cplm : Compound Poisson Linear Models

Yanwei (Wayne) Zhang *
Department of Marketing
Marshall Business School
University of Southern California

Abstract

The Tweedie compound Poisson distribution is a mixture of a degenerate distribution at the origin and a continuous distribution on the positive real line. It has been applied in a wide range of fields in which continuous data with exact zeros regularly arise. Nevertheless, statistical inference based on full likelihood and Bayesian methods is not available in most statistical software, largely because the distribution has an intractable density function and numerical methods that allow fast and accurate evaluation of the density did not appear until fairly recently. The `cplm` package provides likelihood-based and Bayesian procedures for fitting common Tweedie compound Poisson linear models. In particular, models with hierarchical structures or penalized splines can be handled. Further, the package implements the Gini index based on an ordered version of the Lorenz curve as a robust model comparison tool involving zero-inflated and highly skewed distributions.

Keywords: Bayesian, Gini, splines, maximum likelihood, mixed models, Tweedie compound Poisson distribution.

If you use this software, please cite:

Zhang, Y (2013). Likelihood-based and Bayesian Methods for Tweedie Compound Poisson Linear Mixed Models, *Statistics and Computing*, 23, 743-757.

https://github.com/actuaryzhang/cplm/files/144051/TweediePaper.pdf

1. Introduction

In many fields, the data of interest have a continuous distribution on positive values but often with some observation being exact zeros. One standard method in analyzing such data is using a special form of the compound distribution, in which the response is assumed to be generated as a random sum of individual random variables with a positive support. That is, $Y = \sum_i^T X_i$, where T is the number of events following a discrete count distribution and X_i is the magnitude/severity of the outcome in the i_{th} event. When $T = 0$, we have $Y = 0$, thereby allowing the distribution to have a probability mass at the origin. Indeed, for certain applications, there is some intuitive appeal to exploit this distribution, because the underlying data can be considered as generated by a compound process. For example,

- in actuarial science, Y is the aggregate claim amount for a covered risk, T the number of reported claims and X_i the insurance payment for the i_{th} claim;
- in rainfall modeling, Y is the total amount of precipitation in a given period, T the number of rain events and X_i the intensity of the precipitation for the i_{th} rain event.
- in ecological studies of stock abundance, Y is the total biomass in a certain area, T the number of patches of organisms and X_i the measured biomass for the i_{th} patch.

Of special interest is the Tweedie compound Poisson distribution (henceforth the compound Poisson distribution) in which the distributions of T and X_i are specified as:

$$T \sim Pois(\lambda),\ X_i \overset{\text{iid}}{\sim} Gamma(\alpha, \gamma),\ T \perp X_i. \tag{1}$$

*<actuary_zhang@hotmail.com>

For a given α, this distribution can be expressed in the form of the exponential dispersion model (Jørgensen 1987) with a power variance function $V(\mu) = \mu^p$, where the power index $p = (\alpha + 2)/(\alpha + 1) \in (1, 2)$. As a result, estimation procedures developed for the exponential dispersion model are directly applicable to the compound Poisson distribution. Indeed, applications of this particular compound distribution, primarily in the form of generalized linear models [GLM], have been found in actuarial science (Smyth and Jørgensen 2002), animal studies (Perry 1981), assay analysis (Davidian 1990), botany studies (Dunn and Smyth 2005), survival analysis (Hougaard, Harvald, and Holm 1992), rainfall modeling (Dunn 2004) and fishery research (Shono 2008).

An often neglected part of the analysis is the estimation of the unknown variance function, i.e., the index parameter p. This parameter has a significant impact on hypothesis tests and predictive uncertainty measures (Davidian and Carroll 1987; Peters, Shevchenko, and Wüthrich 2009; Zhang 2013), which is of independent interest in many applications. One approach in estimating the variance function is using the profile likelihood (Cox and Reid 1987). For the compound Poisson distribution, such an approach must be implemented based on the true likelihood rather than the extended quasi-likelihood (Nelder and Pregibon 1987). The extended quasi-likelihood is not well suited to this task in that it involves a term $\log(V(y))$ which becomes infinite for $y = 0$, and thus requires adding a small positive constant to the observed zeros which, unfortunately, is highly influential on parameter estimation.

Like most other compound distributions, the density function of the compound Poisson distribution is not analytically tractable. But unlike other compound distributions whose density must be approximated via the slow recursive approach (Klugman, Panjer, and Willmot 2008), methods that enable fast and accurate numerical evaluation of compound Poisson density function are available (Dunn and Smyth 2005, 2008). These methods are provided by the `tweedie` package (Dunn 2011). With the density approximation methods, we can carry out not only maximum likelihood estimation but also Bayesian inference using Markov chain Monte Carlo methods (Gelman, Carlin, Stern, and Rubin 2003).

The `cplm` package provides both likelihood-based and Bayesian methods for various compound Poisson linear models. The following features of the package may be of special interest to the users:

1. All methods available in the package enable the index parameter (i.e., the unknown variance function) to be estimated from the data.
2. The compound Poisson generalized linear model handles large data set using the bounded memory regression facility in `biglm`.
3. For mixed models, we provide likelihood-based methods using Laplace approximation and adaptive Gauss-Hermit quadrature.
4. A convenient interface is offered to fit additive models (penalized splines) using the mixed model estimation procedure.
5. Self-tuned Markov chain Monte Carlo procedures are available for both GLM-type and mixed models.
6. We provide the Gini index based on an ordered Lorenz curve, which is better suited for model comparison involving the compound Poisson distribution.

