

# MGLM package vignette

Yiwen Zhang      Hua Zhou

March 25, 2017

The analysis of multivariate count data arises in numerous fields including genomics, image analysis, text mining, and sport analytics. The multinomial logit model is limiting due to its restrictive mean-variance structure. Moreover, it assumes that counts of different categories are negatively correlated. Models that allow over-dispersion and possess more flexible positive and/or negative correlation structures offer more realism. We implement four models in the R package **MGLM**: multinomial logit (MN), Dirichlet multinomial (DM), generalized Dirichlet multinomial (GDM), and negative multinomial (NegMN). Distribution fitting, regression, hypothesis testing, and variable selection are treated in a unified framework. The multivariate count data we analyze here has  $d$  categories.

## 1 Distribution fitting

The function **MGLMfit** fits various multivariate discrete distributions and outputs a list with the maximum likelihood estimate (MLE) and relevant statistics.

When fitting distributions, i.e. no covariates involved, MN is a sub-model of DM, and DM is a sub-model of GDM. **MGLMfit** outputs the p-value of the likelihood ratio test (LRT) for comparing the fitted model with the most commonly used multinomial model. The NegMN model does not have a nesting relationship with any of the other three models. Therefore, no LRT is performed when fitting a NegMN distribution.

### 1.1 Multinomial (MN)

We first generate data from a multinomial distribution. Note the multinomial parameter (must be positive) supplied to the **rmn** function is automatically scaled to be a probability vector.

```
> require(MGLM)
> set.seed(123)
> n <- 200
> d <- 4
> alpha <- rep(1, d)
> m <- 50
> Y <- rmn(n, m, alpha)
```

Multinomial distribution fitting, although trivial, is implemented.

```
> mnFit <- MGLMfit(Y, dist="MN")
> print(mnFit)
```

	estimate	SE
alpha_1	0.2568	0.03089124
alpha_2	0.2467	0.03048271
alpha_3	0.2451	0.03041595
alpha_4	0.2514	0.03067556

Distribution: Multinomial  
 Log-likelihood: -1457.788  
 BIC: 2931.471  
 AIC: 2921.576  
 LRT test p value:  
 Iterations:

As a comparison, we fit the DM distribution to the same data set. The results indicate that using a more complex model on the multinomial data shows no advantage.

```

> compareFit <- MGLMfit(Y, dist="DM")
> print(compareFit)

```

	estimate	SE
alpha_1	5270901	786946676
alpha_2	5063596	755995891
alpha_3	5030755	751092797
alpha_4	5160065	770398732

Distribution: Dirichlet Multinomial  
 Log-likelihood: -1457.788  
 BIC: 2936.769  
 AIC: 2923.576  
 LRT test p value: 1.000  
 Iterations: 35

Both the DM parameter estimates and their standard errors are large, indicating possible overfitting by the DM model. This is confirmed by the fact that the p-value of the LRT for comparing MN to DM is close to 1.

## 1.2 Dirichlet-multinomial (DM)

DM is a Dirichlet mixture of multinomials and allows over-dispersion. Similar to the MN model, it assumes that the counts of any two different categories are negatively correlated. We generate the data from the DM model and fit the DM distribution.

```

> set.seed(123)
> n <- 200
> d <- 4
> alpha <- rep(1, d)
> m <- 50
> Y <- rdirm(n, m, alpha)

```

```

> dmFit <- MGLMfit(Y, dist="DM")
> print(dmFit)

              estimate              SE
alpha_1 0.9766705 0.07658856
alpha_2 0.9951423 0.07925470
alpha_3 1.0061205 0.08003311
alpha_4 0.9045003 0.07254733

Distribution: Dirichlet Multinomial
Log-likelihood: -2011.225
BIC: 4043.644
AIC: 4030.451
LRT test p value: <0.0001
Iterations: 4

```

The estimate is very close to the true value with small standard errors. The LRT shows that the DM model is significantly better than the MN model.

