Automultinomial Vignette

Stephen Berg

March 27, 2018

This vignette explains the installation and use of the R package automultinomial. The automultinomial package is designed to be used for regressions similar to logistic or multinomial logit regression. However, unlike ordinary logistic and multinomial logit models, the autologistic/automultinomial model includes an autocorrelation parameter to account for spatial dependence between observations.

The organization of this document is:

1. Description of the problem automultinomial solves
2. Github installation how-to
3. A data example with binary response (2 response categories)
4. A data example with 3 response categories

The problem automultinomial solves

Consider a data problem where covariates (the independent variables) and categorical outcomes (the dependent variables) are observed on a spatial grid or lattice. If the outcomes are spatially autocorrelated, then our coefficient estimates and inference procedures ought to take this into account. Intuitively, when outcomes are correlated, the effective sample size of the dataset is smaller than if the outcomes were independent. The statistical information we can gain based on samples from multiple nearby sites may be less than if we were able to sample sites that are "more independent".

A particular practical risk in neglecting spatial correlation is that confidence intervals may be too narrow. We might find "statistically significant" relationships that are in fact byproducts of noise or dependent sampling. The automultinomial package provides a tool for dealing with outcomes with spatial autocorrelation. It may happen that even nearby sites in a dataset are still effectively independent. But we can only assess this using a method that accounts for the spatial arrangement of the samples.

Having given a qualitative description of the data problem, we will now move on to a discussion of the probability model used by automultinomial.

Mathematical description

To describe the setup in automultinomial, we will use the notations

- $i = 1, ..., n$ to index the n observation sites on the spatial grid
- $K \geq 2$ to denote the number of possible values taken by the response
- p to denote the number of covariates observed at each site
- $x_i \in \mathbb{R}^p$ to denote the values of the covariates at site i
- $z_i = 1, ..., K$ to denote the response at site i
- $\mathbf{z}$ to denote the vector of categorical responses for all the sites
- z_{-i} to denote the entire response vector $\mathbf{z}$, *except* for site i
- $\boldsymbol{\beta} \in \mathbb{R}^{p \times (K-1)}$ to denote the coefficient matrix (when $K = 2$, $\boldsymbol{\beta}$ is a vector of length p, as in logistic regression)
- β_k to denote the k-th column of $\boldsymbol{\beta}$.
- γ to denote the autocorrelation parameter

We will assume that for each site i, a collection N_i of neighboring sites is known. If the data are collected in a square lattice fashion, then a natural neighborhood setup is to set N_i to be the 4 Up, Down, Left, and Right

neighbors of site i (sites at the boundary may have less than 4 neighbors). We will assume $i \notin N_i$ for each i, so that site i is not a neighbor of itself. It is allowable for sites to have no neighbors. In this case $N_i = \phi$, where ϕ denotes the empty set. We will use the notation $i' \sim i$ to indicate that site i' is a neighbor of site i.

In the automultinomial package, we will require that the neighborhoods are *symmetric*: if $i' \in N_i$, then necessarily $i \in N_{i'}$ as well.

We will also define an *energy* function $H(\mathbf{z}|\boldsymbol{\beta}, \gamma)$. When $K = 2$ and the two categories are $k = 1$ and $k = 2$, then $\boldsymbol{\beta}$ is just a vector with length p, and

$$H(\mathbf{z}|\beta, \gamma) = \sum_{i=1}^{n} x_i^T \beta I(z_i = 2) + \gamma \sum_{i=1}^{n} \sum_{\substack{i' \sim i \\ i' > i}} \sum_{k=1}^{2} I(z_i = z_{i'} = k) \tag{1}$$

In general, for $K \geq 2$ we define $H(\mathbf{z}|\boldsymbol{\beta}, \gamma)$ by

$$H(\mathbf{z}|\beta, \gamma) = \sum_{i=1}^{n} \sum_{k=1}^{K-1} x_i^T \beta_k I(z_i = k+1) + \gamma \sum_{i=1}^{n} \sum_{\substack{i' \sim i \\ i' > i}} \sum_{k=1}^{K} I(z_i = z_{i'} = k) \tag{2}$$

With this preamble accomplished, we can define the probability model automultinomial seeks to estimate:

$$p(\mathbf{z}|\boldsymbol{\beta}, \gamma) = \frac{\exp\{H(\mathbf{z}|\boldsymbol{\beta}, \gamma)\}}{\sum_{\mathbf{z}'} \exp\{H(\mathbf{z}'|\boldsymbol{\beta}, \gamma)\}} \tag{3}$$

The term $\sum_{\mathbf{z}'}$ indicates a sum over all possible categorical responses for the entire dataset.