2. Models and Examples

2.1. Overview

Table 1 shows the mapping of the user-visible functions to the underlying model structures they handle. Currently, all Bayesian methods are integrated in the `bcplm` function. A call of each function creates an object of a class that has the same name as the function. The `cplm` package is written using S4 classes and methods (Chambers 1998). Most of the user-visible classes are derived from a parent class `"cplm"`, which specifies common slots used in the package. Several methods are defined for this class, some of which allow users to manipulate attributes and slots of an S4 object in a similar way as a list. For example,

Model	Likelihood-based	Bayesian
GLM-type Models	`cpglm`	`bcplm`
Mixed Models	`cpglmm`	`bcplm`
Additive Models	`cpglmm`	`bcplm`

Table 1: Correspondence between models and functions in `cplm` .

```
R> library(cplm)
R> obj <- new("cplm")  # create an object of class "cplm"
R> names(obj)          # or slotNames(obj)

[1] "call"        "formula"     "contrasts"   "link.power"  "model.frame"
[6] "inits"

R> obj$inits           # or obj@inits

NULL

R> obj[["inits"]]      # or slot(obj, "inits")

NULL
```

To find out what methods are available for a particular class, say `"cpglm"`, use the following:

```
R> showMethods(classes = "cpglm")
```

The help documentation of the available classes and methods can be assessed in the following fashion:

```
R> class ? bcplm
R> method ? resid("cpglm") # the "resid" method for class "cpglm"
R> method ? resid("cpglmm")
```

2.2. Generalized linear models

Given the index parameter p, the Tweedie compound Poisson distribution can be written in the form of the exponential dispersion model with $E(Y) = \mu$, $Var(Y) = \phi \cdot V(\mu) = \phi \cdot \mu^p$, where $\phi > 0$ is the dispersion parameter. Using this representation, we denote the compound Poisson distribution as $Y \sim CPois(\mu, \phi, p)$.

In the following, we assume the observed response vector is $\boldsymbol{Y} = (Y_1, \cdots, Y_n)'$, where $Y_i \sim CPois(\mu_i, \phi, p)$. In the compound Poisson GLM, the mean $\boldsymbol{\mu} = E(\boldsymbol{Y})$ is stipulated as a function of some linear predictors through a link function η as

$$\eta(\boldsymbol{\mu}) = \boldsymbol{X}\boldsymbol{\beta}, \tag{2}$$

where $\boldsymbol{X}$ is the design matrix and $\boldsymbol{\beta}$ is the vector of fixed effects.

If the index parameter p is known, the compound Poisson GLM can be estimated using the Fisher's scoring algorithm (McCullagh and Nelder 1989). In R, this can be easily fitted using `glm` with `family = tweedie()`. For unknown p, parameter estimation can be proceeded using the profile likelihood approach (Cox and Reid 1987). We denote $\boldsymbol{\sigma} = (\phi, p)'$. Because for a given value of $\boldsymbol{\sigma}$, the maximum likelihood estimation $\hat{\boldsymbol{\beta}}(\boldsymbol{\sigma})$ can be determined using the scoring algorithm, we can profile $\boldsymbol{\beta}$ out of the likelihood and maximize the profile likelihood to estimate $\boldsymbol{\sigma}$ as

$$\hat{\boldsymbol{\sigma}} = \arg\max_{\boldsymbol{\sigma}} \ell(\boldsymbol{\sigma} | \boldsymbol{y}, \hat{\boldsymbol{\beta}}(\boldsymbol{\sigma})), \tag{3}$$

where $\ell(\boldsymbol{\sigma}|\boldsymbol{y}, \hat{\boldsymbol{\beta}}(\boldsymbol{\sigma}))$ is the loglikelihood evaluated at $\hat{\boldsymbol{\beta}}(\boldsymbol{\sigma})$ and $\boldsymbol{\sigma}$. This likelihood must be approximated using the compound Poisson density evaluation methods provided by Dunn and Smyth (2005, 2008) . The approximated profile loglikelihood is optimized numerically to locate the estimate $\hat{\boldsymbol{\sigma}}$, subject to the constraints $\phi > 0$ and $p \in (1, 2)$. The estimate of the regression parameters is then $\hat{\boldsymbol{\beta}}(\hat{\boldsymbol{\sigma}})$.

The profile likelihood approach is implemented by the `cpglm` function. Indeed, the function `tweedie.profile` in the `tweedie` package also makes available the profile likelihood approach. The `cpglm` function here differs from `tweedie.profile` in two aspects. First, the user does not need to specify the grid of possible values the index parameter can take. Rather, the optimization of the profile likelihood is automated. Second, big data sets can be handled where the `bigglm` function from the `biglm` package (Lumley 2011) is used in fitting GLMs. The `bigglm` function is invoked when the argument `chunksize` is greater than 0.

