Package 'pogit'

July 23, 2025

Type Package

Title Bayesian Variable Selection for a Poisson-Logistic Model

Version 1.3.0

Date 2022-05-24

Author Michaela Dvorzak [aut, cre],
Helga Wagner [aut]

Maintainer Michaela Dvorzak <m.dvorzak@gmx.at>

Description Bayesian variable selection for regression models of under-reported count data as well as for (overdispersed) Poisson, negative binomal and binomial logit regression models using spike and slab priors.

License GPL-2

Encoding UTF-8

Depends R(>= 2.10.0)

Imports ggplot2, logistf, plyr, stats, utils, grDevices

LazyData true

Suggests COUNT

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-25 15:50:02 UTC

Contents

`pogit-package` *Bayesian variable selection for a Poisson-Logistic model*

Description

This package provides Bayesian variable selection for regression models of under-reported count data as well as for (overdispersed) Poisson, negative binomial and binomial logit regression models using spike and slab priors. For posterior inference, MCMC sampling schemes are used that rely on data augmentation and/or auxiliary mixture sampling techniques. Details can be found in Dvorzak and Wagner (2016).

Details

The main function is `pogitBvs` which provides Bayesian variable selection for a Poisson-Logistic (Pogit) model to account for potential under-reporting of count data. The Pogit model, introduced by Winkelmann and Zimmermann (1993), is specified by combining a Poisson model for the data generating process of counts and a logit model for the fallible reporting process, where the outcomes of both processes may depend on a set of potential covariates. By augmenting the observed data with the unobserved counts, the model can be factorized into a Poisson and a binomial logit model part. Hence, the MCMC sampling algorithm for this two-part model is based on data augmentation and sampling schemes for a Poisson and a binomial logit model.

Though part of the main function, the functions `poissonBvs` and `logitBvs` can be used separately to perform Bayesian variable selection for Poisson or binomial logit regression models. An alternative to `poissonBvs` is provided by the function `negbinBvs` to deal with overdispersion of count data. The sampling algorithms are based on auxiliary mixture sampling techniques.

All functions return an object of class "`pogit`" with methods `print.pogit`, `summary.pogit` and `plot.pogit` to summarize and display the results.

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>, Helga Wagner

Maintainer: Michaela Dvorzak <m.dvorzak@gmx.at>

References

Dvorzak, M. and Wagner, H. (2016). Sparse Bayesian modelling of underreported count data. *Statistical Modelling*, **16**(1), 24 - 46, doi:10.1177/1471082x15588398.

Winkelmann, R. and Zimmermann, K. F. (1993). Poisson-Logistic regression. Department of Economics, University of Munich, Working Paper No. 93 - 18.

See Also

pogitBvs, logitBvs, poissonBvs, negbinBvs

Examples

```
## see examples for pogitBvs, logitBvs, poissonBvs and negbinBvs
```

---

cervical *Cervical cancer data*

Description

The data set contains the number of cervical cancer deaths (ICD 180) and woman-years at risk for four age groups in four different European countries during 1969-1973.

Usage

```
data(cervical)
```

Format

A data frame with 16 rows and 19 variables:

`y` number of cervical cancer deaths for different age categories and European countries between 1969-1973

`E` number of woman-years at risk (given in thousands)

`country` factor variable of European countries

`agegroup` factor variable of age categories

`X.1`, `X.2`, `X.3` predictor variables for country effects using dummy coding (i.e. England, France, Italy)

`X.4`, `X.5`, `X.6` predictor variables for age effects using dummy coding (i.e. 35-44, 45-54, 55-64, in years)

`X.7`, `X.8`, `X.9`, `X.10`, `X.11`, `X.12`, `X.13`, `X.14`, `X.15` predictor variables for interaction effects of age and country

Note

The lowest age category (25-34) in Belgium is used as the reference category.

Source

World Health Organization (1976). World Health Statistics Annual: 1969-1976, Vol. I, *Vital Statistics and Causes of Death*. Geneva: WHO.

Whittemore, A. S. and Gong, G. (1991). Poisson regression with missclassified counts: Application to cervical cancer mortality rates. *Applied Statistics*, **40**, 81-93.

See Also

`cervical_validation`, `pogitBvs`

`cervical_validation` *Cervical cancer valiation data*

Description

Additionally to the main study sample (see `cervical`), validation data are available that give information on how likely physicians from different countries are to identify and correctly report a true cervical cancer death. In that study, a sample of physicians in each country completed a death certificate for one specific patient who had died of cervical cancer and the number of correct death certificates in each country was recorded. Validation data are therefore available on country level but provide no information on the reporting probability specific for age.

Usage

```
data(cervical_validation)
```

Format

A data frame with 4 rows and 6 variables:

`v` number of correct death certificates in each country in the validation sample

`m` size of validation sample in each country

`country` factor variable of European countries

`W.1`, `W.2`, `W.3` predictor variables for country effects using dummy coding (i.e. England, France, Italy)

Note

Belgium is used as the reference category.

Source

Kelson, M. and Farebrother, M. (1987). The Effect of Inaccuracies in Death Certification and Coding Practices in the European Economic Community (EEC) on International Cancer Mortality Statistics. *International Journal of Epidemiology*, 16, **3**, 411-414.

Whittemore, A. S. and Gong, G. (1991). Poisson regression with missclassified counts: Application to cervical cancer mortality rates. *Applied Statistics*, **40**, 81-93.

See Also

cervical, pogitBvs

---

logitBvs *Bayesian variable selection for the binomial logit model*

Description

This function performs Bayesian variable selection for binomial logit regression models via spike and slab priors. A cluster-specific random intercept can be included in the model to account for within-cluster dependence with variance selection of the random intercept to determine whether there is between-cluster variation in the model. For posterior inference, a MCMC sampling algorithm is used which is based on data augmentation.

Usage

```
logitBvs(
  y,
  N,
  X,
  model = list(),
  prior = list(),
  mcmc = list(),
  start = NULL,
  BVS = TRUE
)
```

Arguments

y	an integer vector of binomial counts
N	an integer vector containing the number of trials
X	a design matrix (including an intercept term)
model	an (optional) list specifying the structure of the model (see details)
prior	an (optional) list of prior settings and hyper-parameters controlling the priors (see details)
mcmc	an (optional) list of MCMC sampling options (see details)
start	an (optional), numeric vector containing starting values for the regression effects (including an intercept term); defaults to NULL (i.e. a vector of zeros is used).
BVS	if TRUE (default), Bayesian variable selection is performed to identify regressors with a non-zero effect; otherwise, an unrestricted model is estimated (without variable selection).