Some intuition regarding (3) can be gained as follows: when $\gamma = 0$, then $\gamma \sum_{i=1}^{n} \sum_{\substack{i' \sim i \\ i' > i}} \sum_{k=1}^{2} I(z_i = z_{i'} = k) = 0$ so that the z_i are completely independent. In this case the autologistic (automultinomial) model is the same as an ordinary logistic (multinomial logit) model. On the other hand, when $\gamma > 0$, then response configurations $\mathbf{z}$ where $z_i = z_{i'}$ for many neighbor pairs $i \sim i'$ becomes more likely. In this way, γ incorporates positive spatial correlation into the responses. When $\gamma < 0$, we expect neighboring $z_i, z_{i'}$ to *disagree* more frequently than if the z_i were independent, and the $\gamma < 0$ case will in practice be less common.

Estimating $\boldsymbol{\beta}$ and γ

The automultinomial package follows the pseudolikelihood approach of (Besag, 1974) to estimate the parameters β, γ. We briefly explain the procedure and its motivation below.

When γ is known to be 0, then the responses z_i are independent, and we can use common maximum likelihood techniques such as logistic or multinomial regression to estimate β. On the other hand, when spatial correlation is present and we use the automultinomial framework to account for it, then we need to estimate $\boldsymbol{\beta}$ and γ jointly. Unfortunately, maximum likelihood is computationally infeasible due the well known computational intractability of the denominator of (3) when $\gamma \neq 0$.

Fortunately, a proposal of (Besag, 1974) provides a consistent estimation procedure, maximum pseudolikelihood, for the parameters of the model in equation (3). In maximum pseudolikelihood, we maximize

$$\ell_{PL} = \sum_{i=1}^{n} \log\{p(z_i|z_{-i}, \boldsymbol{\beta}, \gamma)\} \tag{4}$$

over β and γ. The expression $p(z_i|z_{-i}, \boldsymbol{\beta}, \gamma)$ refers to the conditional density of z_i given the sites at all of the other grid locations. A short calculation based on (3) shows that for $K = 2$

$$p(z_i = 1|z_{-i}, \boldsymbol{\beta}, \gamma) = \frac{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\}}{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\} + \exp\{x_i^T \beta I(z_i = 2) + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}} \quad (5)$$

$$p(z_i = 2|z_{-i}, \boldsymbol{\beta}, \gamma) = \frac{\exp\{x_i^T \beta I(z_i = 2) + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}}{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\} + \sum_{k=1}^{K-1} \exp\{x_k^T \beta I(z_i = 2) + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}} \quad (6)$$

For $K \geq 2$ response categories, we have

$$p(z_i = 1|z_{-i}, \boldsymbol{\beta}, \gamma) = \frac{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\}}{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\} + \sum_{k=1}^{K-1} \exp\{x_k^T \beta_k I(z_i = k + 1) + \gamma \sum_{i' \sim i} I(z_{i'} = k + 1)\}} \quad (7)$$

and for $k > 1$,

$$p(z_i = k|z_{-i}, \boldsymbol{\beta}, \gamma) = \frac{\exp\{x_i^T \beta_k I(z_i = k + 1) + \gamma \sum_{i' \sim i} I(z_{i'} = k + 1)\}\}}{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\} + \sum_{k=1}^{K-1} \exp\{x_k^T \beta_k I(z_i = k + 1) + \gamma \sum_{i' \sim i} I(z_{i'} = k + 1)\}} \quad (8)$$

In the automultinomial package, equations (5), (6), (7), and (8) and plugged into (4), and (4) is optimized over β and γ using the function optim.

Installing from Github

Installing from Github is (I think) as easy as installing from cran. First, make sure Hadley Wickham's package devtools is installed on your RStudio:

```
install.packages("devtools")
```

Then, run the following:

```
devtools::install_github(repo="stephenberg/automultinomial2")
library(automultinomial2)
```

Now the package is ready to run.

Data example 1: K=2 response categories

Here, we will demonstrate how to simulate data using automultinomial, how to fit data using automultinomial, and how to analyze the output.

