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
5. Some comments on autologistic model parameterization

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 still might happen that nearby sites in a dataset are 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 of a lattice 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 β with length p, and

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

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

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

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 $i \sim i'$ become 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, then we need to estimate $\boldsymbol{\beta}$ and γ jointly. Unfortunately, maximum likelihood for the model in equation (3) is computationally infeasible due to 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

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

over $\boldsymbol{\beta}$ 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 equation (3) shows that for $K = 2$,

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

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

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

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

and for $k > 1$,

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

In the automultinomial package, equations (5), (6), (7), and (8) are plugged into equation (4), and (4) is optimized over $\boldsymbol{\beta}$ and γ using the function optim.

Installing from Github

Installing from Github is 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

First we will use the drawSamples() function to simulate some data from the autologistic model. The drawSamples function takes the following arguments:

- beta: the $p \times 1$ coefficient vector (for $K = 2$ response categories) or the $p \times (K-1)$ coefficient matrix (for $K > 2$ response categories)
- gamma: the value of the correlation parameter
- X: the $n \times p$ design matrix $X = [x_1, x_2, ..., x_n]^T$
- A: a square symmetric adjacency matrix encoding the neighborhood structure
- burnIn: the number of burnIn iterations for the Gibbs sampler. Safe to leave at the default of 300.
- nSamples: the number of simulated samples to draw.

```
library(automultinomial2)
set.seed(33)

#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,] -0.04077736
## [2,] -0.01223909
## [3,]  0.30316170
## [4,] -0.04747873
## [5,] -0.64699125
```

```
#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 $\boldsymbol{\beta}$, and $\gamma = 0$.

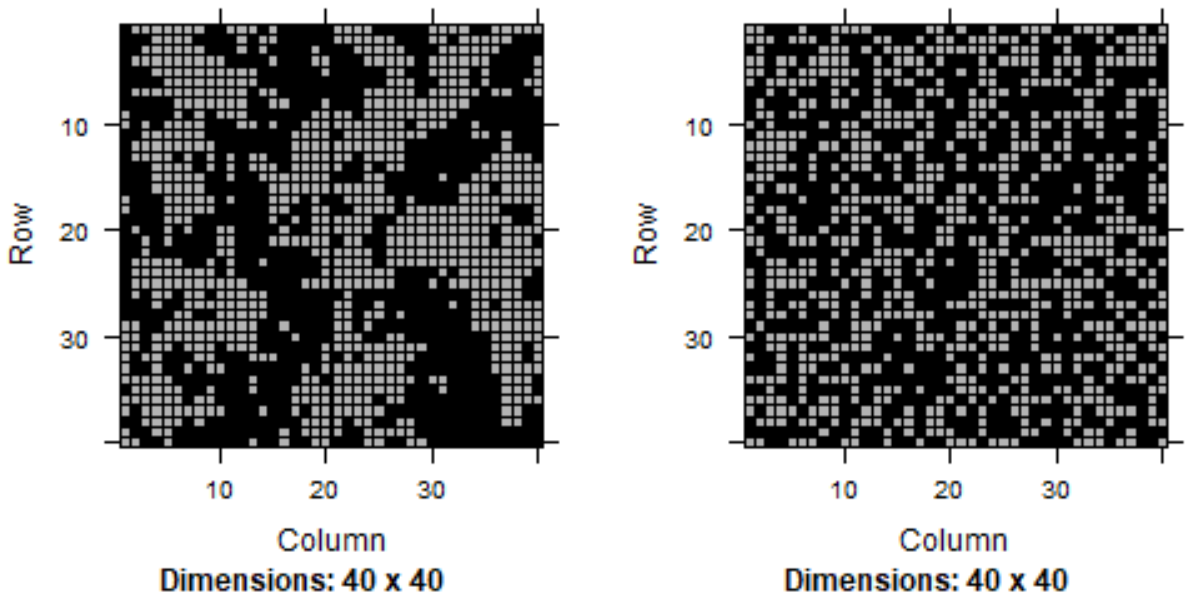


Figure 1: On the left, a plot of the dataset we just simulated, where the truth is $\gamma = 0.7$. For comparison, the plot on the right shows data taken from the same model but with $\gamma = 0$.

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

Now we'll fit an autologistic model using the MPLE function. The MPLE function takes 3 primary arguments:

- X: the design matrix $X = [x_1, x_2, ..., x_n]^T$
- A: a square symmetric adjacency matrix encoding the neighborhood structure
- y: the response vector y, which is required by MPLE to be a factor vector.

There are also several secondary arguments.

- ciLevel: for $xy\%$ confidence intervals, set ciLevel to $0.xy$ (the default is ciLevel=0.95, which produces 95% confidence intervals)
- method: By choosing method="asymptotic", MPLE will output confidence intervals based on the asymptotic distribution of the pseudolikelihood estimator. For bootstrap confidence intervals, use method="boot". The default is method="asymptotic".
- burnIn: the number of burnIn iterations for the Gibbs sampler for method="boot". Safe to leave at the default of 300.
- nBoot: the number of bootstrap samples to use for making bootstrap confidence intervals.

After fitting the model, we will examine confidence intervals for the fitted parameters. The asymptotic type confidence intervals can be computed very quickly, but may be less accurate in practice than bootstrap confidence intervals.

We can use the MPLE_summary function to view the model summary in table form.