Details

The method provides Bayesian variable selection for binomial logit models using mixture priors with a spike and a slab component to identify regressors with a non-zero effect. More specifically, a Dirac spike is used, i.e. a point mass at zero and (by default), the slab component is specified as a scale mixture of normal distributions, resulting in a Student-t distribution with 2`psi.nu` degrees of freedom. In the more general random intercept model, variance selection of the random intercept is based on the non-centered parameterization of the model, where the signed standard deviation θ_α of the random intercept term appears as a further regression effect in the model equation. For details, see Wagner and Duller (2012).

The implementation of Bayesian variable selection further relies on the representation of the binomial logit model as a Gaussian regression model in auxiliary variables. Data augmentation is based on Fussl et al. (2013), who show that the binomial logit model can be represented as a linear regression model in the latent variable, which has an interpretation as the difference of aggregated utilities. The error distribution in the auxiliary model is approximated by a finite scale mixture of normal distributions, where the mixture parameters are taken from the R package `binomlogit`. See Fussl (2014) for details.

For details concerning the sampling algorithm see Dvorzak and Wagner (2016) and Wagner and Duller (2012).

Details for model specification (see arguments):

`model`: `deltafix` an indicator vector of length `ncol(X)-1` specifying which regression effects are subject to selection (i.e., 0 = subject to selection, 1 = fix in the model); defaults to a vector of zeros.

`gammafix` an indicator for variance selection of the random intercept term (i.e., 0 = with variance selection (default), 1 = no variance selection); only used if a random intercept is includued in the model (see `ri`).

`ri` logical. If `TRUE`, a cluster-specific random intercept is included in the model; defaults to `FALSE`.

`clusterID` a numeric vector of length equal to the number of observations containing the cluster ID c = 1,...,C for each observation (required if `ri=TRUE`).

`prior`: `slab` distribution of the slab component, i.e. `"Student"` (default) or `"Normal"`.

`psi.nu` hyper-parameter of the Student-t slab (used for a `"Student"` slab); defaults to 5.

`m0` prior mean for the intercept parameter; defaults to 0.

`M0` prior variance for the intercept parameter; defaults to 100.

`aj0` a vector of prior means for the regression effects (which is encoded in a normal distribution, see notes); defaults to vector of zeros.

`V` variance of the slab; defaults to 5.

`w` hyper-parameters of the Beta-prior for the mixture weight ω; defaults to `c(wa0=1, wb0=1)`, i.e. a uniform distribution.

`pi` hyper-parameters of the Beta-prior for the mixture weight π; defaults to `c(pa0=1, pb0=1)`, i.e. a uniform distribution.

`mcmc`: `M` number of MCMC iterations after the burn-in phase; defaults to 8000.

`burnin` number of MCMC iterations discarded as burn-in; defaults to 2000.

`thin` thinning parameter; defaults to 1.

`startsel` number of MCMC iterations drawn from the unrestricted model (e.g., `burnin/2`); defaults to 1000.

`verbose` MCMC progress report in each `verbose`-th iteration step; defaults to 500. If `verbose=0`, no output is generated.

`msave` returns additional output with variable selection details (i.e. posterior samples for ω, δ, π, γ); defaults to `FALSE`.

Value

The function returns an object of class "`pogit`" with methods `print.pogit`, `summary.pogit` and `plot.pogit`.

The returned object is a list containing the following elements:

`samplesL`	a named list containing the samples from the posterior distribution of the parameters in the binomial logit model (see also `msave`):
	`alpha, thetaAlpha` regression coefficients α and θ_α
	`pdeltaAlpha` P(δ_α=1)
	`psiAlpha` scale parameter ψ_α of the slab component
	`pgammaAlpha` P(γ_α=1)
	`ai` cluster-specific random intercept
`data`	a list containing the data `y`, `N` and `X`
`model.logit`	a list containing details on the model specification, see details for `model`
`mcmc`	see details for `mcmc`
`prior.logit`	see details for `prior`
`dur`	a list containing the total runtime (`total`) and the runtime after burn-in (`durM`), in seconds
`BVS`	see arguments
`start`	a list containing starting values, see arguments
`family`	"logit"
`call`	function call

Note

If prior information on the regression parameters is available, this information is encoded in a normal distribution instead of the spike and slab prior (`BVS` is set to `FALSE`).

For binary observations, a vector of ones for the number of trials `N` is required.

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>, Helga Wagner

References

Dvorzak, M. and Wagner, H. (2016). Sparse Bayesian modelling of underreported count data. *Statistical Modelling*, **16**(1), 24 - 46, doi:10.1177/1471082x15588398.

Fussl, A., Fruehwirth-Schnatter, S. and Fruehwirth, R. (2013). Efficient MCMC for Binomial Logit Models. *ACM Transactions on Modeling and Computer Simulation*, 23, **1**, Article 3, 1-21.

Fussl, A. (2014). `binomlogit`: Efficient MCMC for Binomial Logit Models. R package version 1.2, https://CRAN.R-project.org/package=binomlogit.

Wagner, H. and Duller, C. (2012). Bayesian model selection for logistic regression models with random intercept. *Computational Statistics and Data Analysis*, **56**, 1256-1274.

See Also

pogitBvs

Examples

```
## Not run:
## Examples below should take about 1-2 minutes.

# load simulated data set 'simul_binomial'
data(simul_binomial)
y <- simul_binomial$y
N <- simul_binomial$N
X <- as.matrix(simul_binomial[, -c(1, 2)])

# Bayesian variable selection for simulated data set
m1 <- logitBvs(y = y, N = N, X = X)

# print, summarize and plot results
print(m1)
summary(m1)
plot(m1)

# MCMC sampling without BVS with specific MCMC and prior settings
m2 <- logitBvs(y = y, N = N, X = X, prior = list(slab = "Normal"),
               mcmc = list(M = 4000, burnin = 1000, thin = 5),
               BVS = FALSE)
print(m2)
summary(m2)
plot(m2, maxPlots = 4)

# BVS with specification of regression effects subject to selection
m3 <- logitBvs(y = y, N = N, X = X, mcmc = list(M = 4000, burnin = 1000),
               model = list(deltafix = c(1, 1, 1, 0, 0, 0, 1, 0, 0)))
print(m3)
summary(m3)
plot(m3, burnin = FALSE, maxPlots = 4)
plot(m3, type = "acf", maxPlots = 4)

## End(Not run)
```

negbinBvs *Bayesian variable selection for the negative binomial model*

Description

This function performs Bayesian regression modelling of overdispersed count data including variable selection via spike and slab priors. Posterior inference is based on MCMC sampling techniques.