Simulating data

Generating simulated data using the function drawSamples().

```r
library(automultinomial2)
set.seed(42)

#10 predictors
p=5

#n times n grid
n=40

#make grid and adjacency matrix
latticeGraph=igraph::make_lattice(c(n,n))
A=igraph::get.adjacency(latticeGraph)

#set coefficient values
beta=matrix(rnorm(p),ncol=1)*0.3
beta
```

```
##             [,1]
## [1,]  0.4112875
## [2,] -0.1694095
## [3,]  0.1089385
## [4,]  0.1898588
## [5,]  0.1212805
```

```r
#set covariate values
X=matrix(rnorm(n^2*p),ncol=p)

#set the correlation parameter value (0.7 is a moderate amount of spatial correlation)
gamma=0.7

#use drawSamples to simulate data with parameters beta and gamma by Gibbs sampling
y=drawSamples(beta,gamma,X,A,nSamples = 1)
```

```
## Burn-in samples
## 100  burn-in samples so far
## 200  burn-in samples so far
## 300  burn-in samples so far
## Drawing samples
```

Figure 1 shows plots of the responses on the grid. On the left plot, we can see "clumping" of the responses due to the positive autocorrelation parameter $\gamma = 0.7$. The right hand plot is from the distribution with the same β, and $\gamma = 0$.

Fitting an autologistic model to the data (K=2 categories)

Now we'll fit an autologistic model using the MPLE function. Then, we will examine confidence intervals for the fitted parameters. By choosing method="asymptotic", MPLE will output confidence intervals based on the asymptotic distribution of the pseudolikelihood estimator to confidence intervals based on the parametric bootstrap. The asymptotic type confidence intervals can be computed very quickly, but may be less accurate in practice than bootstrap confidence intervals. To compute bootstrap confidence intervals, set method="boot". We will set the number of bootstrap samples using the nBoot argument.

We can access information from the fitted model object, but we can also use the MPLE_summary function to view the model summary in table form.

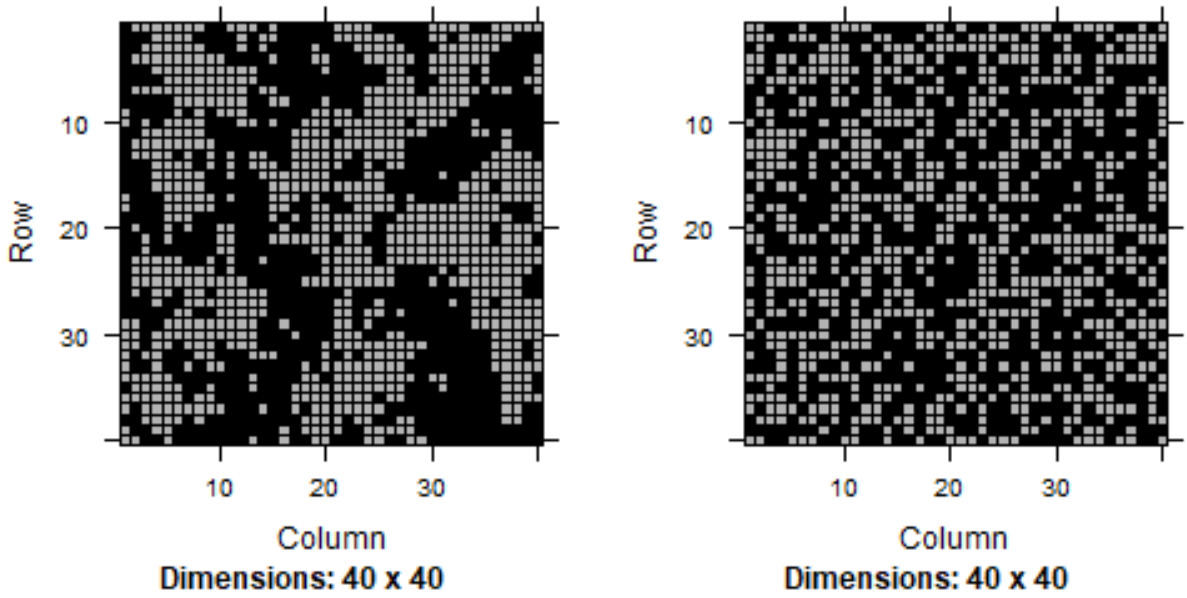


Figure 1: On the left, a plot of the dataset we just simulated, where the truth is $\gamma = 0.7$. For comparison, the plot on the right shows data taken from the same model but with $\gamma = 0$.

```
# responses must be input as a factor
y=factor(y)
fit1=automultinomial2::MPLE(X=X,y=y,A=A,ciLevel = 0.95,method = "asymptotic")
```

```
## Starting model fitting
## Model fitting done, starting variance estimation
## Creating asymptotic confidence intervals
```

```
fit2=automultinomial2::MPLE(X=X,y=y,A=A,ciLevel = 0.95,method = "boot",nBoot = 1)
```

```
## Starting model fitting
## Model fitting done, starting variance estimation
## Creating boot confidence intervals
## Burn-in samples
## 100  burn-in samples so far
## 200  burn-in samples so far
## 300  burn-in samples so far
## Starting bootstrap sampling and fitting
```