We use the auto insurance claim data from Yip and Yau (2005) as an example to illustrate the use of the `cpglm` function. We first load and transform the data:

```
R> da <- subset(AutoClaim, IN_YY == 1) # use data in the Yip and Yau paper
R> da <- transform(da, CLM_AMT5 = CLM_AMT5/1000,
+                  INCOME = INCOME/10000)
```

We are interested at the variable `CLM_AMT5`, which is the aggregate insurance claim amount from an auto insurance policy. A summary of this distribution shows that it has a spike at zero (61%) while the rest follows a continuous distribution on positive values:

```
R> summary(da$CLM_AMT5); sum(da$CLM_AMT5 == 0)/nrow(da)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   4.112   4.740  57.040
[1] 0.6066856
```

We would like to build a model that is predictive of the insurance loss based on some policy-related and demographic variables. For example, we include indicators of whether the car is for commercial use (`CAR_USE`), whether the policyholder is married (`MARRIED`), and whether the car is driven in the urban area (`AREA`), as well as a continuous covariate- the MVR violation points (`MVR_PTS`).

```
R> P1 <- cpglm(CLM_AMT5 ~ CAR_USE + MARRIED + AREA + MVR_PTS, data = da)
R> summary(P1)

Call:
cpglm(formula = CLM_AMT5 ~ CAR_USE + MARRIED + AREA + MVR_PTS,
    data = da)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.4951  -2.5154  -1.9716  -0.1866  12.5669
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       0.05648    0.14710   0.384    0.701
CAR_USECommercial 0.12521    0.09299   1.347    0.178
MARRIEDYes       -0.14730    0.08985  -1.639    0.101
AREAUrban         1.00957    0.14135   7.142 1.16e-12 ***
MVR_PTS           0.21683    0.01738  12.474  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Estimated dispersion parameter: 7.1348
Estimated index parameter: 1.4021

Residual deviance: 18332  on 2807  degrees of freedom
```

```
AIC:  10804

Number of Fisher Scoring iterations:  6
```

The `cpglm` function has the same syntax as the `glm` function except that there is no `family` argument. Desired link functions can be directly specify through the `link` argument, which uses the log link by default. The output shows the estimates for the index and the dispersion parameter in addition to those for the fixed effects. These coefficients can also be extracted directly from the object using

```
R> c(coef(P1), p = P1$p, phi = P1$phi)
```

Similarly, most utility extraction methods defined for a linear model object such as `fitted`, `resid`, `model.matrix` and so on can be used.

By default, the optimization of the profile likelihood in `cpglm` calls the `glm` function in estimating the fixed effects. However, it is known that the `glm` function can run into memory issues for large data sets. To overcome this problem, the `cpglm` function implements an alternative method that relies on the bounded memory regression provided by the `biglm` package (Lumley 2011). The `bigglm` function divides the data frame into chunks and regression estimates are updated chunk by chunk. This method is invoked when the argument `chunksize` is greater than 0, which specifies the size of chunks for processing the data frame.

Most arguments in the `bigglm` function can be directly passed to `cpglm` through the `...` mechanism. When using this functionality, the users are warned that

- when factors are present, the levels of each factor must be the same across all data chunks;
- `bigglm` is substantially slower than `glm` when applied on the same data.

As an example, we can refit the previous model using `bigglm` with chunks of 500 data points as follows:

```
R> P1.big <- cpglm(CLM_AMT5 ~ CAR_USE + MARRIED + AREA + MVR_PTS,
+     data = da, chunksize = 500)
```

2.3. Mixed models

In the mixed model, the mean of the response is specified as

$$\eta(\boldsymbol{\mu}) = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{b}, \tag{4}$$

$$\boldsymbol{b} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{5}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{b}$ are the vector of fixed and random effects, $\boldsymbol{X}$ and $\boldsymbol{Z}$ are the associated design matrices and $\boldsymbol{\Sigma}$ is the variance component. The goal is to make inference of the parameters $(\boldsymbol{\beta}, \phi, p, \boldsymbol{\Sigma})$. The `cplm` package implements maximum likelihood estimation in the mixed model, in which the underlying marginal likelihood can be derived as:

$$p(\boldsymbol{y}|\boldsymbol{\beta}, \phi, p, \boldsymbol{\Sigma}) = \int p(\boldsymbol{y}|\boldsymbol{\beta}, \phi, p, \boldsymbol{b}) \cdot p(\boldsymbol{b}|\boldsymbol{\Sigma}) d\boldsymbol{b}. \tag{6}$$

The above integral is often intractable, but can be approximated using the Laplace method or the adaptive Gauss-Hermite quadrature approach. The implementation of these methods is based on the old `lme4` package (version < 1.0), with changes made on the updating of the mean, the variance function and the marginal loglikelihood. For the Laplace method, the contribution of the dispersion parameter to the approximated loglikelihood is explicitly accounted for, which should be more accurate and more consistent with the quadrature estimate. Indeed, both the dispersion parameter and the index parameter are included as a part of the optimization process. In computing the marginal loglikelihood, the density of the compound Poisson distribution is approximated using numerical methods provided by the `tweedie` package (Dunn 2011). For details of the Laplace approximation and adaptive Gauss-Hermite quadrature methods for generalized linear mixed models, see the documentation associated with `lme4`.

We use the fine root data set (de Silva, Hall, Tustin, and Gandar 1999) to illustrate various features of the `cpglmm` function. We are interested at how the distribution of the fine root length density [RLD] is affected by the geometry of the structural root system and the type of the root stock.

```
R> head(FineRoot)
```

```
  Plant Stock Spacing  Zone RLD
1     1  Mark     5x3 Inner   0
2     1  Mark     5x3 Outer   0
3     1  Mark     5x3 Inner   0
4     1  Mark     5x3 Outer   0
5     1  Mark     5x3 Inner   0
6     1  Mark     5x3 Outer   0
```