Usage

```
negbinBvs(
  y,
  offset = NULL,
  X,
  model = list(),
  prior = list(),
  mcmc = list(),
  start = NULL,
  BVS = TRUE
)
```

Arguments

y	an integer vector of count data
offset	an (optional) offset term; should be NULL or an integer vector of length equal to the number of counts.
X	a design matrix (including an intercept term)
model	an (optional) list specifying the structure of the model (see details)
prior	an (optional) list of prior settings and hyper-parameters controlling the priors (see details)
mcmc	an (optional) list of MCMC sampling options (see details)
start	an (optional), numeric vector containing starting values for the regression effects (including an intercept term); defaults to NULL (i.e. a vector of zeros is used).
BVS	if TRUE (default), Bayesian variable selection is performed to identify regressors with a non-zero effect; otherwise, an unrestricted model is estimated (without variable selection).

Details

The method provides Bayesian variable selection in regression modelling of overdispersed count data. The negative binomial distribution is derived marginally from a Poisson-Gamma (mixture) model, which can be interpreted as an overdispersed Poisson model with observation-specific random intercept $\log \gamma$, where $\gamma|\rho \sim \Gamma(\rho, \rho)$. A hyper-prior for ρ is specified as $\rho \sim \Gamma(c_0, C_0)$, see details for prior below. By default, variable selection is incorporated in the model based on

mixture priors with a spike and a slab component for the regression effects β. More specifically, a Dirac spike is used, i.e. a point mass at zero, and (by default), the slab component is specified as a scale mixture of normal distributions, resulting in a Student-t distribution with 2`psi.nu` degrees of freedom.

The MCMC sampling scheme relies on the representation of the conditional Poisson model as a Gaussian regression model in auxiliary variables, as described in `poissonBvs`. Data augmentation is based on the auxiliary mixture sampling algorithm of Fruehwirth-Schnatter et al. (2009). For details concerning the algorithm, see Dvorzak and Wagner (2016b), available on request from the authors.

Details for model specification (see arguments):

`model`: `deltafix` an indicator vector of length `ncol(X)-1` specifying which regression effects are subject to selection (i.e., 0 = subject to selection, 1 = fix in the model); defaults to a vector of zeros.

`prior`: `slab` distribution of the slab component, i.e. `"Student"` (default) or `"Normal"`.

`psi.nu` hyper-parameter of the Student-t slab (used for a `"Student"` slab); defaults to 5.

`m0` prior mean for the intercept parameter; defaults to 0.

`M0` prior variance for the intercept parameter; defaults to 100.

`aj0` a vector of prior means for the regression effects (which is encoded in a normal distribution, see notes); defaults to vector of zeros.

`V` variance of the slab; defaults to 5.

`w` hyper-parameters of the Beta-prior for the mixture weight ω; defaults to `c(wa0=1, wb0=1)`, i.e. a uniform distribution.

`c0, C0` scale and rate of the gamma prior for the hyper-parameter ρ; defaults to 2 and 1.

`eps` tuning parameter in the MH-step to sample ρ; defaults to 0.05.

`mcmc`: `M` number of MCMC iterations after the burn-in phase; defaults to 8000.

`burnin` number of MCMC iterations discarded as burn-in; defaults to 2000.

`thin` thinning parameter; defaults to 1.

`startsel` number of MCMC iterations drawn from the unrestricted model (e.g., `burnin/2`); defaults to 1000.

`verbose` MCMC progress report in each `verbose`-th iteration step; defaults to 500. If `verbose=0`, no output is generated.

`msave` returns additional output with variable selection details (i.e. posterior samples for ω, δ); defaults to `FALSE`.

Value

The function returns an object of class `"pogit"` with methods `print.pogit`, `summary.pogit` and `plot.pogit`.

The returned object is a list containing the following elements:

`samplesNB` a named list containing the samples from the posterior distribution of the parameters in the negative binomial model (see also `msave`):

`beta, rho` regression coefficients β and ρ

`pdeltaBeta` P(δ_β=1)

	psiBeta scale parameter ψ_β of the slab component
data	a list containing the data y, offset and X
model.nb	a list containing details on the model specification, see details for model
mcmc	see details for mcmc
prior.nb	see details for prior
dur	a list containing the total runtime (total) and the runtime after burn-in (durM), in seconds
acc.rho	acceptance rate of parameter ρ
BVS	see arguments
start	a list containing starting values, see arguments
family	"negbin"
call	function call

Note

Alternatively, a Poisson model with observation-specific normal random intercept (i.e., a Poisson-log-normal mixture model) can be used to deal with overdispersion of count data, which is provided in the function poissonBvs.

If prior information on the regression parameters is available, this information is encoded in a normal distribution instead of the spike and slab prior (consequently, BVS is set to FALSE).

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>

References

Dvorzak, M. and Wagner, H. (2016b). Bayesian inference for overdispersed count data subject to underreporting - An application to norovirus illness in Germany. (Unpublished) working paper.

Fruehwirth-Schnatter, S., Fruehwirth, R., Held, L. and Rue, H. (2009). Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data. *Statistics and Computing*, **19**, 479 - 492.