```
# 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 = 500)
```

```
## 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
## 100  bootstrap samples so far
## 200  bootstrap samples so far
## 300  bootstrap samples so far
## 400  bootstrap samples so far
## 500  bootstrap samples so far
```

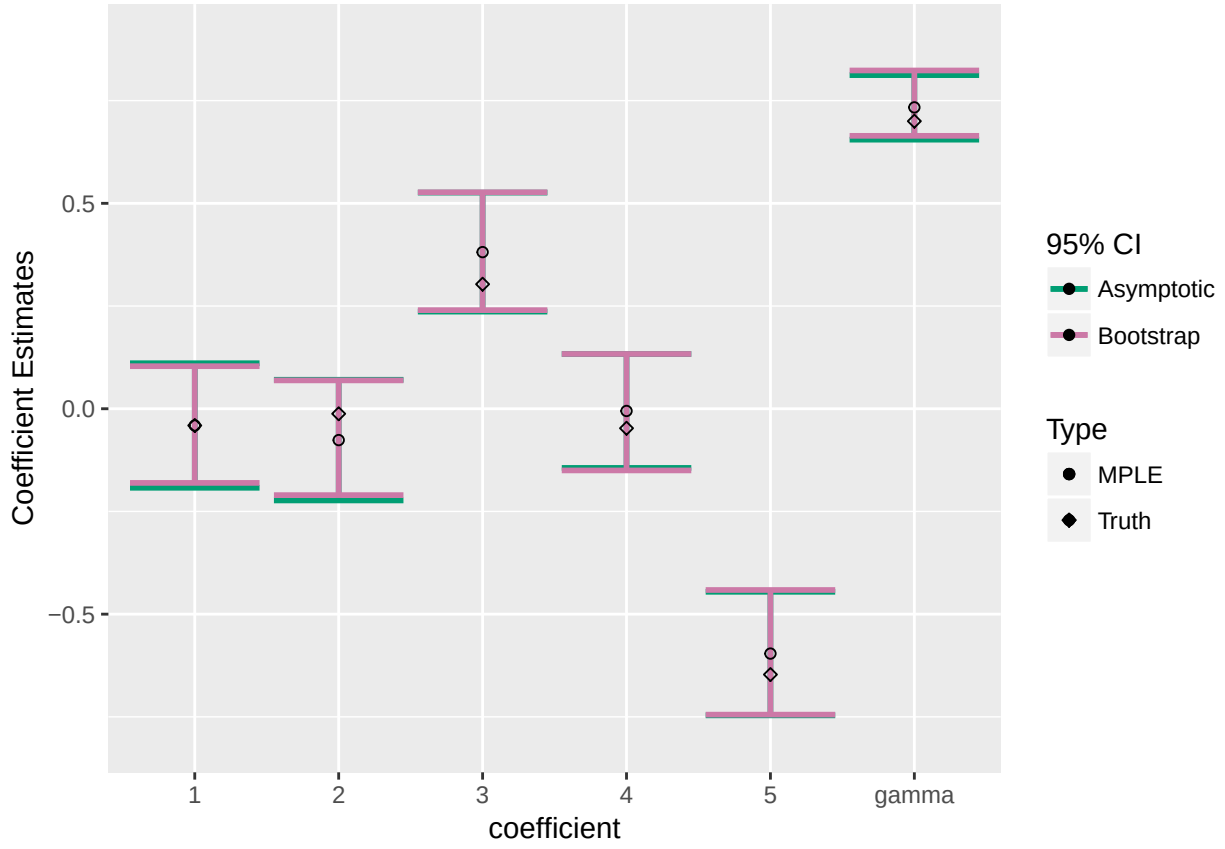
```
#to see the information contained in fit1 and fit2, use str() (not run to save space)
#str(fit1)
#str(fit2)