Following Zhang (2013), we specify a model that includes both main effects of stock and zone and their interactions, with the intercept varying by plant:

```
R> f0 <- cpglmm(RLD ~ Stock * Zone +  (1|Plant), data = FineRoot)
R> summary(f0)
```

```
Compound Poisson linear mixed model fit by the Laplace approximation
Formula: RLD ~ Stock * Zone + (1 | Plant)
   Data: FineRoot
    AIC    BIC logLik deviance
 -172.5 -138.6  94.27   -188.5
Random effects:
 Groups   Name        Variance  Std.Dev.
 Plant    (Intercept) 0.0077274 0.087906
 Residual             0.3286284 0.573261
Number of obs: 511, groups: Plant, 8

Fixed effects:
                     Estimate Std. Error t value
(Intercept)          -2.09823    0.16528 -12.695
StockMM106           -0.06637    0.21888  -0.303
StockMark            -0.46346    0.20234  -2.290
ZoneOuter            -0.44690    0.25546  -1.749
StockMM106:ZoneOuter  0.02563    0.31241   0.082
StockMark:ZoneOuter  -1.16566    0.32468  -3.590

Estimated dispersion parameter: 0.3286
Estimated index parameter: 1.4131
```

The output is similar to that from a call of `glmer`, but reports additionally the estimates of the dispersion and the index parameters.

```
R> inherits(f0, "mer")
```

```
[1] TRUE
```

As a result, most of the methods defined in (the old) `lme4` are directly applicable:

```
R> fixef(f0)  #coefficients
```

```
         (Intercept)           StockMM106            StockMark
            -2.09823             -0.06637             -0.46346
          ZoneOuter StockMM106:ZoneOuter  StockMark:ZoneOuter
            -0.44690              0.02563             -1.16566
```

```
R> ranef(f0)

$Plant
  (Intercept)
1    0.024275
2   -0.052466
3   -0.043943
4    0.073502
5   -0.039262
6    0.040255
7   -0.007266
8    0.008004

attr(,"class")
[1] "ranef.mer"

R> VarCorr(f0)  #variance components

$Plant
            (Intercept)
(Intercept)    0.007727
attr(,"stddev")
(Intercept)
    0.08791
attr(,"correlation")
            (Intercept)
(Intercept)           1

attr(,"sc")
[1] 0.5733
```

Models with crossed and nested random effects can also be handled. We illustrate this with a slightly different model structure (we explicitly convert plant to a factor because this is required when nested effects are present):

```
R> FineRoot$Plant <- factor(FineRoot$Plant)
R> f1 <- cpglmm(RLD ~ Stock + Spacing + (1|Plant), data = FineRoot)
```

We can add another random effect from zone as crossed or nested within plant as follows:

```
R> f2 <- update(f1, . ~ . + (1|Zone))
R> f3 <- update(f1, . ~ . + (1|Plant:Zone))
R> # test the additional random effect
R> anova(f1, f2)

Data: FineRoot
Models:
f1: RLD ~ Stock + Spacing + (1 | Plant)
f2: RLD ~ Stock + Spacing + (1 | Plant) + (1 | Zone)
   Df     AIC      BIC logLik  Chisq Chi Df Pr(>Chisq)
f1  6 -107.78  -82.367 59.893
f2  7 -145.58 -115.924 79.789 39.794      1  2.823e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R> anova(f1, f3)
```

```
Data: FineRoot
Models:
f1: RLD ~ Stock + Spacing + (1 | Plant)
f3: RLD ~ Stock + Spacing + (1 | Plant) + (1 | Plant:Zone)
   Df     AIC      BIC logLik Chisq Chi Df Pr(>Chisq)
f1  6 -107.78  -82.367 59.893
f3  7 -147.04 -117.390 80.523 41.26      1  1.333e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In addition, the `cpglmm` function allows users to choose alternative optimizers that may perform better in certain circumstances. The `optimizer` argument determines which optimization routine is to be used. Possible choices are `"nlminb"` (the default), `"bobyqa"` and `"L-BFGS-B"`. We can refit the first example with a different optimizer as follows:

```
R> f4 <- cpglmm(RLD ~  Stock * Zone +  (1|Plant),
+             data = FineRoot, optimizer = "L-BFGS-B",
+             control = list(trace = 2, PQL.init = FALSE))

N = 9, M = 5 machine precision = 2.22045e-16

iterations 21
function evaluations 27
segments explored during Cauchy searches 23
BFGS updates skipped 0
active bounds at final generalized Cauchy point 0
norm of the final projected gradient 0.0424422
final function value -188.534

final  value -188.534323
converged
```

By default, the `cpglmm` function implements a penalized quasi-likelihood procedure before optimizing the approximated likelihood. In the above, we have specified `PQL.init = FALSE` to skip that step.

The default fitting method is the Laplace approximation. The quadrature method can be invoked by setting `nAGQ` to be greater than 1, which specifies the number of quadrature points to be used for the numerical evaluation. Note that this method only works when there is one single grouping factor in the model:

```
R> f5 <- cpglmm(RLD ~  Stock * Zone +  (1|Plant),
+             data = FineRoot, nAGQ = 10)
```

2.4. Additive models

Further, similar to the `amer` package (Scheipl 2011), we provide convenient interfaces for fitting additive models using penalized splines (Ruppert, Wand, and Carroll 2003). All the spline constructors available in `amer` or those defined by the users following the `amer` requirement can be directly used here.

For example, we update the auto insurance loss example by specifying a smooth effect on the continuous covariate `MVR_PTS`.

```
R> P2 <- cpglmm(CLM_AMT5 ~ CAR_USE + MARRIED + AREA + tp(MVR_PTS), data = da)
```

We can extract and plot the smooth terms using the `getF` and `plotF` function from `amer`. For example, `plotF` generates the graph of the smooth term on the linear predictor scale as in Figure 1.