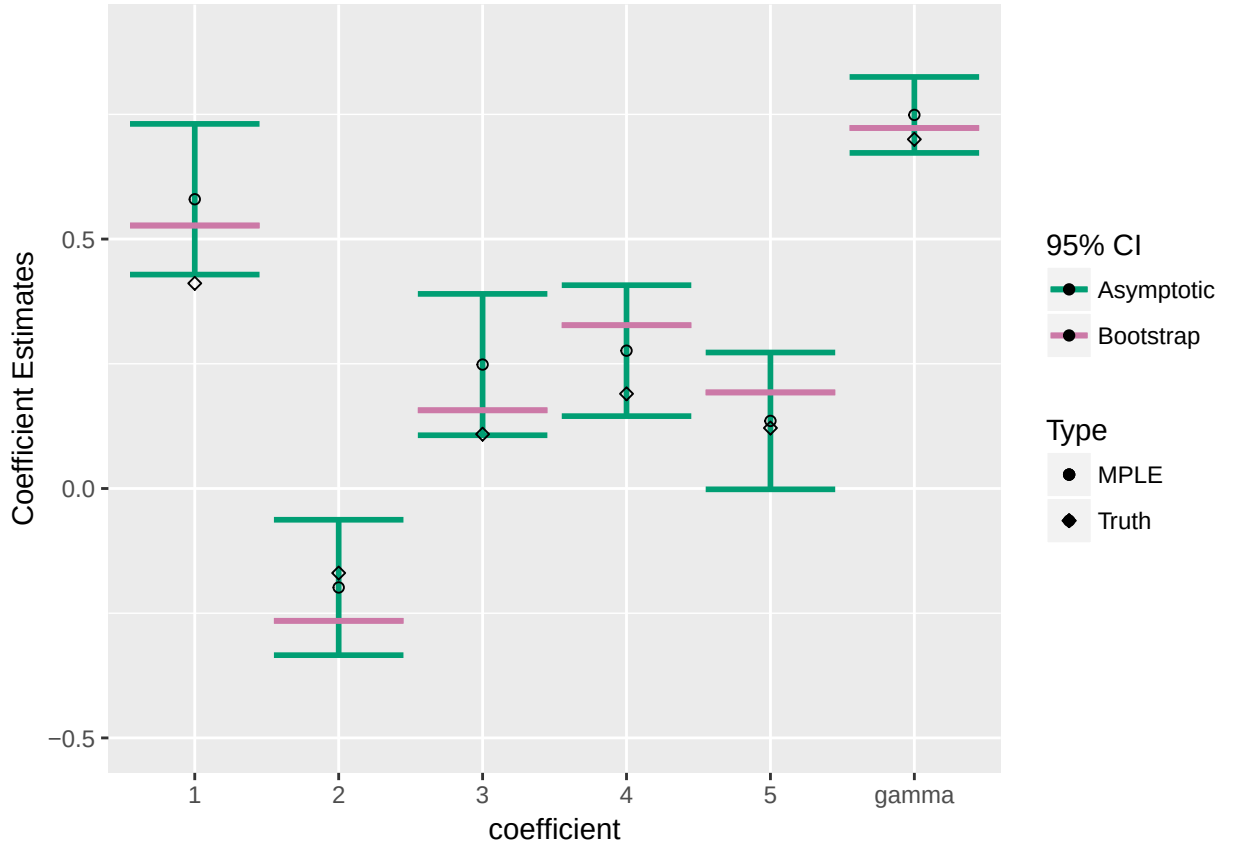
```
#to see the information contained in fit1 and fit2, use str() (not run to save space)
#str(fit1)
#str(fit2)

fitSummary1=MPLE_summary(fit1)
```

```
##
##
```

```
## Table: Summary with confidence intervals
##
##      2 vs. 1                  gamma
## ---  -----------------------  --------------------
## 1    0.58 (0.429,0.731)       0.749 (0.673,0.825)
## 2    -0.198 (-0.334,-0.063)
## 3    0.248 (0.107,0.39)
## 4    0.276 (0.145,0.408)
## 5    0.135 (-0.002,0.273)
##
##
## Table: Summary with p-values
##
##      2 vs. 1          gamma
## ---  ---------------  ----------
## 1    0.58 (0)         0.749 (0)
## 2    -0.198 (0.004)
## 3    0.248 (0.001)
## 4    0.276 (0)
## 5    0.135 (0.053)
```

```
fitSummary2=MPLE_summary(fit2)
```

```
##
##
## Table: Summary with confidence intervals
##
##      2 vs. 1                  gamma
## ---  -----------------------  --------------------
## 1    0.58 (0.527,0.527)       0.749 (0.723,0.723)
## 2    -0.198 (-0.265,-0.265)
## 3    0.248 (0.157,0.157)
## 4    0.276 (0.327,0.327)
## 5    0.135 (0.193,0.193)
##
##
## Table: Summary with p-values
##
##      2 vs. 1      gamma
## ---  -----------  ----------
## 1    0.58 (0)     0.749 (0)
## 2    -0.198 (0)
## 3    0.248 (0)
## 4    0.276 (0)
## 5    0.135 (0)
```