fitSummary1=MPLE_summary(fit1)
```

```
##
##
## Table: Summary with confidence intervals
##
##      2 vs. 1                  gamma
## ---  -----------------------  --------------------
## 1    -0.041 (-0.192,0.111)    0.733 (0.655,0.812)
## 2    -0.076 (-0.223,0.071)
## 3    0.381 (0.236,0.526)
## 4    -0.005 (-0.144,0.133)
## 5    -0.596 (-0.746,-0.446)
##
##
## Table: Summary with p-values
##
##      2 vs. 1          gamma
## ---  ---------------  ----------
## 1    -0.041 (0.598)   0.733 (0)
## 2    -0.076 (0.308)
## 3    0.381 (0)
## 4    -0.005 (0.939)
## 5    -0.596 (0)
```

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

```
##
##
## Table: Summary with confidence intervals
##
##      2 vs. 1                  gamma
```

```
## ---  ----------------------  --------------------
## 1    -0.041 (-0.181,0.104)    0.733 (0.664,0.823)
## 2    -0.076 (-0.21,0.069)
## 3    0.381 (0.24,0.527)
## 4    -0.005 (-0.15,0.133)
## 5    -0.596 (-0.745,-0.441)
##
##
## Table: Summary with p-values
##
##      2 vs. 1          gamma
## ---  ---------------  ----------
## 1    -0.041 (0.57)    0.733 (0)
## 2    -0.076 (0.292)
## 3    0.381 (0)
## 4    -0.005 (0.942)
## 5    -0.596 (0)
```



Data example 2: K=3 response categories

Now, we demonstrate the use of the package when there are 3 response categories. Essentially, everything is still the same, and most of the previous code doesn't change at all. The only difference is that now, the $\boldsymbol{\beta}$ parameter 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
#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
```

```
#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 $\boldsymbol{\beta}$, and $\gamma = 0$.

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

```
#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
```

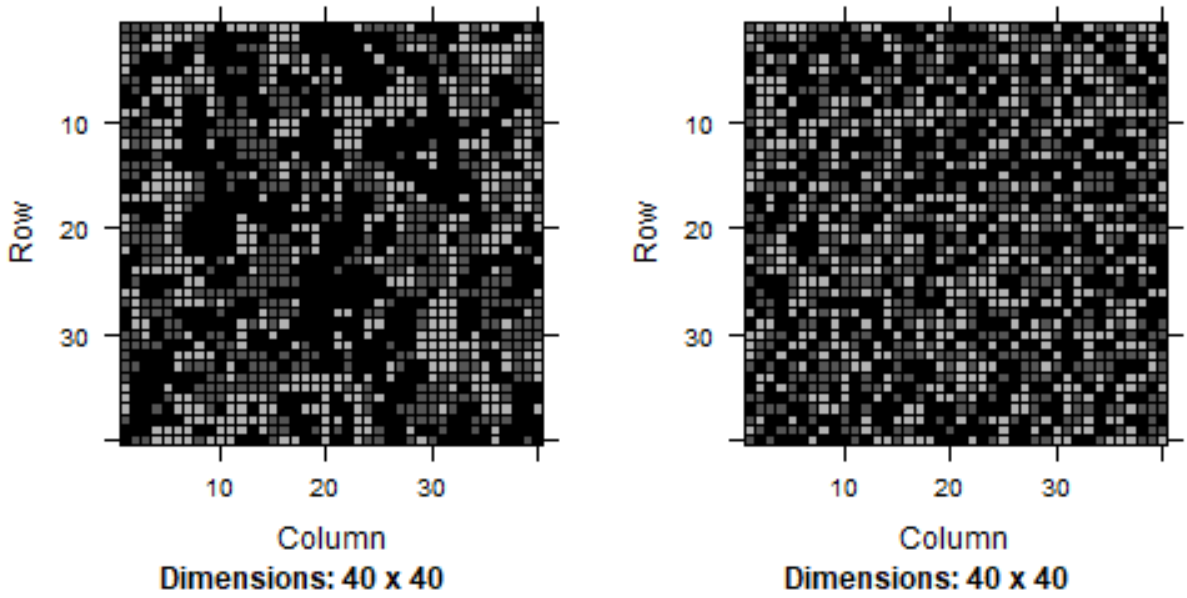


Figure 2: On the left, a plot of the 3-category dataset we just simulated, where the truth is $\gamma = 0.7$. For comparison, the plot on the right shows data taken from the same model but with $\gamma = 0$.