See Also

poissonBvs

Examples

```
## Not run:
## Examples below should take about 1-2 minutes.

## ------ (use simul_pois1) ------
data(simul_pois1)
y <- simul_pois1$y
X <- as.matrix(simul_pois1[, -1])

# Bayesian variable selection for simulated data set
```

```
m1 <- negbinBvs(y = y, X = X)

# print results (check acceptance rate for 'rho')
print(m1)

# re-run with adapted tuning parameter 'eps'
m2 <- negbinBvs(y = y, X = X, prior = list(eps = 0.4))

# print and summarize results
print(m2)
summary(m2)

# alternatively, compare results to overdispersed Poisson model with
# normal random intercept (subject to selection), provided in 'poissonBvs'

# specify observation-specific random intercept
cID <- seq_along(y)
m3  <- poissonBvs(y = y, X = X, model = list(ri = TRUE, clusterID = cID))

# print, summarize and plot results
print(m3)
summary(m3)
# note that thetaB is not selected (!)

plot(m3, burnin = FALSE, thin = FALSE)


## ------ (use data set "azdrg112" from package "COUNT") ------

if (!requireNamespace("COUNT", quietly = TRUE)){
 stop("package 'COUNT' is needed for this example to work.
       Please install it.")
}

library(COUNT)
# load data set 'azdrg112'
# (Arizona Medicare data for DRG (Diagnostic Related Group) 112)
data(azdrg112)

y <- as.numeric(azdrg112$los) # hospital length of stay: 1-53 days
X <- as.matrix(azdrg112[,-1]) # covariates (gender, type1, age75)
m4 <- negbinBvs(y = y, X = X, mcmc = list(M = 4000))

# print results (check acceptance rate for 'rho')
print(m4)
summary(m4)
plot(m4, burnin = FALSE)

# adapte tuning parameter eps (and set BVS to FALSE)
prior <- list(eps = 0.1)
m5 <- negbinBvs(y = y, X = X, mcmc = list(M = 4000), prior = prior,
                BVS = FALSE)
```

```
# print, summarize and plot results
print(m5)
summary(m5)
plot(m5, burnin = FALSE, thin = FALSE)
plot(m5, type = "acf", lag.max = 50)

## End(Not run)
```

---

plot.pogit *Plot an object of class* pogit

Description

This function provides traceplots, autocorrelation plots and density plots of the MCMC samples for an object of class "pogit" to graphically assess convergence of the MCMC simulations. It also displays the (model averaged) posterior means and 95%-HPD intervals for the regression effects.

Usage

```
## S3 method for class 'pogit'
plot(
  x,
  type = "traceplot",
  burnin = TRUE,
  thin = TRUE,
  lag.max = NULL,
  ci = TRUE,
  maxPlots = NULL,
  ...
)
```

Arguments

x	an object of class pogit
type	type of plot: "traceplot" (default) for traceplots of the MCMC draws, "acf" for autocorrelation plots of the MCMC draws, "density" for density plots and "hpd" to display (model averaged) posterior means with 95%-HPD intervals for the regression effects.
burnin	logical. If TRUE (default), burn-in draws (as specified in x) are discarded.
thin	logical. If TRUE (default), thinning (as specified in x) is considered for diagnostic MCMC plots.
lag.max	maximum lag for autocorrelation plot; if NULL (default), the default of acf is used.
ci	logical. If TRUE (default), the confidence interval in the autocorrelation plot is shown (see acf for details).
maxPlots	maximum number of plots on a single page; if NULL (default), the number of plots dispayed on a single page is specified according to the used model.
...	further arguments (not used)

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>

Examples

```
## see examples for pogitBvs, logitBvs, poissonBvs and negbinBvs
```

`pogitBvs` *Bayesian variable selection for the Pogit model*

Description

This function performs Bayesian variable selection for a Poisson-Logistic (Pogit) model via spike and slab priors. For posterior inference, a MCMC sampling scheme is used that relies on augmenting the observed data by the unobserved counts and involves only Gibbs sampling steps.

Usage

```
pogitBvs(
  y,
  E = NULL,
  X,
  W = NULL,
  validation = NULL,
  method = "val",
  model = list(),
  prior = list(),
  mcmc = list(),
  start = list(),
  BVS = TRUE
)
```

Arguments

`y`	an integer vector of observed counts for units i = 1,...,I
`E`	an (optional) vector containing total exposure times (offset); should be `NULL` or an integer vector of length equal to the number of counts.
`X`	a design matrix in the Poisson part of the joint model
`W`	a design matrix in the logit part of the joint model (can be a subset of `X`) or `NULL`, if the same design matrix is used in both sub-models, i.e. `W = X`.
`validation`	a two-column data frame or list with the number of reported cases (= `v`) in the validation sample and the number of total cases (= `m`) subject to the fallible reporting process (i.e. validation sample size) for each unit (or sub-category); required if `method = "val"`, otherwise `NULL`. The number of rows must conform with the number of rows in `W` or with the number of units I (if `X = W`), respectively.

method	the method to be used to obtain parameter identification: The default method "val" requires a small sample of validation data (see validation). If the information on all or some parameters of the reporting process is not provided by validation data, an informative prior distribution for the regression effects in the logit sub-model can be used (method = "infprior"). This prior information is encoded in a normal distribution instead of the spike and slab prior (see the details for prior).
model	a list specifying the structure of the model (see details)
prior	an (optional) list of prior settings and hyper-parameters controlling the priors (see details)
mcmc	an (optional) list of MCMC sampling options (see details)
start	an (optional) list containing starting values for the regression effects in both sub-models (see details)
BVS	if TRUE (default), Bayesian variable selection (in at least one part of the joint model) is performed to identify regressors with a non-zero effect; otherwise, an unrestricted model is estimated (without variable selection).

Details

The method provides Bayesian variable selection for regression models of count data subject to under-reporting using mixture priors with a spike and a slab component. By augmenting the observed count data with the unobserved counts, the resulting model can be factorized into a Poisson and a binomial logit model part. Hence, for this two-part model, sampling algorithms for a Poisson and a binomial logit model can be used which are described in poissonBvs and logitBvs. Bayesian variable selection is incorporated in both parts of the joint model using mixture priors with a Dirac spike and (by default) a Student-t slab. The implementation relies on the representation of the respective model as a Gaussian regression model in auxiliary variables (see again the help for the respective function). Though variable selection is primarily used to identify regressors with a non-zero effect, it can also be useful for identification of the Pogit model.

By default, identification of the Pogit model is achieved by additional information on the reporting process through validation data and incorporation of variable selection. If the information on the parameters of the reporting process is not provided by validation data, the identification of the model parameters has to be guaranteed by specifying an informative prior distribution (see arguments).

To model under-reported clustered data, a cluster-specific random intercept can be included in both model parts of the Pogit model to account for dependence within clusters. Bayesian variance selection is applied to determine whether there is within-cluster dependence in either part of the model. Note that an observation-specific random intercept in the Poisson sub-model yields an overdispersed Pogit model for unobserved heterogeneity.

For details concerning the sampling algorithm see Dvorzak and Wagner (2016).

Details for model specification (see arguments):

model: deltaBetafix, deltaAlphafix indicator vectors of length ncol(X)-1 and ncol(W)-1, respectively, for the Poisson and the logit sub-model, to specify which regression effects are subject to selection (i.e., 0 = subject to selection, 1 = fix in the model); defaults to vectors of zeros.