Data example 2: K=3 response categories

Here, we demonstrate the use of the package when there are 3 response categories. Essentially, everything is still the same, and most of the code doesn't change at all. The only difference is that now, the $\boldsymbol{\beta}$ is a matrix with $2 = K - 1$ columns, rather than a vector as in the $K = 2$ case.

Simulating data

Generating simulated data using the function drawSamples().

```
library(automultinomial2)
set.seed(42)

#10 predictors
p=5

#n times n grid
n=40

#make grid and adjacency matrix
latticeGraph=igraph::make_lattice(c(n,n))
A=igraph::get.adjacency(latticeGraph)

#set coefficient values
```

```r
#with 3 categories in the response, beta is now a matrix
beta=matrix(rnorm(p*2),ncol=2)*0.3
beta
```

```
##            [,1]        [,2]
## [1,]  0.4112875 -0.03183735
## [2,] -0.1694095  0.45345660
## [3,]  0.1089385 -0.02839771
## [4,]  0.1898588  0.60552711
## [5,]  0.1212805 -0.01881423
```

```r
#set covariate values
X=matrix(rnorm(n^2*p),ncol=p)

#set the correlation parameter value (0.7 is a moderate amount of spatial correlation)
gamma=0.7

#use drawSamples to simulate data with parameters beta and gamma by Gibbs sampling
y=drawSamples(beta,gamma,X,A,nSamples = 1)
```

```
## Burn-in samples
## 100  burn-in samples so far
## 200  burn-in samples so far
## 300  burn-in samples so far
## Drawing samples
```

Figure 2 shows plots of the 3-category responses on the grid. On the left plot, we again see "clumping" of the responses due to the positive autocorrelation parameter. The right hand plot is from the distribution with the same β, and $\gamma = 0$.

Fitting an automultinomial model to the data (K=3 categories)

```r
#responses must be input as a factor
y=factor(y)
fit1=automultinomial2::MPLE(X=X,y=y,A=A,ciLevel = 0.95,method = "asymptotic")
```

```
## Starting model fitting
## Model fitting done, starting variance estimation
## Creating asymptotic confidence intervals
```

```r
fit2=automultinomial2::MPLE(X=X,y=y,A=A,ciLevel = 0.95,method = "boot",nBoot = 1)
```

```
## Starting model fitting
## Model fitting done, starting variance estimation
## Creating boot confidence intervals
## Burn-in samples
## 100  burn-in samples so far
## 200  burn-in samples so far
## 300  burn-in samples so far
## Starting bootstrap sampling and fitting
```

```r
#to see the information contained in fit1 and fit2, use str() (not run to save space)
#str(fit1)
#str(fit2)

fitSummary1=MPLE_summary(fit1)
```

