# Package 'copCAR'

September 7, 2014

**Version** 1.0

**Date** 2014

**Title** Fitting the copCAR regression model for discrete areal data

**Type** Package

**Author** Emily Goren <emily.goren@gmail.com> and John Hughes <hughesj@umn.edu>

**Maintainer** John Hughes <hughesj@umn.edu>

**Depends** numDeriv, Rcpp, spam

**Suggests** lattice

**LinkingTo** Rcpp, RcppArmadillo

**RcppModules** buildM, inverse

**Description** copCAR provides tools for fitting the copCAR regression model for discrete areal data. Three types of estimation are supported: continuous extension, composite marginal likelihood, and distributional transform.

**License** GPL (>= 2)

**URL** http://www.biostat.umn.edu/~johnh

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2014-09-07 08:33:10

## R topics documented:

## adjacency.matrix *Create adjacency matrix.*

### Description

Create adjacency matrix.

### Usage

```
adjacency.matrix(m, n = NULL)
```

### Arguments

| | |
|---|---|
| m | the number of rows in the lattice. |
| n | the number of columns in the lattice. Defaults to NULL. If missing, the lattice is assumed to be m by m. |

### Details

This function builds the adjacency matrix for a m by n square lattice.

### Value

A matrix $A$ of 0s and 1s, where $(A_{ij})$ is equal to 1 if and only if vertices $i$ and $j$ are adjacent.

## copCAR *Fit copCAR model to discrete areal data.*

### Description

Fit the copCAR model to areal data consisting of Poisson or Bernoulli marginal observations.

### Usage

```
copCAR(formula, A, family, method = c("CML", "DT", "CE"),
  conf.int = c("none", "bootstrap", "asymptotic"), data, offset = NULL,
  control = list())
```

## Arguments

| | |
|---|---|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of the model specification are given under "Details". |
| A | the symmetric binary adjacency matrix for the underlying graph. adjacency.matrix generates an adjacency matrix for the $m$ by $n$ square lattice. |
| family | the marginal distribution of the observations at the areal units and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) Supported families are binomial and poisson. |
| method | the method for inference. copCAR supports the continous extension ("CE"), distributional transform ("DT"), and composite marginal likelihood ("CML"). |
| conf.int | the method for computing confidence intervals. "asympototic" is appropriate for the continous extension. "bootstrap" performs a parametric boostrap appropriate for the distributional transform and composite marginal likelihood. |
| data | an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which copCAR is called. |
| offset | this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of observations. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset. |
| control | a list of parameters for controlling the fitting process. |

- conf.level the value $1-\alpha$ used for computing confidence intervals. Defaults to 0.95.
- boot.iter the size of the parametric bootstrap sample. This applies when conf.int = "bootstrap". Defaults to 500.
- m the number of independent standard uniforms used to approximate the expected likelhood when method = "CE". This applies when method = "CE". Defaults to 1000.
- mcse logical. Should the Monte Carlo standard error for $\rho$ be computed? Use only when method = "CE". Defaults to FALSE. The Monte Carlo standard error is calculated using a sample size of mcse.iter.
- mcse.iter the Monte Carlo standard error sample size for $\rho$ when method = "CE" and mcse = TRUE. Defaults to 100.
- rhoMax the value $\rho^{\max}$, which is the maximum value of $\rho$ used to approximate the CAR variances when method = "CE" or method = "DT". If missing, assumed to be 0.999.
- epsilon the tolerance $\epsilon > 0$ used to approximate the CAR variances when method = "CE" or method = "DT". If missing, assumed to be 0.001.
- verbose a logical value indicating whether to print bootstrap or mcse progress to the screen. Defaults to FALSE.

## Details

This function performs frequentist inference for the copCAR model of Hughes (2014), a copula-based areal regression model that uses the conditional autoregression (CAR) from the spatial generalized linear mixed model (Besag, 1974). Specifically, copCAR uses the CAR copula, a Caussian copula based on the proper CAR. The CAR copula is specified as

$$C_{Q^{-1}}(u) = \Phi_{Q^{-1}}(\Phi_{\sigma_1}^{-1}(u_1), \ldots, \Phi_{\sigma_n}^{-1}(u_n)),$$

where $\Phi_{\sigma_i}$ denotes the cdf of the normal distribution with mean zero and variance $\sigma_i^2$, $Q = D - \rho A$ such that $\tau Q$ is the precision matrix of the proper CAR, $A$ is the adjacency matrix for the underlying graph, $D = diag(d_1, \ldots, d_n)$ where $d_i$ is the degree of vertex $i$ of the underlying graph, and $u = (u_1, \ldots, u_n)'$ is a realization of the copula such that $z_i = F_i^{-1}(u_i)$ for the marginal observation $z_i$ having desired marginal distribution function $F_i$. For Bernoulli marginals, the expectation is $(1 + \exp(-x_i'\beta))^{-1}$; for Poisson marginals, the expectation is $\exp(x_i'\beta)$, where $\beta = (\beta_1, \ldots, \beta_p)'$ is the regression coefficient. Note that the CAR variances $(\sigma_1^2, \ldots, \sigma_n^2)' = vecdiag(Q^{-1})$ are not free parameters but are determined by the spatial dependence parameter $\rho$.