`gammaBetafix`, `gammaAlphafix` indicators for variance selection of the random intercept term in the Poisson and the logit sub-model (i.e., 0 = with variance selection (default), 1 = no variance selection); only used if a random intercept is included in either part of the joint model (see `riBeta` and `riAlpha`, respectively).

`riBeta`, `riAlpha` logical. If `TRUE`, a cluster-specific random intercept is included in the respective part of the joint model; defaults to `FALSE`.

`clBetaID`, `clAlphaID` numeric vectors of length equal to the number of observations containing the cluster ID c = 1,...,C for each unit (or sub-category) in the respective sub-model (required if `riBeta=TRUE` or `riAlpha=TRUE`, respectively).

`subcat` a factor variable of length equal to the number of units that specifies for which sub-category validation data are available (is required if `W` is a subset of `X`). If `NULL` (default), it is presumed that validation data are available for each unit (see also examples).

`prior`: `slabP`, `slabL` distribution of the slab component in the Poisson and logit sub-model, i.e. `"Student"` (default) or `"Normal"`.

`psi.nuP`, `psi.nuL` hyper-parameter of the Student-t slab in the respective sub-model (used for a Student-t slab); defaults to 5.

`m0b`, `m0a` prior mean for the intercept parameter in the Poisson and the logit model; defaults to 0. If the argument `method = "infprior"`, the specification of `m0a` is required.

`M0b`, `M0a` prior variance for the intercept parameter in the Poisson and the logit model; defaults to 100.

`bj0`, `aj0` a vector of prior means for the regression effects in the Poisson and the logit sub-model (which is encoded in a normal distribution, see notes); defaults to a vector of zeros. If the argument `method = "infprior"`, the specification of `aj0` is mandatory.

`VP`, `VL` variance of the slab in the respective sub-model; defaults to 5.

`wBeta`, `wAlpha` hyper-parameters of the Beta-prior for the mixture weights ω_β and ω_α in the respective sub-model; defaults to `c(wa0=1, wb0=1)`, i.e. a uniform distribution.

`piBeta`, `piAlpha` hyper-parameters of the Beta-prior for the mixture weights π_β and π_α in the respective sub-model; defaults to `c(pa0=1, pb0=1)`, i.e. a uniform distribution.

`mcmc`: `M` number of MCMC iterations after the burn-in phase; defaults to 8000.

`burnin` number of MCMC iterations discarded as burn-in; defaults to 2000.

`thin` thinning parameter; defaults to 1.

`startsel` number of MCMC iterations drawn from the unrestricted model (e.g., `burnin/2`); defaults to 1000.

`verbose` MCMC progress report in each `verbose`-th iteration step; defaults to 500. If `verbose=0`, no output is generated.

`msave` returns additional output with variable selection details (i.e. posterior samples for ω_β, ω_α, δ_β, δ_α, π_β, π_α, γ_β, γ_α); defaults to `FALSE`.

`start`: `beta` a vector of length `ncol(X)` containing starting values for the regression parameters β in the Poisson model part. By default, a Poisson glm is fitted to the observed counts.

`alpha` a vector of length `ncol(W)` containing starting values for the regression parameters α in the logit model part. By default, a binomial glm is fitted to the validation data for `method = "val"`. If `method = "infprior"`, starting values for α are sampled from the (informative) prior distribution.

`firth` logical. If `TRUE`, a logistic regression model applying Firth's correction to the likelihood using `logistf` is fitted to the validation data (only used if `method = "val"`).

Value

The function returns an object of class "pogit" with methods print.pogit, summary.pogit and plot.pogit.

An object of class "pogit" is a list containing the following elements:

samplesL	a named list containing the samples from the posterior distribution of the parameters in the logit part of the joint model (see also msave): alpha, thetaAlpha regression coefficients α and θ_α pdeltaAlpha P(δ_α=1) psiAlpha scale parameter ψ_α of the slab component pgammaAlpha P(γ_α=1) ai cluster-specific random intercept
samplesP	a named list containing the samples from the posterior distribution of the parameters in the Poisson part of the joint model (see also msave): beta, thetaBeta regression coefficients β and θ_β pdeltaBeta P(δ_β=1) psiBeta scale parameter ψ_β of the slab component pgammaBeta P(γ_β=1) bi cluster-specific random intercept
data	a list containing the data y, E, X, W, val and subcat
model.logit	a list containing details on the model specification in the logit sub-model, see details for model
model.pois	a list containing details on the model specification in the Poisson sub-model, see details for model
mcmc	see details for mcmc
prior.logit	see details for prior
prior.pois	see details for prior
dur	a list containing the total runtime (total) and the runtime after burn-in (durM), in seconds
BVS	see arguments
method	see arguments
start	a list containing starting values, see arguments
family	"pogit"
call	function call

Note

If method = "infprior", an informative prior for the regression parameters in the logit model is required to guarantee identification of the model parameters. Otherwise, identification of the Pogit model may be weak and inference will be biased.

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>, Helga Wagner

References

Dvorzak, M. and Wagner, H. (2016). Sparse Bayesian modelling of underreported count data. *Statistical Modelling*, **16**(1), 24 - 46, doi:10.1177/1471082x15588398.