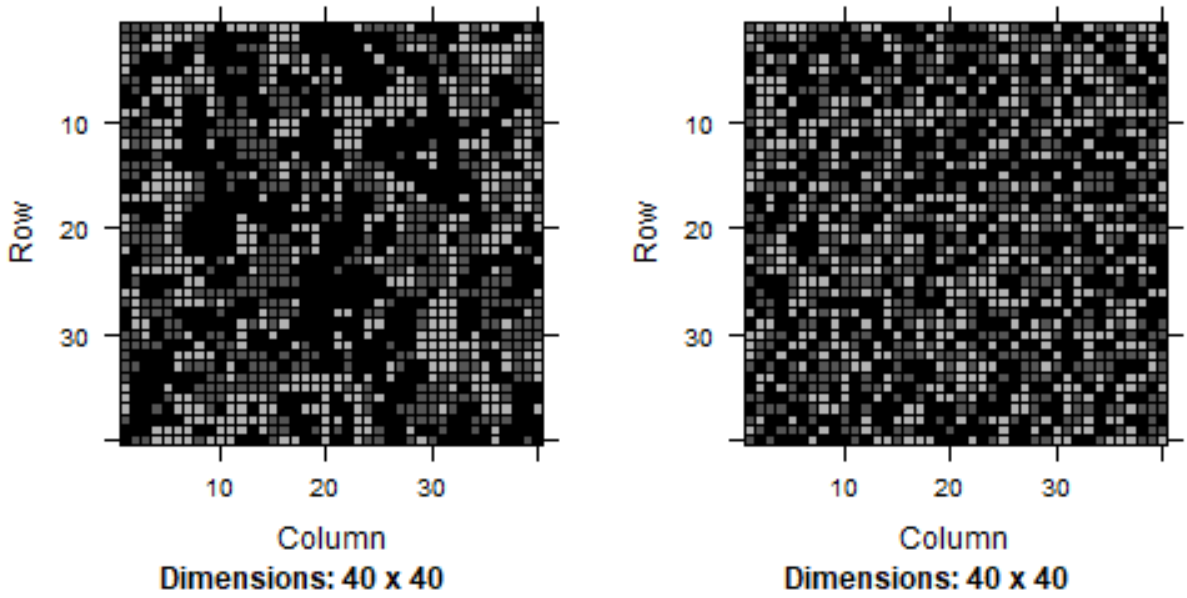


Figure 2: On the left, a plot of the 3-category dataset we just simulated, where the truth is $\gamma = 0.7$. For comparison, the plot on the right shows data taken from the same model but with $\gamma = 0$.

```
##
##
## Table: Summary with confidence intervals
##
##      2 vs. 1                  3 vs. 1                 gamma
## ---  -----------------------  ----------------------  -------------------
## 1    0.407 (0.255,0.56)       -0.039 (-0.187,0.109)   0.74 (0.672,0.808)
## 2    -0.268 (-0.409,-0.127)   0.382 (0.243,0.521)
## 3    0.129 (-0.025,0.283)     0.008 (-0.138,0.154)
## 4    0.168 (0.019,0.317)      0.632 (0.483,0.78)
## 5    0.134 (-0.019,0.286)     -0.008 (-0.149,0.133)
##
##
## Table: Summary with p-values
##
##      2 vs. 1          3 vs. 1          gamma
## ---  ---------------  ---------------  ---------
## 1    0.407 (0)        -0.039 (0.607)   0.74 (0)
## 2    -0.268 (0)       0.382 (0)
## 3    0.129 (0.101)    0.008 (0.915)
## 4    0.168 (0.027)    0.632 (0)
## 5    0.134 (0.086)    -0.008 (0.916)
```

```
fitSummary2=MPLE_summary(fit2)
```

```
##
##
## Table: Summary with confidence intervals
##
##      2 vs. 1                  3 vs. 1                gamma
## ---  -----------------------  ---------------------  -------------------
## 1    0.407 (0.418,0.418)      -0.039 (-0.02,-0.02)   0.74 (0.783,0.783)
## 2    -0.268 (-0.247,-0.247)   0.382 (0.552,0.552)
## 3    0.129 (0.162,0.162)      0.008 (0.037,0.037)
## 4    0.168 (0.161,0.161)      0.632 (0.715,0.715)
## 5    0.134 (0.124,0.124)      -0.008 (0.037,0.037)
##
##
## Table: Summary with p-values
##
##      2 vs. 1      3 vs. 1      gamma
## ---  -----------  -----------  ---------
## 1    0.407 (0)    -0.039 (0)   0.74 (0)
## 2    -0.268 (0)   0.382 (0)
## 3    0.129 (0)    0.008 (1)
## 4    0.168 (0)    0.632 (0)
## 5    0.134 (0)    -0.008 (1)
```