```
## 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 = 500)
```

```
## 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
## 100  bootstrap samples so far
## 200  bootstrap samples so far
## 300  bootstrap samples so far
## 400  bootstrap samples so far
## 500  bootstrap samples so far
```

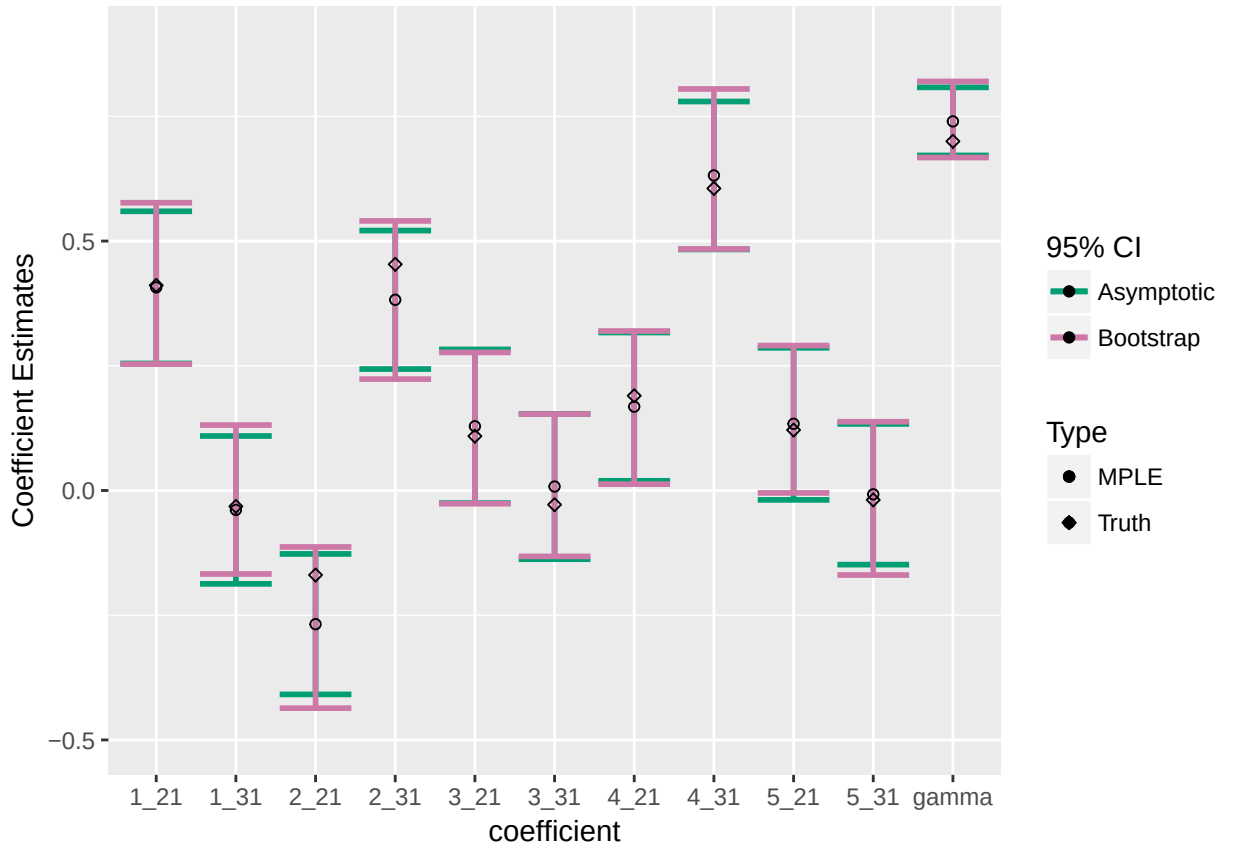
```
#to see the information contained in fit1 and fit2, use str() (not run to save space)
#str(fit1)
#str(fit2)