See Also

logitBvs, poissonBvs

Examples

```
## Not run:
## Examples below (except for m3) should take 3-4 minutes.

## ------ (use simul1) ------
# load simulated data set 'simul1'
data(simul1)
y <- simul1$y
E <- simul1$E
X <- as.matrix(simul1[, -c(1,2,8,9)]) # W = X
validation <- simul1[, c("m", "v"), drop = FALSE]

# function call (with specific MCMC settings)
m1 <- pogitBvs(y = y, E = E, X = X, validation = validation,
               mcmc = list(M = 4000, thin = 5, verbose = 1000))

# print, summarize and plot results
print(m1)
summary(m1)
plot(m1)

# show traceplots disregarding burn-in and thinning
plot(m1, burnin = FALSE, thin = FALSE)
# show density plot of MCMC draws
plot(m1, type = "density")

# informative prior instead of validation data (change prior settings)
# e.g. available prior information on reporting probabilities
p.a0 <- 0.9
p.a  <- c(0.125, 0.5, 0.5, 0.5)
m0a_inf <- log(p.a0/(1 - p.a0))  # prior information for alpha_0
aj0_inf <- log(p.a/(1 - p.a))    # prior information for alpha

prior.set <- list(m0a = m0a_inf, aj0 = aj0_inf, VL = 0.005, slabL = "Normal")
m2 <- pogitBvs(y = y, E = E, X = X, method = "infprior", prior = prior.set,
               mcmc = list(M = 4000, burnin = 2000, thin = 2), BVS = FALSE)
print(m2)
summary(m2)
plot(m2)
```

```
plot(m2, type = "acf", lag.max = 30)

## ------ (use simul2) ------
# complex model (with a long (!) runtime)

# load simulated data set 'simul2'
data(simul2)
y <- simul2$y
E <- simul2$E
cID <- simul2$cID
X <- as.matrix(simul2[, -c(1:3,9,10)])
validation <- simul2[, c("v", "m"), drop = FALSE]

# function call (with random intercept in both sub-models)
model <- list(riBeta = 1, riAlpha = 1, clBetaID = cID, clAlphaID = cID)
m3 <- pogitBvs(y = y, E = E, X = X, validation = validation, model = model,
               mcmc = list(M = 6000, burnin = 200, thin = 10), BVS = TRUE)
print(m3)
summary(m3)
plot(m3)

## ------ (use cervical cancer data) ------
# load cervical cancer data
data(cervical)
data(cervical_validation)
y <- cervical$y
E <- cervical$E
X <- as.matrix(cervical[, -c(1:4)])
validation <- cervical_validation[, c(1, 2), drop = FALSE]
W          <- as.matrix(cervical_validation[, -c(1:3)])
subcat     <- factor(as.numeric(cervical$country))

# function call
m4 <- pogitBvs(y = y, E = E, X = X, W = W, validation = validation,
               model = list(subcat = subcat), mcmc = list(M = 10000,
               burnin = 2000, thin = 10), start = list(firth = TRUE),
               BVS = TRUE)
print(m4)
# additionally compute estimated risks and reporting probabilities
summary(m4, printRes = TRUE)
plot(m4, burnin = FALSE, thin = FALSE)
plot(m4, type = "acf", lag.max = 50)

# informative prior instead of validation data (change prior settings)
# e.g. prior information on country-specific reporting probabilities
p.a0 <- 0.85
p.a  <- c(0.99, 0.70, 0.85)
m0a_inf <- log(p.a0/(1 - p.a0))  # prior information for alpha_0
aj0_inf <- log(p.a/(1 - p.a))    # prior information for alpha

prior.set <- list(m0a = m0a_inf, aj0 = aj0_inf, VL = 0.005, slabL = "Normal")
m5 <- pogitBvs(y = y, X = X, W = W, E = E, method = "infprior",
               model = list(subcat = subcat), prior = prior.set,
```

```
                mcmc = list(M = 10000, burnin = 2000, thin = 10))
print(m5)
summary(m5, printRes = TRUE)
plot(m5, burnin = FALSE, thin = FALSE)
plot(m5, type = "acf", lag.max = 50)

## End(Not run)
```

---

poissonBvs *Bayesian variable selection for the Poisson model*

Description

This function performs Bayesian variable selection for Poisson regression models via spike and slab priors. A cluster- (or observation-) specific random intercept can be included in the model to account for within-cluster dependence (or overdispersion) with variance selection of the random intercept. For posterior inference, a MCMC sampling scheme is used which relies on data augmentation and involves only Gibbs sampling steps.

Usage

```
poissonBvs(
  y,
  offset = NULL,
  X,
  model = list(),
  mcmc = list(),
  prior = list(),
  start = NULL,
  BVS = TRUE
)
```

Arguments

y	an integer vector of count data
offset	an (optional) offset term; should be NULL or an integer vector of length equal to the number of counts.
X	a design matrix (including an intercept term)
model	an (optional) list specifying the structure of the model (see details)
mcmc	an (optional) list of MCMC sampling options (see details)
prior	an (optional) list of prior settings and hyper-parameters controlling the priors (see details)
start	an (optional), numeric vector containing starting values for the regression effects (including an intercept term); defaults to NULL (i.e. a vector of zeros is used).
BVS	if TRUE (default), Bayesian variable selection is performed to identify regressors with a non-zero effect; otherwise, an unrestricted model is estimated (without variable selection).

Details

The method provides a Bayesian framework for variable selection in regression modelling of count data using mixture priors with a spike and a slab component to identify regressors with a non-zero effect. More specifically, a Dirac spike is used, i.e. a point mass at zero, and (by default), the slab component is specified as a scale mixture of normal distributions, resulting in a Student-t distribution with 2`psi.nu` degrees of freedom. In the more general random intercept model, variance selection of the random intercept is based on the non-centered parameterization of the model, where the signed standard deviation θ_β of the random intercept term appears as a further regression effect in the model equation. For further details, see Wagner and Duller (2012).

The implementation of Bayesian variable selection further relies on the representation of the Poisson model as a Gaussian regression model in auxiliary variables. Data augmentation is based on the auxiliary mixture sampling algorithm of Fruehwirth-Schnatter et al. (2009), where the inter-arrival times of an assumed Poisson process are introduced as latent variables. The error distribution, a negative log-Gamma distribution, in the auxiliary model is approximated by a finite mixture of normal distributions where the mixture parameters of the matlab package `bayesf`, Version 2.0 of Fruehwirth-Schnatter (2007) are used. See Fruehwirth-Schnatter et al. (2009) for details.

For details concerning the sampling algorithm, see Dvorzak and Wagner (2016) and Wagner and Duller (2012).

Details for model specification (see arguments):

`model`: `deltafix` an indicator vector of length `ncol(X)-1` specifying which regression effects are subject to selection (i.e., 0 = subject to selection, 1 = fix in the model); defaults to a vector of zeros.

`gammafix` an indicator for variance selection of the random intercept term (i.e., 0 = with variance selection (default), 1 = no variance selection); only used if a random intercept is includued in the model (see `ri`).

`ri` logical. If `TRUE`, a cluster- (or observation-) specific random intercept is included in the model; defaults to `FALSE`.

`clusterID` a numeric vector of length equal to the number of observations containing the cluster ID c = 1,...,C for each observation (required if `ri=TRUE`). Note that `seq_along(y)` specifies an overdispersed Poisson model with observation-specific (normal) random intercept (see note).

`prior`: `slab` distribution of the slab component, i.e. `"Student"` (default) or `"Normal"`.

`psi.nu` hyper-parameter of the Student-t slab (used for a `"Student"` slab); defaults to 5.