```
R> plotF(P2, rug = FALSE)
```

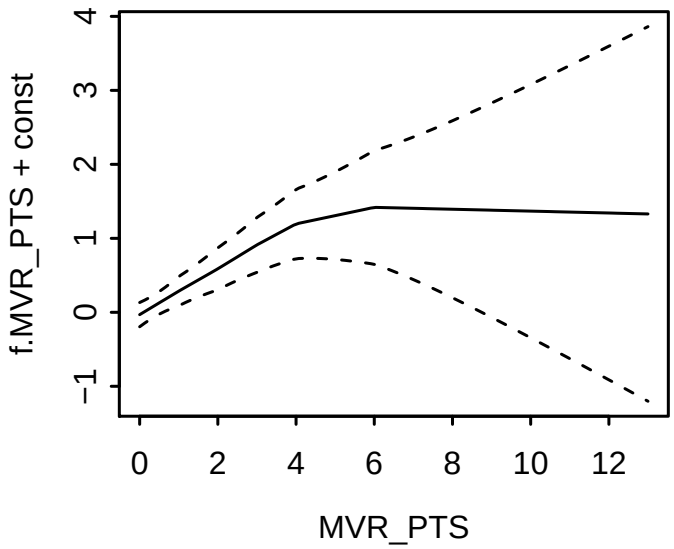

Figure 1: Plot of the smooth term on the linear predictor scale.

2.5. Bayesian methods

The above models can also be formulated in the Bayesian setting, in which parameter inference is carried out via Markov chain Monte Carlo [MCMC] methods.

In the Bayesian model, prior distributions have to be specified for all parameters in the model. The prior distributions we use are

$$\boldsymbol{\beta} \sim N(\mathbf{0}, 10000 \cdot \boldsymbol{I}); \tag{7}$$
$$\phi \sim U(0, 100); \tag{8}$$
$$p \sim U(1.01, 1.99), \tag{9}$$

where the prior mean and variance of $\boldsymbol{\beta}$ and the bounds for ϕ and p can be modified using the argument `prior.beta.mean`, `prior.beta.var`, `bound.phi` and `bound.p`, respectively.

If a mixed model is specified, we specify the prior distribution of the variance component as follows. If there is one random effect in a group, the inverse Gamma distribution is used; and if there is more than one random effects in a group, the inverse Wishart distribution is used:

$$\sigma_b^{-2} \sim Gamma(0.001, 1000); \tag{10}$$
$$\boldsymbol{\Sigma}^{-1} \sim Wishart_d(\boldsymbol{I}), \tag{11}$$

where d is the dimension of $\boldsymbol{\Sigma}$. Currently, the parameter values in these prior distributions cannot be modified by the user.

In implementing the MCMC, we construct a Gibbs sampler in which parameters are updated one at a time given the current values of all the other parameters. Specifically, we use the random-walk Metropolis-Hastings algorithm in updating each parameter except for the variance components, which can be simulated directly due to conjugacy.

There is a tuning process in the MCMC algorithm, where the variances of the proposal (truncated) normal distributions are updated according to the sample variances computed from the simulations. The goal is to make the acceptance rate roughly between 40% and 60% for univariate M-H updates.

The argument `tune.iter` determines how many iterations are used for the tuning process, and `n.tune` determines how many loops the tuning iterations should be divided into. These iterations will not be used in the final output.

The `bcplm` function is used to fit both GLMs and mixed models using MCMC. We illustrate the Bayesian GLM using the claim triangle data from Peters *et al.* (2009), in which the Tweedie compound Poisson distribution is specified for a strictly positive but highly skewed response variable `increLoss`.

```
R> set.seed(10)
R> B1 <- bcplm(increLoss ~ factor(year) + factor(lag), data = ClaimTriangle,
+              n.chains = 2, n.iter = 7000, tune.iter = 4000,
+              n.burnin = 2000, n.thin = 5, bound.p = c(1.1, 1.95))
```

In the above, we specify the lower bound of p to be greater than 1.1. This is often a safe guard to regions where the compound Poisson distribution is multi-modal (Dunn and Smyth 2005). By default, the function prints out intermediate acceptance information of the M-H algorithms, which is often useful to determine whether good proposal variances have been obtained in the tuning phase. For example, a run of the above function prints out the following:

```
Tuning phase...
Acceptance rate: min(34.00%), mean(51.62%), max(63.00%)
----------------------------------------
Start Markov chain 1
Iteration: 3500
 Acceptance rate: min(41.63%), mean(48.60%), max(55.14%)
Iteration: 7000
 Acceptance rate: min(42.64%), mean(48.63%), max(55.27%)
----------------------------------------
Start Markov chain 2
Iteration: 3500
 Acceptance rate: min(42.60%), mean(49.41%), max(57.60%)
Iteration: 7000
 Acceptance rate: min(42.59%), mean(49.33%), max(57.31%)
----------------------------------------
Markov Chain Monte Carlo ends!
```

The simulation results are saved in the `sims.list` slot, which is an `"mcmc.list"` object that works with many methods defined in the `coda` package (Plummer, Best, Cowles, and Vines 2012). For example, we can use the `gelman.diag` function to compute the 'potential scale reduction factor' (Gelman and Rubin 1992) in order to determine approximate convergence:

```
R> summary(gelman.diag(B1$sims.list)[[1]][, 1])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.9992  1.0000  1.0030  1.0040  1.0080  1.0220
```

In addition, we can also use visual diagnostic tools such as the trace and density plots- Figure 2 and 3 are generated by the following commands:

```
R> xyplot(B1$sims.list[, c(1:2, 20, 21)], xlab = NULL)
R> densityplot(B1$sims.list[, c(1:2, 20, 21)], ylab = NULL)
```

We see that despite that all the potential scale reduction factors are very close to 1, these diagnostic plots suggest that better mixing can be achieved if the chains are run longer.

We conclude this section with a Bayesian example on compound Poisson mixed models. We now use MCMC methods to draw inference of the fine root mixed model (`f0`). The estimates of the variance components in Bayesian models are generally less biased than those based on maximum likelihood procedures when there are few levels in the grouping factor (Zhang 2013).

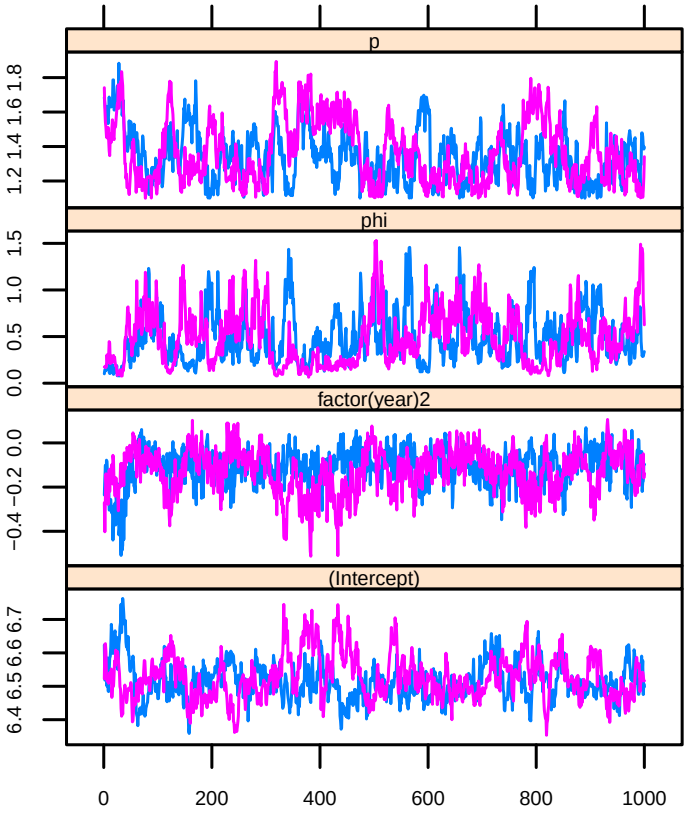

Figure 2: Trace plot of the two chains in the claim triangle example.

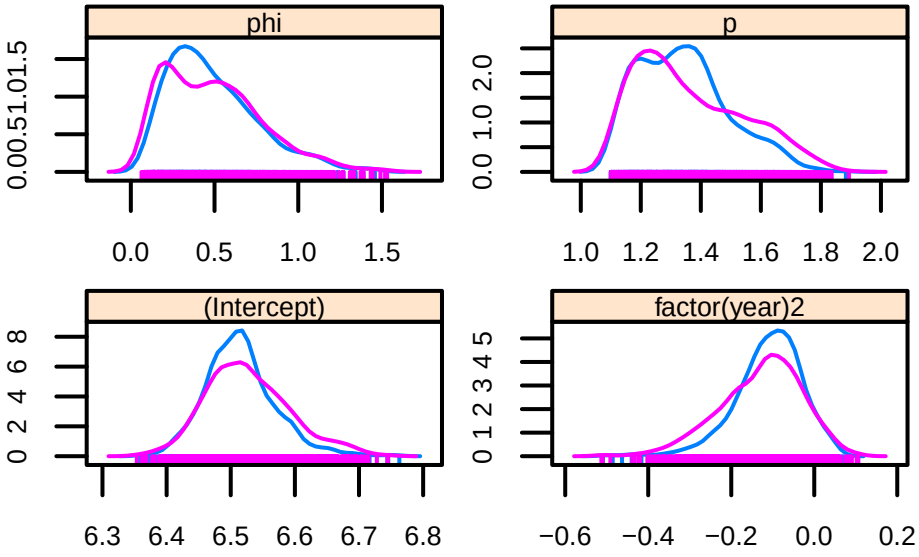

Figure 3: Density plot of the two chains in the claim triangle example.

```
R> set.seed(10)
R> B2 <- bcplm(RLD ~ Stock * Zone + (1|Plant),
+              data = FineRoot, n.iter = 11000,
+              n.burnin = 1000, n.thin = 10)
```

The inference results can be summarized by

```
R> summary(B2)

Compound Poisson linear models via MCMC
3 chains, each with 11000 iterations (first 1000 discarded), n.thin = 10
n.sims = 3000 iterations saved

Formula: RLD ~ Stock * Zone + (1 | Plant)
   Data: FineRoot

Random and dynamic variance components:
 Groups   Name        Variance Std.Dev.
 Plant    (Intercept) 0.024784 0.15743
 Residual             0.337777 0.58119
Number of obs: 511 , groups: Plant, 8