fitSummary1=MPLE_summary(fit1)
```

```
##
##
```

```
## Table: Summary with confidence intervals
##
##      2 vs. 1                  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 (0.253,0.577)      -0.039 (-0.167,0.131)   0.74 (0.667,0.82)
## 2    -0.268 (-0.436,-0.113)   0.382 (0.223,0.54)
## 3    0.129 (-0.027,0.277)     0.008 (-0.132,0.153)
## 4    0.168 (0.013,0.32)       0.632 (0.484,0.805)
## 5    0.134 (-0.005,0.291)     -0.008 (-0.169,0.138)
##
##
## Table: Summary with p-values
##
##      2 vs. 1         3 vs. 1          gamma
## ---  --------------  ---------------  ---------
## 1    0.407 (0)       -0.039 (0.596)   0.74 (0)
## 2    -0.268 (0)      0.382 (0)
## 3    0.129 (0.102)   0.008 (0.924)
## 4    0.168 (0.026)   0.632 (0)
## 5    0.134 (0.082)   -0.008 (0.918)
```



A comparison of different autologistic parameterizations

Finally, we'll conclude with a discussion of different parameterizations of the autologistic model. We will focus on the case $K = 2$ here. The goal for now is to show by example that the automultinomial parameterization will be invariant to the choice of reference category (not guaranteed by all of the parameterizations) and to give some concrete data examples.

Commonly used autologistic parameterizations

Frequently, autologistic parameterizations are given in terms of the conditional probability distributions at each site, rather than in terms of the joint distribution over all sites. This convention might perhaps lead to confusion about what model is actually being fit, but we will follow the common convention here.

We suppose that the response at each site is binary, so that there are $K = 2$ response categories.

Parameterization 1 (Asymmetric): In what we will call the asymmetric parameterization, the set of allowed responses at each site is usually taken to be $\{0, 1\}$. The conditional probabilities for the set $\{0, 1\}$ are

$$p(z_i = 1|z_{-i}) = \frac{\exp\{x_i^T\beta + \gamma \sum_{i' \sim i} z_{i'}\}}{1 + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} z_{i'}\}} \tag{9}$$

$$p(z_i = 0|z_{-i}) = \frac{1}{1 + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} z_{i'}\}} \tag{10}$$

If we shift the values of the responses up by 1, so that we identify 0 with 1 and 1 with 2, then with a slight abuse of notation we have

$$p(z_i = 2|z_{-i}) = \frac{\exp\{x_i^T\beta + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}}{1 + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}} \tag{11}$$

$$p(z_i = 1|z_{-i}) = \frac{1}{1 + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}} \tag{12}$$

This form of the conditional distributions corresponds to the model

$$p_{asy}(\mathbf{z}|\boldsymbol{\beta}, \gamma) = \frac{\exp\{H_{asy}(\mathbf{z}|\boldsymbol{\beta}, \gamma)\}}{\sum_{\mathbf{z}'} \exp\{H_{asy}(\mathbf{z}'|\boldsymbol{\beta}, \gamma)\}} \tag{13}$$

where

$$H_{asy}(\mathbf{z}|\boldsymbol{\beta}, \gamma) = \sum_{i=1}^{n} x_i^T \beta I(z_i = 2) + \gamma \sum_{i=1}^{n} \sum_{\substack{i' \sim i \\ i' > i}} I(z_i = z_{i'} = 2) \tag{14}$$

By the Hammersley-Clifford theorem, the density in equation (13) is the unique joint distribution corresponding to the conditional distributions in equation (12).

Parameterization 2 (Symmetric): This parameterization is sometimes referred to as the ± 1 parameterization. We will now show that for $K = 2$, the symmetric parameterization is equivalent to the parameterization used by automultinomial. On the other hand, the formulation given for the automultinomial parameterization in section 1 is much more cleanly generalized to $K > 2$ categories.