`m0` prior mean for the intercept parameter; defaults to 0.

`M0` prior variance for the intercept parameter; defaults to 100.

`aj0` a vector of prior means for the regression effects (which is encoded in a normal distribution, see note); defaults to vector of zeros.

`V` variance of the slab; defaults to 5.

`w` hyper-parameters of the Beta-prior for the mixture weight ω; defaults to `c(wa0=1, wb0=1)`, i.e. a uniform distribution.

`pi` hyper-parameters of the Beta-prior for the mixture weight π; defaults to `c(pa0=1, pb0=1)`, i.e. a uniform distribution.

`mcmc`: `M` number of MCMC iterations after the burn-in phase; defaults to 8000.

`burnin` number of MCMC iterations discarded as burn-in; defaults to 2000.

`thin` thinning parameter; defaults to 1.

`startsel` number of MCMC iterations drawn from the unrestricted model (e.g., `burnin/2`); defaults to 1000.

`verbose` MCMC progress report in each `verbose`-th iteration step; defaults to 500. If `verbose=0`, no output is generated.

`msave` returns additional output with variable selection details (i.e. posterior samples for ω, δ, π, γ); defaults to `FALSE`.

Value

The function returns an object of class "`pogit`" with methods `print.pogit`, `summary.pogit` and `plot.pogit`.

The returned object is a list containing the following elements:

`samplesP`	a named list containing the samples from the posterior distribution of the parameters in the Poisson model (see also `msave`): `beta, thetaBeta` regression coefficients β and θ_β `pdeltaBeta` P(δ_β=1) `psiBeta` scale parameter ψ_β of the slab component `pgammaBeta` P(γ_β=1) `bi` cluster- (or observation-) specific random intercept
`data`	a list containing the data `y`, `offset` and `X`
`model.pois`	a list containing details on the model specification, see details for `model`
`mcmc`	see details for `mcmc`
`prior.pois`	see details for `prior`
`dur`	a list containing the total runtime (`total`) and the runtime after burn-in (`durM`), in seconds
`BVS`	see arguments
`start`	a list containing starting values, see arguments
`family`	"poisson"
`call`	function call

Note

If prior information on the regression parameters is available, this information is encoded in a normal distribution instead of the spike and slab prior (consequently, `BVS` is set to `FALSE`).

This function can also be used to accommodate overdispersion in count data by specifying an observation-specific random intercept (see details for `model`). The resulting model is an alternative to the negative binomial model, see `negbinBvs`. Variance selection of the random intercept may be useful to examine whether overdispersion is present in the data.

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>, Helga Wagner

References

Dvorzak, M. and Wagner, H. (2016). Sparse Bayesian modelling of underreported count data. *Statistical Modelling*, **16**(1), 24 - 46, doi:10.1177/1471082x15588398.

Fruehwirth-Schnatter, S. (2007). Matlab package `bayesf` 2.0 on *Finite Mixture and Markov Switching Models*, Springer. https://statmath.wu.ac.at/~fruehwirth/monographie/.

Fruehwirth-Schnatter, S., Fruehwirth, R., Held, L. and Rue, H. (2009). Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data. *Statistics and Computing*, **19**, 479 - 492.

Wagner, H. and Duller, C. (2012). Bayesian model selection for logistic regression models with random intercept. *Computational Statistics and Data Analysis*, **56**, 1256 - 1274.

See Also

pogitBvs, negbinBvs

Examples

```
## Not run:
## Examples below should take about 1-2 minutes.

## ------ (use simul_pois1) ------
# load simulated data set 'simul_pois1'
data(simul_pois1)
y <- simul_pois1$y
X <- as.matrix(simul_pois1[, -1])

# Bayesian variable selection for simulated data set
m1 <- poissonBvs(y = y, X = X)

# print, summarize and plot results
print(m1)
summary(m1)
plot(m1, maxPlots = 4)
plot(m1, burnin = FALSE, thin = FALSE, maxPlots = 4)
plot(m1, type = "acf")

# MCMC sampling without BVS with specific MCMC and prior settings
m2 <- poissonBvs(y = y, X = X, prior = list(slab = "Normal"),
                 mcmc = list(M = 6000, thin = 10), BVS = FALSE)
print(m2)
summary(m2, IAT = TRUE)
plot(m2)
# show traceplots disregarding thinning
plot(m2, thin = FALSE)

# specification of an overdispersed Poisson model with observation-specific
# (normal) random intercept
cID <- seq_along(y)
m3  <- poissonBvs(y = y, X = X, model = list(ri = TRUE, clusterID = cID))

# print, summarize and plot results
```

```
print(m3)
summary(m3)
# note that variance selection of the random intercept indicates that
# overdispersion is not present in the data
plot(m3, burnin = FALSE, thin = FALSE)

## ------ (use simul_pois2) ------
# load simulated data set 'simul_pois2'
data(simul_pois2)
y <- simul_pois2$y
X <- as.matrix(simul_pois2[, -c(1,2)])
cID <- simul_pois2$cID

# BVS for a Poisson model with cluster-specific random intercept
m4 <- poissonBvs(y = y, X = X, model = list(ri = TRUE, clusterID = cID),
                 mcmc = list(M = 4000, burnin = 2000))
print(m4)
summary(m4)
plot(m4)

# similar to m4, but without variance selection of the random intercept term
model <- list(gammafix = 1, ri = 1, clusterID = cID)
m5 <- poissonBvs(y = y, X = X, model = model, mcmc = list(M = 4000, thin = 5))
print(m5)
summary(m5)
plot(m5)

# MCMC sampling without BVS for clustered observations
m6 <- poissonBvs(y = y, X = X, model = list(ri = 1, clusterID = cID),
                 BVS = FALSE)
print(m6)
summary(m6)
plot(m6, maxPlots = 4)

## End(Not run)
```

`print.pogit` *Print an object of class* `pogit`

Description

The default print method for a `pogit` object.

Usage

```
## S3 method for class 'pogit'
print(x, ...)
```

Arguments

x	an object of class pogit
...	further arguments passed to or from other methods (not used)

Details

Returns basic information about the model, the number of observations and covariates used, the number of regression effects subject to selection, MCMC options and the runtime used for the sampling algorithm. See summary.pogit for more details.