The spatial dependence parameter $\rho \in [0, 1)$ and regression coefficient $\beta = (\beta_1, \ldots, \beta_p)' \in R^p$ can be estimated using the continous extension (CE) (Madsen, 2009), distribtional transform (DT) (Kazianka and Pilz, 2010), or composite marginal likelihood (CML) (Varin, 2008).

The CE approach optimizes an approximate maximum likelihood by sampling $m$ independent standard uniform vectors of length $n$ used to transform the discrete observations into continous random variables via convolution (Denuit and Lambert, 2005). The size of $m$ can be choosen by computing Monte Carlo standard errors (Flegal et al., 2008). If the Monte Carlo standard error of the estimate for $\rho$ is small relative to the sample mean, that is, if the estimated coefficient of variation is sufficiently small, the current value of $m$ is sufficiently large. The CE is exact up to Monte Carlo standard error, but is computationally intensive and not suitable for Bernoulli marginals. If requested, asymptotic confidence intervals for the parameters are computed using the observed inverse Fisher information.

The DT stochastically "smoothes" the jumps of the discrete distribution function, an approach that goes at least as far back as Ferguson (1967). The DT-based approximation performs well for Poisson marginals. Since the log-likelihood is misspecified, the asympototic covariance matrix is the Godambe information matrix (Godambe, 1960). This is estimated using a parametric bootstrap for the variance of the score when computing confidence intervals for the parameters.

The CML approach specifies the likelihood as a product of pairwise likelihoods of adjacent observations, and performs well for both Poisson and Bernoulli data. Similar to the DT, the log-likelihood is misspecified, so the confidence intervals for the parameters are computed via a parametric bootstrap.

In the CE and DT approaches, the CAR variances are approximated by $(\tilde{\sigma}_1^2, \ldots, \tilde{\sigma}_n^2)'$ such that $(\sigma_i^2 - \tilde{\sigma}_i^2) < \epsilon$ for every $i = 1, \ldots, n$ for a specified tolerance $\epsilon > 0$ and every $\rho \in [0, \rho^{\max})$.

## Value

`copCAR` returns an object of S3 class `"copCAR"`, which is a list containing the following components:

`coefficients` the point estimate of $(\rho, \beta')'$.

`conf.int` (if `conf.int` is not `"none"`) the confidence intervals for $(\rho, \beta')'$.

`conf.type` the type of confidence interval specified.

`conf.level` (if `conf.int` is not `"none"`) the confidence level, $1 - \alpha$.

`mcse` (if `method = "CE"` and `mcse = TRUE`) the Monte Carlo standard error of $\rho$.

`mcse.iter` (if `method = "CE"` and `mcse = TRUE`) the Monte Carlo standard error sample size.

`mcse.cv` (if `method = "CE"` and `mcse = TRUE`) the estimated coefficient of variation of $\rho$.

`I.inv` (if `conf.int = "asymptotic"`) the estimated inverse observed Fisher information matrix (hessian) for $(\Phi^{-1}(\rho), \beta')'$.

`G.inv` (if `conf.int = "bootstrap"`) the estimated inverse Godambe information matrix (sandwich) for $(\Phi^{-1}(\rho), \beta')'$.

`se` (if `conf.int = "bootstrap"` or `conf.int = "asymptotic"`) the estimated standard errors for $(\Phi^{-1}(\rho), \beta')'$.

`boot.iter` (if `conf.int = "bootstrap"`) the number of parametric bootstrap samples.

`Z` the response vector used.

`X` the design matrix.

`model` the model frame.

`npar` the number of model parameters.

`marginal.linear.predictors`
linear predictors for the margins.

`marginal.fitted.values`
fitted values for the margins.

`call` the matched call.

`formula` the formula supplied.

`method` the method used for inference.

`convergence` the integer code returned by `optim` subsequent to optimizing the log-likelihood.

`message` a character string to go along with `convergence`.

`terms` the `terms` object used.

`data` the `data` argument.

`xlevels` (where relevant) a record of the levels of the factors used in fitting.

`control` a list containing the names and values of the control parameters.

`value` the value of the negative log-likelihood at its minimum.

## References

Besag, J. (1974) Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Soceity, Series B, Methodological*, 36(2), 192–236.

Denuit, M. and Lambert, P. (2005) Constraints on concordance measures in bivariate discrete data. *Journal of Multivariate Analysis*, 93, 40–57.

Ferguson, T. (1967) *Mathematical statistics: a decision theoretic approach*, New York: Academic Press.

Flegal, J., Haran, M., and Jones, G. (2008) Markov Chain Monte Carlo: can we trust the third significant figure? *Statistical Science*, 23(2), 250–260.

Godambe, V. (1960) An optimum property of regular maximum likelihood estimation. *The Annals of Mathmatical Statistics*, 31(4), 1208–1211.

Kazianka, H. and Pilz, J. (2010) Copula-based geostatistical modeling of continuous and discrete data including covariates. *Stochastic Environmental Research and Risk Assessment*, 24(5), 661–673.