In the symmetric parameterization, the set of allowed outcomes is taken to be $\{-1, 1\}$, and the conditional probabilities at each site are

$$p(z_i = 1|z_{-i}) = \frac{\exp\{x_i^T\beta + \gamma \sum_{i' \sim i} z_{i'}\}}{1 + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} z_{i'}\}} \tag{15}$$

Now, as with the asymmetric model, we will map the response set $\{-1, 1\}$ to $\{1, 2\}$. We will identify -1 with 1 and 1 with 2, so that the conditional probabilities become

$$\begin{aligned} p(z_i = 2|z_{-i}) &= \frac{\exp\left[x_i^T\beta + \gamma \sum_{i' \sim i} \{I(z_{i'} = 2) - I(z_{i'} = 1)\}\right]}{1 + \exp\left[x_i^T\beta + \gamma \sum_{i' \sim i} \{I(z_{i'} = 2) - I(z_{i'} = 1)\}\right]} & (16) \\ &= \frac{\exp\{x_i^T\beta + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}}{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\} + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}} & (17) \\ p(z_i = 1|z_{-i}) &= \frac{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\}}{\exp\{\gamma \sum_{i' \sim i} I(z_{i'} = 1)\} + \exp\{x_i^T\beta + \gamma \sum_{i' \sim i} I(z_{i'} = 2)\}} & (18) \end{aligned}$$

This form of the conditional distributions corresponds to the model

$$p_{sym}(\mathbf{z}|\boldsymbol{\beta}, \gamma) = \frac{\exp\{H_{sym}(\mathbf{z}|\boldsymbol{\beta}, \gamma)\}}{\sum_{\mathbf{z}'} \exp\{H_{sym}(\mathbf{z}'|\boldsymbol{\beta}, \gamma)\}} \tag{19}$$

$$H_{sym}(\mathbf{z}|\boldsymbol{\beta},\gamma) = \sum_{i=1}^{n} x_i^T \beta I(z_i = 2) + \gamma \sum_{i=1}^{n} \sum_{\substack{i' \sim i \\ i' > i}} \sum_{k=1}^{2} I(z_i = z_{i'} = k) \tag{20}$$

By the Hammersley-Clifford theorem, the density in equation (19) is the unique joint distribution corresponding to the conditional distributions in equation (18).

Parameterization 3: the parameterization used in automultinomial: This parameterization is the parameterization defined in section 2 of this document.

Comparison

Having defined the 3 parameterizations, we will now compare them. Our first conclusion is that when $K = 2$, the symmetric (± 1) parameterization is *exactly* equivalent to the automultinomial parameterization. This conclusion follows from determining that the energies in equations (1) and (20) are exactly the same, so that the densities in equation (19) and (3) are exactly the same.

Therefore, from now on we need only compare the automultinomial parameterization and the asymmetric parameterization. Our conclusions are the following:

- When each site has the same number of neighbors, and the covariate matrix $X = [x_1, x_2, ..., x_n]^T$ includes an intercept column, then the asymmetric and the automultinomial parameterizations are equivalent, in the sense that any joint distribution in the automultinomial parameterization has an equivalent representation in the asymmetric parameterization. The pseudolikelihoods obtained by fitting the automultinomial and the asymmetric parameterizations to a dataset will be identical. The coefficients from the automultinomial parameterization can be obtained from the asymmetric parameterization, and vice versa.
- When the number of neighbors at each site is different, and/or the covariate matrix does not include an intercept column, then the asymmetric and the automultinomial parameterizations will in general *not* be equivalent. The automultinomial parameterization is invariant to the choice of reference category, whereas the asymmetric parameterization will in general not be invariant to the choice of reference category.

We will now give some examples on simulated data.