### 1.3 Generalized Dirichlet-multinomial (GDM)

GDM model uses  $d - 2$  more parameters than the DM model and allows both positive and negative correlations among different categories. DM is a sub-model of GDM. Here we fit a GDM model to the above DM sample.

```

> gdmFit <- MGLMfit(Y, dist="GDM")
> print(gdmFit)

              estimate              SE
alpha_1 1.1584741 0.12340343
alpha_2 0.9932931 0.43723945
alpha_3 0.8399666 0.10637444
beta_1  3.7068631 0.22641418
beta_2  1.9793891 0.09464476
beta_3  0.7596409 0.08440347

Distribution: Generalized Dirichlet Multinomial
Log-likelihood: -2007.559
BIC: 4046.907
AIC: 4027.117
LRT test p value: <0.0001
Iterations: 27

```

GDM yields a slightly larger log-likelihood value but a larger BIC, suggesting DM as a preferred model. p-value indicates GDM is still significantly better than the MN model. Now we simulate data from GDM and fit the GDM distribution.

```

> set.seed(124)
> n <- 200
> d <- 4
> alpha <- rep(1, d-1)

```

```

> beta <- rep(1, d-1)
> m <- 50
> Y <- rgdirm(n, m, alpha, beta)
> gdmFit <- MGLMfit(Y, dist="GDM")
> print(gdmFit)

              estimate              SE
alpha_1 1.0198347 0.10348011
alpha_2 0.8261251 0.10931635
alpha_3 0.7743869 0.09256818
beta_1  1.0621116 0.09556607
beta_2  0.8462160 0.10872545
beta_3  0.9245595 0.13500770

Distribution: Generalized Dirichlet Multinomial
Log-likelihood: -1820.616
BIC: 3673.021
AIC: 3653.231
LRT test p value: <0.0001
Iterations: 24

```

## 1.4 Negative multinomial (NegMN)

NegMN model is a multivariate analog of the negative binomial model. It assumes positive correlation among the counts. The following code generates data from the NegMN model and fit the NegMN distribution,

```

> set.seed(1220)
> n <- 100
> d <- 4
> p <- 5
> prob <- rep(0.2, d)
> beta <- 10
> Y <- rnegmn(n, prob, beta)
> negmnFit <- MGLMfit(Y, dist="NegMN")
> print(negmnFit)

              estimate              SE
p_1  0.1881512 0.009583840
p_2  0.1943109 0.009837429
p_3  0.1915110 0.009722206
p_4  0.1961775 0.009914205
phi 12.3139348 2.266096911

Distribution: Negative Multinomial
Log-likelihood: -1104.579
BIC: 2232.184
AIC: 2219.158
LRT test p value: NA
Iterations: 4

```

Because MN is not a sub-model of NegMN, no LRT is performed here.

## 2 Regression

In regression, the  $n \times p$  covariate matrix  $X$  is similar to that used in the `glm` function. The response should be a  $n \times d$  count matrix. Unlike estimating a parameter vector  $\beta$  in GLM, we need to estimate a parameter matrix  $B$  when the responses are multivariate. The dimension of the parameter matrix depends on the model:

- MN:  $p \times (d - 1)$
- DM:  $p \times d$
- GDM:  $p \times 2(d - 1)$
- NegMN:  $p \times (d + 1)$

The GDM model provides the most flexibility, but also requires most parameters. In the function `MGLMreg` for regression, the option `dist="MN"`, `"DM"`, `"GDM"` or `"NegMN"` specifies the model.

The rows  $B_{j\cdot}$  of the parameter matrix correspond to covariates. By default, the function output the Wald test statistics and p-values for testing  $H_0 : B_{j\cdot} = \mathbf{0}$  vs  $H_a : B_{j\cdot} \neq \mathbf{0}$ . If specifying the option `LRT=TRUE`, the function also outputs LRT statistics and p-values.

Next, we demonstrate that model mis-specification results in failure of hypothesis testing. We simulate response data from the GDM model. Covariates  $X_1$  and  $X_2$  have no effect while  $X_3$ ,  $X_4$ ,  $X_5$  have different effect sizes.

```
> set.seed(1234)
> n <- 200
> p <- 5
> d <- 4
> X <- matrix(runif(p * n), n, p)
> alpha <- matrix(c(0.6, 0.8, 1), p, d - 1, byrow=TRUE)
> alpha[c(1, 2),] <- 0
> Alpha <- exp(X %*% alpha)
> beta <- matrix(c(1.2, 1, 0.6), p, d - 1, byrow=TRUE)
> beta[c(1, 2),] <- 0
> Beta <- exp(X %*% beta)
> m <- runif(n, min=0, max=25) + 25
> Y <- rgdirm(n, m, Alpha, Beta)
```

We fit various regression models and test significance of covariates.