Madsen, L. (2009) Maximum likelihood estimation of regression parameters with spatially dependent discrete data. *Journal of Agricultural, Biological, and Environmental Statistics*, 14(4), 375–391.

Varin, C. (2008) On composite marginal likelihoods. *Advances in Statistical Analysis*, 92(1), 1–28.

## Examples

```
## Not run:
# Simulate data and fit copCAR model.

# Use the 20 x 20 square lattice as the underlying graph.
m = 20
A = adjacency.matrix(m)

# Set dependence parameter and regression coefficients.
rho = 0.8
beta = c(1, 1)

# Create design matrix by assigning coordinates to each vertex
# such that the coordinates are restricted to the unit square.
x = rep(0:(m - 1) / (m - 1), times = m)
y = rep(0:(m - 1) / (m - 1), each = m)
X = cbind(x, y)

# Draw Poisson data from copCAR model.
Z = rcopCAR(rho, beta, X, A, family = poisson(link = "log"))

# Fit the copCAR model using the continous extension and
# compute 95% (default) aysmptotic CI for rho and beta.
fit.CE = copCAR(Z ~ X - 1, A, family = poisson, method = "CE", conf.int = "asymptotic")
summary(fit.CE)

# Fit the copCAR model using the distributional transform and
# compute 90% CI for rho and beta using 100 bootstrap iterations.
```

```
fit.DT = copCAR(Z ~ X - 1, A, family = poisson, method = "DT", conf.int = "bootstrap",
                control = list(conf.level = 0.90, boot.iter = 100))
summary(fit.DT)

# Fit the copCAR model using composite marginal likelihood and
# do not compute a CI for rho and beta.
fit.CML = copCAR(Z ~ X - 1, A, family = poisson, method = "CML", conf.int = "none")
summary(fit.CML)

## End(Not run)
```

---

rcopCAR *Simulate areal data.*

---

## Description

rcopCAR simulates areal data from the copCAR model.

## Usage

```
rcopCAR(rho, beta, X, A, family)
```

## Arguments

| | |
|---|---|
| rho | the spatial dependence parameter $\rho$ such that $\rho \in [0, 1)$. |
| beta | the vector of regression coefficients $\beta = (\beta_1, \ldots, \beta_p)'$. |
| X | the $n$ by $p$ design matrix $X$. |
| A | the symmetric binary adjacency matrix for the underlying graph. adjacency.matrix generates an adjacency matrix for the $m$ by $n$ square lattice. |
| family | the marginal distribution of the observations and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) Supported familes are binomial and poisson. |

## Details

This function randomly generates Poisson or Bernoulli areal data with adjacency matrix $A$ from the copCAR model with the given spatial dependence parameter $\rho$, regression coefficients $\beta = (\beta_1, \ldots, \beta_p)'$, and design matrix $X$. For more details on the copCAR model, see copCAR.

## Value

A vector of length $n$ distributed according to the copCAR model with the given design matrix and parameter values.

## Examples

```
## Not run:
require(lattice)

# Use the 20 x 20 square lattice as the underlying graph.
m = 20
A = adjacency.matrix(m)

# Set dependence parameter and regression coefficients.
rho = 0.8
beta = c(1, 1)

# Create design matrix by assigning coordinates to each vertex
# such that the coordinates are restricted to the unit square.
x = rep(0:(m - 1) / (m - 1), times = m)
y = rep(0:(m - 1) / (m - 1), each = m)
X = cbind(x, y)

# Draw Poisson data from copCAR model.
Z.pois = rcopCAR(rho, beta, X, A, family = poisson(link = "log"))

# Create a level plot of the simulated data.
dev.new()
levelplot(Z ~ x * y, aspect = "iso")

# Draw Bernoulli data from copCAR model.
Z.ber = rcopCAR(rho, beta, X, A, family = binomial(link = "logit"))

# Create a level plot of the simulated data.
dev.new()
levelplot(Z2 ~ x * y, aspect = "iso")

## End(Not run)
```

---

`summary.copCAR` *Print a summary of a copCAR model fit.*

---

## Description

Print a summary of a copCAR model fit.

## Usage

```
## S3 method for class 'copCAR'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of class `copCAR`, the result of a call to `copCAR`. |
| `...` | additional arguments. |

## Details

This function displays (1) the call to copCAR, (2) the values of the control parameters, (3) a table of estimates, and (4) confidence intervals.

Each row of the table of estimates shows a parameter estimate, the confidence interval for the parameter, and, where applicable, the Monte Carlo standard error.

## See Also

copCAR

---

`vcov.copCAR` *Return the covariance matrix of the parameters of a* `copCAR` *model object.*

---

## Description

Return the covariance matrix of the parameters of a `copCAR` model object.

## Usage

```
## S3 method for class 'copCAR'
vcov(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | a fitted `copCAR` model object. |
| `...` | additional arguments. |

## Value

An estimate of the covariance matrix of $(\Phi^{-1}(\rho), \beta')'$.

## See Also

copCAR

# Index