```
#making a square lattice graph and adjacency matrix with toroidal boundary conditions
#every site has 4 neighbors
t1=igraph::make_lattice(c(40,40),circular=TRUE)
a1=igraph::get.adjacency(t1)


#making a square lattice graph and adjacency matrix with free boundary conditions
#sites have 2 neighbors at the corner of the lattice, 3 neighbors on edges of the lattice,
#and 4 neighbors internal to the lattice
t2=igraph::make_lattice(c(40,40),circular=FALSE)
a2=igraph::get.adjacency(t2)

#making the X matrices: X does not have an intercept, but X_intercept has an intercept
X=matrix(rnorm(1600*2),ncol=2)
X_intercept=cbind(rep(1,1600),X)

beta=rnorm(3)
gamma=0.5
```

```
#random responses
y=automultinomial2::drawSamples(beta=beta,gamma=gamma,X = X_intercept, A=a1,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
```

Case 1: same number of neighbors, covariate matrix includes intercept

Here, we see that the pseudolikelihoods match and that coefficients (excepting γ and the intercept) also match. The intercept and γ from the two models are related by a simple linear transformation.

```
fitAsymmetric=glm(factor(y)~cbind(X,as.matrix(a1%*%(y-1))),family=binomial())
fitAutomultinomial=automultinomial2::MPLE(X=X_intercept,y=factor(y),A = a1,method="asymptotic")
```

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

```
#comparing pseudolikelihoods: identical
fitAsymmetric$deviance/-2
```

```
## [1] -325.2283
```

```
fitAutomultinomial$pseudolikelihood
```

```
## [1] -325.2283
```

```
#comparing beta: identical
fitAsymmetric$coefficients[2:3]
```

```
## cbind(X, as.matrix(a1 %*% (y - 1)))1 cbind(X, as.matrix(a1 %*% (y - 1)))2
##                           -0.2279777                           -0.4658640
```

```
fitAutomultinomial$betaHat[2:3]
```

```
## [1] -0.2281012 -0.4659530
```

```
#comparing intercept and gamma: not identical, but there is a mapping between them
gammaAsymmetric=fitAsymmetric$coefficients[4]
gammaAutomultinomial=fitAutomultinomial$gammaHat
gammaAsymmetric
```

```
## cbind(X, as.matrix(a1 %*% (y - 1)))3
##                            1.153929
```

```
gammaAutomultinomial*2
```

```
## [1] 1.153822
```

```
interceptAutomultinomial=fitAutomultinomial$betaHat[1]
interceptAsymmetric=fitAsymmetric$coefficients[1]
interceptAsymmetric
```

```
## (Intercept)
##   -1.292221
```

```
interceptAutomultinomial-4*gammaAutomultinomial
```

```
## [1] -1.291825
```

Case 2: different number of neighbors, covariate matrix includes intercept

Here, we see that the pseudolikelihoods and coefficients no longer match between parameterizations. Additionally, while the automultinomial parameterization is invariant to changes in the reference category, the asymmetric parameterization is not.

```
fitAsymmetric1=glm(factor(y)~cbind(X,as.matrix(a2%*%(y-1))),family=binomial())
fitAutomultinomial1=automultinomial2::MPLE(X=X_intercept,y=factor(y),A = a2,method="asymptotic")
```

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

```
#refitting with different reference category
y2=as.factor(y)
y2=as.numeric(relevel(y2,ref = "2"))
fitAsymmetric2=glm(factor(y2)~cbind(X,as.matrix(a2%*%(y2-1))),family=binomial())
fitAutomultinomial2=automultinomial2::MPLE(X=X_intercept,y=factor(y2),A = a2,method="asymptotic")
```

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

```
#comparing pseudolikelihoods: not identical
fitAsymmetric1$deviance/-2
```

```
## [1] -330.8151
```

```
fitAsymmetric2$deviance/-2
```

```
## [1] -324.5341
```

```
fitAutomultinomial1$pseudolikelihood
```

```
## [1] -326.5898
```

```
fitAutomultinomial2$pseudolikelihood
```

```
## [1] -326.5898
```

```
#comparing beta: not identical
fitAsymmetric1$coefficients[2:3]
```

```
## cbind(X, as.matrix(a2 %*% (y - 1)))1 cbind(X, as.matrix(a2 %*% (y - 1)))2
##                           -0.2244218                           -0.4480677
```

```
fitAsymmetric2$coefficients[2:3]
```

```
## cbind(X, as.matrix(a2 %*% (y2 - 1)))1
##                            0.2278134
## cbind(X, as.matrix(a2 %*% (y2 - 1)))2
##                            0.4630238
```