### 2.1 Multinomial regression

```
> mnReg <- MGLMreg(Y~0+X, dist="MN")
> print(mnReg)
```

Call: `MGLMreg(formula = Y ~ 0 + X, dist = "MN")`

Coefficients:

	Col_1	Col_2	Col_3
X1	0.2770632	-0.1827597	-0.1232039

```

X2  0.5430639  0.4227301  0.2465230
X3  0.3332517  0.5176055  0.2218513
X4  0.3568425  0.4867224  0.5654272
X5 -0.3024545  0.1925076  0.3237132

```

Hypothesis test:

```

      wald value      Pr(>wald)
X1    24.63244 1.842859e-05
X2    21.99680 6.533133e-05
X3    23.10908 3.832310e-05
X4    25.07475 1.489470e-05
X5    49.37327 1.086326e-10

```

```

Distribution: Multinomial
Log-likelihood: -2194.448
BIC: 4468.371
AIC: 4418.896
Iterations: 5

```

The Wald test shows that all predictors, including the null predictors  $X_1$  and  $X_2$ , are significant.

## 2.2 Dirichlet-multinomial regression

```

> dmReg <- MGLMreg(Y~0+X, dist="DM")
> print(dmReg)

```

```

Call: MGLMreg(formula = Y ~ 0 + X, dist = "DM")

```

Coefficients:

```

      Col_1      Col_2      Col_3      Col_4
X1 0.1541366 -0.1182637 -0.1883392 -0.01317229
X2 0.1832642  0.1420338 -0.1833943 -0.33388600
X3 1.1431456  1.2548276  1.0926352  0.81125874
X4 0.3927028  0.5454214  0.5900146  0.13113218
X5 0.2263497  0.6601081  0.9395988  0.48703415

```

Hypothesis test:

```

      wald value      Pr(>wald)
X1    3.349794 5.010817e-01
X2    7.845339 9.741075e-02
X3   25.497386 3.995529e-05
X4    8.735121 6.807217e-02
X5   23.136042 1.189431e-04

```

```

Distribution: Dirichlet Multinomial
Log-likelihood: -1683.961
BIC: 3473.889
AIC: 3407.922
Iterations: 7

```

Wald test declares that  $X_1$ ,  $X_2$  and  $X_4$  have not effects, but  $X_3$  and  $X_5$  are significant.

## 2.3 Generalized Dirichlet-multinomial Regression

```
> gdmReg <- MGLMreg(Y~0+X, dist="GDM")
> print(gdmReg)
```

Call: MGLMreg(formula =  $Y \sim 0 + X$ , dist = "GDM")

Coefficients:

	alpha_Col_1	alpha_Col_2	alpha_Col_3	beta_Col_1	beta_Col_2	beta_Col_3
X1	-0.2839174	0.19050322	0.31570552	-0.4002027	0.6846506	0.4675918
X2	-0.2091710	0.39554111	0.01442559	-0.5082543	0.5526452	-0.1714096
X3	1.0901404	1.24378465	1.17178640	1.3049550	1.5375262	0.9160329
X4	0.2968186	0.40533348	0.80908676	0.4878379	0.5736954	0.3058887
X5	0.6018408	0.03012328	1.28121907	1.4309306	0.2601967	0.8534981

Hypothesis test:

	wald value	Pr(>wald)
X1	9.424379	1.510802e-01
X2	4.865683	5.611523e-01
X3	26.828718	1.559064e-04
X4	12.607034	4.971848e-02
X5	38.637877	8.427803e-07

Distribution: Generalized Dirichlet Multinomial  
 Log-likelihood: -1676.399  
 BIC: 3511.748  
 AIC: 3412.798  
 Iterations: 21

When using the correct model GDM, the Wald test is able to differentiate the null effects from the significant ones. GDM regression yields the highest log-likelihood and smallest BIC.