Fixed effects:
                    Estimate Std. Error Lower (2.5%) Upper (97.5%)
(Intercept)          -2.08925    0.21928     -2.52060        -1.626
StockMM106           -0.07313    0.29603     -0.71670         0.505
StockMark            -0.47743    0.27231     -1.04190         0.041
ZoneOuter            -0.45524    0.25598     -0.96740         0.036
StockMM106:ZoneOuter  0.03198    0.30952     -0.56080         0.646
StockMark:ZoneOuter  -1.15131    0.32381     -1.76360        -0.505
---
Estimated dispersion parameter: 0.33778
Estimated index parameter: 1.4178
```

These results are in line with those in Zhang (2013). We see that the estimate of the variance component is about twice as large as that based on maximum likelihood estimation.

2.6. The Gini index

For model comparison involving the compound Poisson distribution, the usual mean squared loss function is not quite informative for capturing the differences between predictions and observations, due to the high proportions of zeros and the skewed heavy-tailed distribution of the positive outcomes. For this reason, Frees, Meyers, and Cummings (2011) develop an ordered version of the Lorenz curve and the associated Gini index as a statistical measure of the association between distributions, through which different predictive models can be compared. The idea is that a score (model) with a greater Gini index produces a greater separation among the observations. In the insurance context, a higher Gini index indicates greater ability to distinguish good risks from bad risks. Therefore, the model with the highest Gini index is preferred.

Continuing using insurance as an example, we let y_i be the observed loss payment, P_i be the baseline premium, S_i be the insurance score (predictions from the model), $R_i = S_i/P_i$ be the relativity, for the i_{th} observation. We sort all the policies by the relativities in an increasing order, and compute the empirical cumulative premium and loss distributions as

$$\hat{F}_P(s) = \frac{\sum_{i=1}^n P_i \cdot \mathbb{1}(R_i \leq s)}{\sum_{i=1}^n P_i}, \ \hat{F}_L(s) = \frac{\sum_{i=1}^n y_i \cdot \mathbb{1}(R_i \leq s)}{\sum_{i=1}^n y_i}. \tag{12}$$

The graph $\left(\hat{F}_P(s), \hat{F}_L(s)\right)$ is an ordered Lorenz curve. The area between the Lorenz curve and the line of equality (a 45 degree line) is measure of the discrepancy between the premium and loss distributions, and twice this area is defined as the Gini index. Intuitively, the Gini index can be regarded as the average expected profit that can be gained under the new pricing scheme defined by the score.

The function `gini` computes the Gini indices and their asymptotic standard errors based on the ordered Lorenz curve. These metrics are mainly used for model comparison. Depending on the problem, there are generally two ways to do this. Take insurance predictive modeling as an example. First, when there is a baseline premium, we can compute the Gini index for each score (predictions from the model), and select the model with the highest Gini index. Second, when there is no baseline premium (`base = NULL`), we successively specify the prediction from each model as the baseline premium and use the remaining models as the scores. This results in a matrix of Gini indices, and we select the model that is least vulnerable to alternative models using a "mini-max" argument - we select the score that provides the smallest of the maximal Gini indices, taken over competing scores.

For example, we use the Gini index to compare the three models `P1`, `P2` and `P3` in the auto insurance example. This can be done as follows:

```
R> da <- transform(da, P1 = fitted(P1), P2 = fitted(P2), P3 = fitted(P3))
R> gg <- gini(loss = "CLM_AMT5", score  = paste("P", 1:3, sep = ""),
+             data = da)
R> gg

Call:
gini(loss = "CLM_AMT5", score = paste("P", 1:3, sep = ""), data = da)

Gini indices:
    P1      P2      P3
P1   0.000  12.028  11.880
P2  -1.085   0.000   3.345
P3   3.875   5.977   0.000

Standard errors:
    P1     P2     P3
P1  0.000  2.148  2.098
P2  2.191  0.000  2.295
P3  2.106  2.275  0.000
```

We see that the GLM-type model is the worst, while according to the "mini-max" argument, the selected best model is the additive model `P2`. The `"gini"` object has a slot named `lorenz` that stores the graph for the underlying ordered Lorenz curve. A `plot` method is defined to extract and plot the graphs of the Lorenz curve using `ggplot2`:

```
R> theme_set(theme_bw())
R> plot(gg, overlay = FALSE)
```

3. Acknowledgement

- The `cpglmm` function is based on the source code of the `lme4.0` package, with changes made on updating of the mean, the variance function and the marginal loglikelihood. Major credits go to the authors of the package.
- The functionality on penalized splines is due to Fabian Scheipl `<fabian.scheipl@googlemail.com>`, who wrote the original `amer` package.

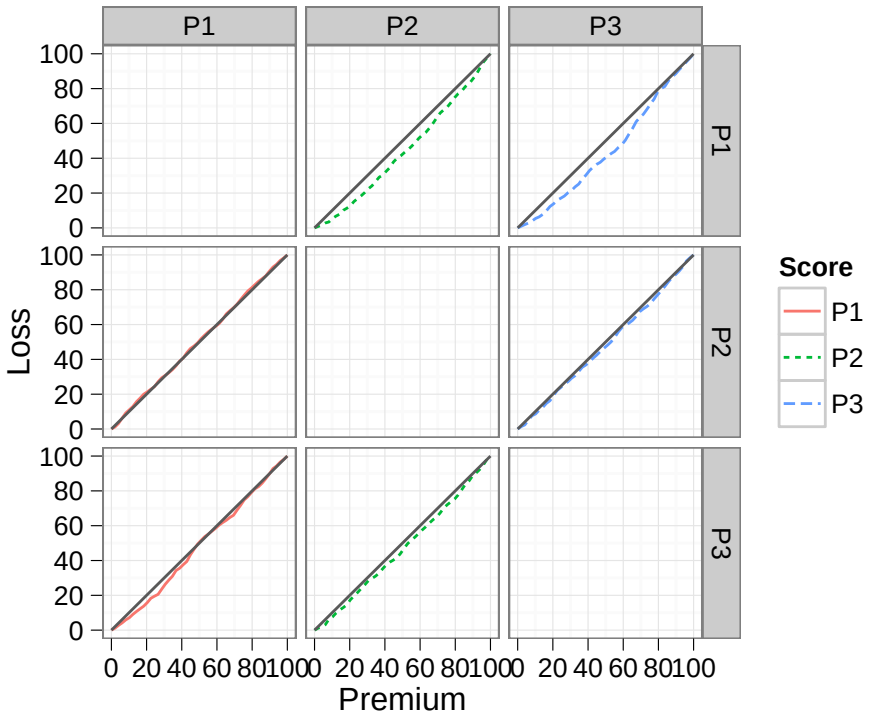

Figure 4: Plot of the ordered Lorenz curves for the auto insurance example.

References

Chambers J (1998). *Programming with Data. A Guide to the S Language*. Springer-Verlag, New York.

Cox DR, Reid N (1987). "Parameter Orthogonality and Approximate Conditional Inference." *Journal of the Royal Statistical Society, Series B*, **49**, 1–39.

Davidian M (1990). "Estimation of Variance Functions in Assays with Possible Unequal Replication and Nonnormal Data." *Biometrika*, **77**, 43–54.

Davidian M, Carroll RJ (1987). "Variance Function Estimation." *Journal of the American Statistical Association*, **82**, 1079–1091.

de Silva HN, Hall AJ, Tustin DS, Gandar PW (1999). "Analysis of Distribution of Root Length Density of Apple Trees on Different Dwarfing Rootstocks." *Annals of Botany*, **83**, 335–345.

Dunn PK (2004). "Occurrence and Quantity of Precipitation Can Be Modelled Simultaneously." *International Journal of Climatology*, **24**, 1231–1239.

Dunn PK (2011). **tweedie**: *Tweedie Exponential Family Models*. R package version 2.1.1, URL http://cran.r-project.org/web/packages/tweedie.

Dunn PK, Smyth GK (2005). "Series Evaluation of Tweedie Exponential Dispersion Models Densities." *Statistics and Computing*, **15**, 267–280.

Dunn PK, Smyth GK (2008). "Evaluation of Tweedie Exponential Dispersion Model Densities by Fourier Inversion." *Statistics and Computing*, **18**, 73–86.

Frees EW, Meyers G, Cummings DA (2011). "Summarizing Insurance Scores Using a Gini Index." *Journal of the American Statistical Association*, **495**, 1085 – 1098.

Gelman A, Carlin J, Stern H, Rubin D (2003). *Bayesian Data Analysis*. CRC Press, Boca Raton, 2 edition.

Gelman A, Rubin D (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**, 457–511.

Hougaard P, Harvald B, Holm N (1992). "Measuring the Similarities Between the Lifetimes of Adult Danish Twins Born Between 1881-1930." *Journal of the American Statistical Association*, **87**, 17–24.

Jørgensen B (1987). "Exponential Dispersion Models (with discussion)." *Journal of the Royal Statistical Society, Series B*, **49**, 127–162.

Klugman AS, Panjer HH, Willmot EG (2008). *Loss models: from data to decisions*. John Wiley & Sons.

Lumley T (2011). **biglm***: Bounded Memory Linear and Generalized Linear Models*. R package version 0.8, URL http://cran.r-project.org/web/packages/biglm.

McCullagh P, Nelder J (1989). *Generalized Linear Models*. Chapman and Hall, Boca Raton, 2 edition.

Nelder J, Pregibon D (1987). "An Extended Quasi-likelihood Function." *Biometrika*, **74**, 221–232.

Perry J (1981). "Taylor's Power Law for Dependence of Variance on Mean in Animal Populations." *Journal of the Royal Statistical Society, Series C*, **30**, 254–263.

Peters GW, Shevchenko PV, Wüthrich MV (2009). "Model Uncertainty in Claims Reserving within Tweedie's Compound Poisson Models." *ASTIN Bulletin*, **39**, 1–33.

Plummer M, Best N, Cowles K, Vines K (2012). **coda***: Output Analysis and Diagnostics for MCMC*. R package version 0.14-7, URL http://cran.r-project.org/web/packages/coda.

Ruppert D, Wand M, Carroll R (2003). *Semiparametric Regression*. Cambridge University Press.

Scheipl F (2011). **amer***: Additive Mixed Models with* **lme4**. R package version 0.6.10, URL http://cran.r-project.org/web/packages/amer.

Shono H (2008). "Application of the Tweedie Distribution to Zero-catch Data in CPUE Analysis." *Fisheries Research*, **93**, 154–162.

Smyth G, Jørgensen B (2002). "Fitting Tweedie's Compound Poisson Model to Insurance Claims Data: Dispersion Modelling." *ASTIN Bulletin*, **32**, 143–157.

Yip KCH, Yau KKW (2005). "On Modeling Claim Frequency Data In General Insurance With Extra Zeros." *Insurance: Mathematics and Economics*, **36**, 153–163.

Zhang Y (2013). "Likelihood-based and Bayesian Methods for Tweedie Compound Poisson Linear Mixed Models." *Statistics and Computing*, **23**, 743–757.