```
fitAutomultinomial1$betaHat[2:3]
```

```
## [1] -0.2247864 -0.4590982
```

```
fitAutomultinomial2$betaHat[2:3]
```

```
## [1] 0.2247864 0.4590982
```

```
#comparing gamma: not identical. Asymmetric parameterization not invariant to
#choice of reference category. Automultinomial parameterization invariant to
```

```r
#choice of reference category
gammaAsymmetric1=fitAsymmetric1$coefficients[4]
gammaAsymmetric2=fitAsymmetric2$coefficients[4]
gammaAutomultinomial1=fitAutomultinomial1$gammaHat
gammaAutomultinomial2=fitAutomultinomial2$gammaHat
gammaAsymmetric1
```

```
## cbind(X, as.matrix(a2 %*% (y - 1)))3
##                            0.8653329
```

```r
gammaAsymmetric2
```

```
## cbind(X, as.matrix(a2 %*% (y2 - 1)))3
##                               1.18745
```

```r
gammaAutomultinomial1
```

```
## [1] 0.5553927
```

```r
gammaAutomultinomial2
```

```
## [1] 0.5553927
```

Case 2: same number of neighbors, covariate matrix does not include intercept

Here, we see that the pseudolikelihoods and coefficients no longer match between parameterizations. Additionally, while the automultinomial parameterization is invariant to changes in the reference category, the asymmetric parameterization is not.

```r
fitAsymmetric1=glm(factor(y)~cbind(X,as.matrix(a2%*%(y-1)))-1,family=binomial())
fitAutomultinomial1=automultinomial2::MPLE(X=X,y=factor(y),A = a2,method="asymptotic")
```

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

```r
#refitting with different reference category
y2=as.factor(y)
y2=as.numeric(relevel(y2,ref = "2"))
fitAsymmetric2=glm(factor(y2)~cbind(X,as.matrix(a2%*%(y2-1)))-1,family=binomial())
fitAutomultinomial2=automultinomial2::MPLE(X=X,y=factor(y2),A = a2,method="asymptotic")
```

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

```r
#comparing pseudolikelihoods: not identical
fitAsymmetric1$deviance/-2
```

```
## [1] -330.8517
```

```r
fitAsymmetric2$deviance/-2
```

```
## [1] -1008.83
```

```r
fitAutomultinomial1$pseudolikelihood
```

```
## [1] -334.6936
```

```r
fitAutomultinomial2$pseudolikelihood
```

```
## [1] -334.6936
```

```r
#comparing beta: not identical
fitAsymmetric1$coefficients[1:2]
```

```
## cbind(X, as.matrix(a2 %*% (y - 1)))1 cbind(X, as.matrix(a2 %*% (y - 1)))2
##                           -0.2257393                           -0.4477906
```

```r
fitAsymmetric2$coefficients[1:2]
```

```
## cbind(X, as.matrix(a2 %*% (y2 - 1)))1
##                            0.02324860
## cbind(X, as.matrix(a2 %*% (y2 - 1)))2
##                            0.03277904
```

```r
fitAutomultinomial1$betaHat[1:2]
```

```
## [1] -0.1933032 -0.4484902
```

```r
fitAutomultinomial2$betaHat[1:2]
```

```
## [1] 0.1933032 0.4484902
```

```r
#comparing gamma: not identical. Asymmetric parameterization not invariant to
#choice of reference category. Automultinomial parameterization invariant to
#choice of reference category
gammaAsymmetric1=fitAsymmetric1$coefficients[3]
gammaAsymmetric2=fitAsymmetric2$coefficients[3]
gammaAutomultinomial1=fitAutomultinomial1$gammaHat
gammaAutomultinomial2=fitAutomultinomial2$gammaHat
gammaAsymmetric1
```

```
## cbind(X, as.matrix(a2 %*% (y - 1)))3
##                            0.8198363
```

```r
gammaAsymmetric2
```

```
## cbind(X, as.matrix(a2 %*% (y2 - 1)))3
##                             -1.760343
```

```r
gammaAutomultinomial1
```

```
## [1] 0.8948406
```

```r
gammaAutomultinomial2
```

```
## [1] 0.8948406
```