## 2.4 Negative multinomial regression

```
> negReg <- MGLMreg(Y~0+X, dist="NegMN", regBeta=FALSE)
> print(negReg)
```

Call: MGLMreg(formula =  $Y \sim 0 + X$ , dist = "NegMN", regBeta = FALSE)

Coefficients:

	Col_1	Col_2	Col_3	Col_4
X1	0.24360582	-0.21636792	-0.15652499	-0.03137138
X2	0.06189622	-0.05588488	-0.23263498	-0.47977959
X3	-0.16091394	0.02268791	-0.27123839	-0.49512726
X4	-0.17618094	-0.04382845	0.03478118	-0.53012642
X5	-0.60797439	-0.11582939	0.01293699	-0.31585689

\$phi

13.77531

Hypothesis test:

	wald	value	Pr(>wald)
X1	24.99903	5.033252e-05	
X2	24.90077	5.267448e-05	
X3	29.32830	6.704198e-06	
X4	28.18693	1.143075e-05	
X5	60.50179	2.275444e-12	

Distribution: Negative Multinomial

Log-likelihood: -2908.896

BIC: 5929.056

AIC: 5859.792

Iterations: 15

Again, the Wald test declares all predictors to be significant.

## 2.5 Prediction

We can use the fitted model for prediction. The `prediction` function outputs the probabilities of each category. This helps answer questions such as whether certain features increase the probability of observing category  $j$ . Take the fitted GDM model as an example:

```
> newX <- matrix(runif(1*p), 1, p)
> pred <- predict(gdmReg, newX)
> pred
```

	Col_1	Col_2	Col_3	Col_4
[1,]	0.3218286	0.2235816	0.3304694	0.1241204

## 3 Sparse regression

Regularization is an important tool for model selection and improving the risk property of the estimates. In the package, we implemented three types of penalties on the parameter matrix  $B$ :

- selection by entries
- selection by rows/predictors
- selection by rank

The function `MGLMtune` finds the optimal tuning parameter with the smallest BIC and outputs the estimate using the chosen tuning parameter. The output from `MGLMtune` is a list containing the solution path and the final estimate.



Users can either provide a vector of tuning parameters with option `lambdas` or specify the number of grid points via option `ngridpt` and let the function decide the default tuning parameters. The function `MGLMsparsereg` computes the regularized estimate at a given tuning parameter value `lambda`.

We generate the data from the DM model, with row sparsity, and show how each penalty type works.

```
> set.seed(118)
> n <- 100
> p <- 10
> d <- 5
> m <- rbinom(n, 200, 0.8)
> X <- matrix(rnorm(n * p), n, p)
> alpha <- matrix(0, p, d)
> alpha[c(1, 3, 5), ] <- 1
> Alpha <- exp(X %*% alpha)
> Y <- rdirm(size=m, alpha=Alpha)
```

### 3.1 Select by entries

```
> sweep <- MGLMtune(Y ~ 0 + X, dist="DM", penalty="sweep", ngridpt=30)
> print(sweep$select)
```

```
Call: MGLMtune(formula = Y ~ 0 + X, dist = "DM", penalty = "sweep",
               ngridpt = 30)
```

```
Distribution: Dirichlet Multinomial
Log-likelihood: -1461.992
BIC: 3066.744
AIC: 2985.984
Degrees of freedom: 31
Lambda: 4.858822
Iterations: 41
```

### 3.2 Select by rows

Since the rows of the parameter matrix correspond to predictors, selecting by rows performs variable selection at the predictor level.

```
> group <- MGLMtune(Y ~ 0 + X, dist="DM", penalty="group", ngridpt=30)
> print(group$select)
```

```
Call: MGLMtune(formula = Y ~ 0 + X, dist = "DM", penalty = "group",
               ngridpt = 30)
```

```
Distribution: Dirichlet Multinomial
Log-likelihood: -1485.475
BIC: 3079.238
AIC: 3017.978
Degrees of freedom: 23.51454
Lambda: 20.20407
Iterations: 27
```

### 3.3 Select by singular values

Nuclear norm regularization encourages low rank in the regularized estimate.

```
> nuclear <- MGLMtune(Y ~ 0 + X, dist="DM", penalty="nuclear", ngridpt=30, warm.start=FALSE)
> print(nuclear$select)
```

```
Call: MGLMtune(formula = Y ~ 0 + X, dist = "DM", penalty = "nuclear",
  ngridpt = 30, warm.start = FALSE)
```

```
Distribution: Dirichlet Multinomial
Log-likelihood: -1492.063
BIC: 3070.422
AIC: 3021.604
Degrees of freedom: 18.7391
Lambda: 37.53776
Iterations: 32
```