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>

---

simul1 *Simulated data set*

Description

The simulated data set simul1 considers a situation with four binary covariates in both sub-models of the Pogit model, i.e. X = W. The respective design matrix is built by computing all 2^4 possible 0/1 combinations and one observation is generated for each covariate pattern. The regression effects are set to beta = {0.75,0.5,-2,0,0} in the Poisson and to alpha = {2.2,-1.9,0,0,0} in the logit model. Additionally to the main study sample, validation data are available for each covariate pattern. For details concerning the simulation setup, see Dvorzak and Wagner (2016).

Usage

```
data(simul1)
```

Format

A data frame with 16 rows and the following 9 variables:

- y number of observed counts for each covariate pattern
- E total exposure time
- X.0 intercept
- X.1, X.2, X.3, X.4 binary covariates
- v number of reported cases for each covariate pattern in the validation sample
- m number of true cases subject to the fallible reporting process (sample size of validation data)

Source

Dvorzak, M. and Wagner, H. (2016). Sparse Bayesian modelling of underreported count data. *Statistical Modelling*, **16**(1), 24 - 46, doi:10.1177/1471082x15588398.

See Also

pogitBvs

---

simul2 *Simulated data set*

Description

The simulated data set simul2 considers a situation with clustered observations and four binary covariates in both sub-models of the Pogit model, i.e. X = W. The respective design matrix is built by computing all 2^4 possible 0/1 combinations and one observation is generated for each covariate pattern. C=50 clusters are built containing one unit with each of the resulting 16 covariate patterns, i.e. a total of I=800 units. The regression effects are set to beta = {0.75,0.1,0.1,0,0} in the Poisson and to alpha = {2.2,-0.3,0,-0.3,0} in the logit model. Random intercepts in both sub-models are simulated from a normal distribution with standard deviations θ_β=0.1 and θ_α=0.3. Additionally to the main study sample, validation data are available for each covariate pattern and cluster. For details concerning the simulation setup, see Dvorzak and Wagner (2016).

Usage

```
data(simul2)
```

Format

A data frame with 800 rows and the following 10 variables:

- y number of observed counts for each covariate pattern in each cluster
- E total exposure times for each unit
- cID cluster ID for each unit
- X.0 intercept
- X.1, X.2, X.3, X.4 binary covariates
- v number of reported cases for each covariate pattern in each cluster in the validation sample
- m number of true cases subject to the fallible reporting process (sample size of validation data)

Source

Dvorzak, M. and Wagner, H. (2016). Sparse Bayesian modelling of underreported count data. *Statistical Modelling*, **16**(1), 24 - 46, doi:10.1177/1471082x15588398.

See Also

pogitBvs

simul_binomial *Simulated data set*

Description

The data set simul_binomial contains simulated binomial data with 9 binary covariates. The design matrix is built by computing all 2^9 possible 0/1 combinations. The regression effects are set to alpha = {-0.5,0.2,-0.15,0.1,-1.1,0,0,1.2,-0.1,0.3}. The number of trials N are simulated from a Poisson distribution with parameter $\exp(\alpha)/(1+\exp(\alpha))$*100.

Usage

```
data(simul_binomial)
```

Format

A data frame with 512 rows and the following 12 variables:

y number of successes for each covariate pattern

N number of trials for each covariate pattern

X.0 intercept

X.1, X.2, X.3, X.4, X.5, X.6, X.7, X.8, X.9 binary covariates

See Also

logitBvs

simul_pois1 *Simulated data set*

Description

The data set simul_pois1 contains 300 simulated Poisson counts. 10 regressors are generated, six of them continuous N(0,1)-variables and four binary with $p(x_i) = 0.5$. The regression effects are set to beta = {2,1,0.6,0,0,1.2,0,0,0.4,-0.2,0.3}.

Usage

```
data(simul_pois1)
```

Format

A data frame with 300 rows and the following 12 variables:

y number of counts for each covariate pattern

X.0 intercept

X.1, X.2, X.3, X.4, X.5, X.6, X.7, X.8, X.9, X.10 covariates

See Also

poissonBvs

---

simul_pois2 *Simulated data set*

Description

The same simulation setup is used as in simul_pois1 but considers clustered observations. 10 regressors are generated, six of them continuous N(0,1)-variables and four binary with $p(x_i) = 0.5$. The regression effects are set to beta = {2,1,0.6,0,0,1.2,0,0,0.4,-0.2,0.3}. To simulate clustering, it is assumed that each of C=10 clusters is formed of 30 subjects and 10 random intercepts are generated from a normal distribution with zero mean and standard deviation $\theta = 0.1$.

Usage

```
data(simul_pois2)
```

Format

A data frame with 300 rows and the following 12 variables:

y number of counts for each covariate pattern in each cluster

cID cluster ID of each count

X.0 intercept

X.1, X.2, X.3, X.4, X.5, X.6, X.7, X.8, X.9, X.10 covariates

See Also

simul_pois1, poissonBvs

---

summary.pogit *Summary for posterior of a* pogit *object*

Description

Returns basic information about the model and the priors, MCMC details and (model averaged) posterior means with 95%-HPD intervals for the regression effects and estimated posterior inclusion probabilities.

Usage

```
## S3 method for class 'pogit'
summary(object, IAT = FALSE, printRes = FALSE, ...)

## S3 method for class 'summary.pogit'
print(x, ...)
```

Arguments

object	an object of class pogit
IAT	if TRUE, integrated autocorrelation times (IAT) and effective samples sizes (ESS) of the MCMC samples are computed (see details); defaults to FALSE.
printRes	if TRUE, model averaged posterior means for the reporting probabilities and risks are computed for the Pogit model; defaults to FALSE.
...	further arguments passed to or from other methods (not used)
x	a summary.pogit object produced by summary.pogit()

Details

To assess mixing and efficiency of MCMC sampling, the effective sample size (ESS) and the integrated autocorrelation time (IAT) are computed. ESS estimates the equivalent number of independent draws corresponding to the dependent MCMC draws and is defined as ESS = M/τ, where τ is the IAT and M is the number of MCMC iterations after the burn-in phase. IAT is computed as $\tau = 1 + 2\sum_{k=1}^{K} \rho(k)$ using the initial monotone sequence estimator (Geyer, 1992) for K and $\rho(k)$ is the empirical autocorrelation at lag k.

Value

an object of class summary.pogit

Author(s)

Michaela Dvorzak <m.dvorzak@gmx.at>

References

Geyer, C. J. (1992). Practical Markov Chain Monte Carlo. *Statistical Science*, **7**, 473-483.

Index