE)

(Intercept)         x1B          x2
  0.6375359  -0.6071284   0.5360923
```

To assess model fit, standard residual diagnostics and plots are available:

```
R> plot(fit)
```

The variance-covariance matrix for parameter estimates is:

```
R> vcov(fit)

             (Intercept)          x1B           x2           mu
(Intercept)  0.023075014 -0.018395473 -0.002512385 -0.010214396
x1B         -0.018395473  0.052596614  0.000287386  0.002777711
x2          -0.002512385  0.000287386  0.011644724 -0.002747529
mu          -0.010214396  0.002777711 -0.002747529  0.027438819
sigma       -0.007321724 -0.001478573 -0.001565976  0.008114203
                   sigma
```

```
(Intercept) -0.007321724
x1B         -0.001478573
x2          -0.001565976
mu           0.008114203
sigma        0.033661888
```

For hypothesis testing, a Type I analysis of deviance evaluates the incremental contribution of each predictor:

```
R> anova(fit)
```

```
      Term     Df Deviance Resid. Df Resid. Dev   Pr(>Chi) Signif
 Residuals                       97      478.2
        x1      1   6.7993        96      471.4  0.0091193     **
        x2      1   21.657        95     449.74 3.2607e-06    ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, predictions can be generated on new data or the original dataset using the `predict()()` method:

```
R> new_df <- data.frame(
+    x1 = c("A", "B"),
+    x2 = c(0.5, -0.3)
+  )
R> predict(fit, newdata = new_df, type = "response", interval = "confidence")
```

```
        fit       lwr      upr
1 2.3743431 1.3004823 3.448204
2 0.8425844 0.4024248 1.282744
```

This concludes a basic illustration of the Compound Poisson-Normal regression model using simulated data. For further examples and advanced options, consult the full package documentation.

4. Comparison to Classical Linear Model

In this section, we compare the results from the Compound Poisson-Normal (CPN) model to those from a classical linear regression model fitted to the same data. This comparison provides insight into whether the standard linear model and the CPN model lead to similar statistical conclusions, particularly regarding the group effect of `x1`.

First, we fit a standard linear model:

```
R> lm_fit <- lm(y ~ x1 + x2, data = data)
```

Next, we extract the ANOVA tables for both the linear model and the CPN model:

We now extract the p-values for `x1` from both models. For the CPN model, the p-value is based on a likelihood ratio test (Chi-squared test), while for the linear model, the p-value is derived from an F-test:

The p-values from both models are presented below:

```
R> list("Linear model p-value for x1" = lm_pval_tex,
+       "CPN model p-value for x1" = cpn_pval_tex)

$`Linear model p-value for x1`
[1] "0.396"

$`CPN model p-value for x1`
[1] "0.009"
```

These results illustrate how the conclusions regarding the group effect `x1` may differ between the classical linear model and the more flexible CPN model. If the p-values are substantially different, it suggests that the standard linear model may not adequately account for the data's distributional characteristics, whereas the CPN model provides a potentially more appropriate framework.

Computational Details

The results presented in this paper were obtained using **R 4.4**. R itself is freely available from the Comprehensive R Archive Network (CRAN) at: https://CRAN.R-project.org.

The **Compound Poisson-Normal (CPN)** model used in this work can be installed directly from GitHub using the following R code:

```
if (!require("remotes")) install.packages("remotes")
remotes::install_github("laszlopecze77/CPN")
```

This provides access to the latest development version of the `CPN` package.

Acknowledgments

All acknowledgments (if any).

References

Delong Ł, Lindholm M, Wüthrich MV (2021). "Making Tweedie's compound Poisson model more accessible." *European Actuarial Journal*, **11**(1), 185–226.

Foster SD, Bravington MV (2013). "A Poisson–Gamma model for analysis of ecological non-negative continuous data." *Environmental and ecological statistics*, **20**(4), 533–552.

Hu H, Wang X, Feng S, Xu Z, Liu J, Heidrich-O'Hare E, Chen Y, Yue M, Zeng L, Rong Z, *et al.* (2024). "A unified model-based framework for doublet or multiplet detection in single-cell multiomics data." *Nature Communications*, **15**(1), 5562.

Jackman S (2024). *pscl: Classes and Methods for R Developed in the Political Science Computational Laboratory*. University of Sydney, Sydney, Australia. R package version 1.5.9, URL https://github.com/atahk/pscl/.

Nelder JA, Mead R (1965). "A simplex method for function minimization." *The computer journal*, **7**(4), 308–313.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Fourth edition. Springer, New York. ISBN 0-387-95457-0, URL https://www.stats.ox.ac.uk/pub/MASS4/.

Zhang Y (2013). "Likelihood-based and Bayesian methods for Tweedie compound Poisson linear mixed models." *Statistics and Computing*, **23**, 743–757.

A. More technical details

Appendices can be included after the bibliography (with a page break). Each section within the appendix should have a proper section title (rather than just *Appendix*).

For more technical style details, please check out JSS's style FAQ at https://www.jstatsoft.org/pages/view/style#frequently-asked-questions which includes the following topics:

- Title vs. sentence case.
- Graphics formatting.
- Naming conventions.
- Turning JSS manuscripts into R package vignettes.
- Trouble shooting.
- Many other potentially helpful details...

B. Using BibTeX

References need to be provided in a BibTeX file (`.bib`). All references should be made with `\cite`, `\citet`, `\citep`, `\citealp` etc. (and never hard-coded). This commands yield different formats of author-year citations and allow to include additional details (e.g., pages, chapters, ...) in brackets. In case you are not familiar with these commands see the JSS style FAQ for details.

Cleaning up BibTeX files is a somewhat tedious task – especially when acquiring the entries automatically from mixed online sources. However, it is important that informations are complete and presented in a consistent style to avoid confusions. JSS requires the following format.

- JSS-specific markup (`\proglang`, `\pkg`, `\code`) should be used in the references.
- Titles should be in title case.
- Journal titles should not be abbreviated and in title case.
- DOIs should be included where available.
- Software should be properly cited as well. For R packages `citation("pkgname")` typically provides a good starting point.

Affiliation:

Laszlo Pecze
Journal of Statistical Software
and
Department of Pharmacology
Section of Medicine
University of Fribourg
Ch. du Musée 8
1700 Fribourg, Switzerland
E-mail: laszlo.pecze@unifr.ch

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd