Package ‘bpr’

April 16, 2024

Type Package
Title Fitting Bayesian Poisson Regression
Version 1.0.8
Date 2024-04-15
Author Laura D'Angelo
Maintainer Laura D'Angelo <laura.dangelo@live.com>
Description Posterior sampling and inference for Bayesian Poisson regression models. The model specification makes use of Gaussian (or conditionally Gaussian) prior distributions on the regression coefficients. Details on the algorithm are found in D'Angelo and Canale (2023) <doi:10.1080/10618600.2022.2123337>.
License GPL (>= 2)
Encoding UTF-8
RoxygenNote 7.2.0
Imports Rcpp (>= 1.0.7), coda, MASS
LinkingTo Rcpp, RcppArmadillo, BH
NeedsCompilation yes
Repository CRAN
Date/Publication 2024-04-16 15:20:02 UTC

R topics documented:

mcmc_diagnostics .................................................. 2
merge_sim .............................................................. 3
plot.poisreg .......................................................... 4
plot.posterior_check ................................................. 4
posterior_predictive ............................................... 5
sample_bpr ........................................................... 6
summary.poisreg .................................................... 10

Index 13
mcmc_diagnostics  MCMC Convergence Diagnostics

Description

This function is a method for class poisreg. It prints convergence diagnostics and accuracy statistics of the MCMC output.

Usage

mcmc_diagnostics(object)

Arguments

object  object of class "poisreg" (usually, the result of a call to sample_bpr).

Details

The printed output of mcmc_diagnostics summarizes some common convergence diagnostics for Markov chains. The first part recaps the total length, burn-in and thinning used for the simulation.

The second part is a table with diagnostic statistics about each chain of the regression parameters. The first column is the effective sample size computed after removing the burn-in and thinning. The last two columns report the value and observed p-value of the Geweke test of equality of the first and last part of the chain.

The last part is printed only if multiple chains are computed. In this case, it reports the Gelman-Rubin statistics to test convergence to the same stationary distribution. Values much larger than 1 suggest lack of convergence to a common distribution.

Value

mcmc_diagnostics returns a list with elements:

- chain_length: total length of the MCMC chains.
- len_burnin: the length of the burn-in used to compute the estimates.
- thin: the thinning frequency used (from object).
- effSize: effective sample size of each parameter chain after removing burn-in and thinning. See effectiveSize.
- geweke: Geweke diagnostics of convergence of the chains (value of the test and p-value). See geweke.diag
- gelman_rubin: if nchains > 1, Gelman-Rubin diagnostics of convergence. See gelman.diag.

See Also

summary.poisreg, plot.poisreg, merge_sim, effectiveSize, geweke.diag, gelman.diag
**Examples**

# For examples see example(sample_bpr)

---

### Description

This function is a method for class `poisreg`. Merge multiple MCMC chains into a unique chain when sampling with `nchains > 1` is used.

### Usage

```r
merge_sim(object)
```

### Arguments

- **object**: object of class "poisreg" (usually, the result of a call to `sample_bpr`), with `nchains > 1`.

### Value

The function returns an object of class `poisreg` with a single element `$sim`. The returned chains (elements of `$sim`) are obtained by appending the simulated values of each independent chain, under the assumption that they all have reached the same stationary distribution.

### Examples

```r
library(MASS) # load the data set
define head(epil)

# Simulate multiple chains by setting nchains > 1
fit4 = sample_bpr( y ~ lbase*trt + lage + V4, data = epil,
                   iter = 1000,
                   nchains = 4, thin = 2)
# fit4 contains 4 elements with simulation ($sim, $sim2, $sim3, $sim4)

cmc_diagnoses(fit4)
# the Gelman-Rubin diagnostics confirms convergence of the 4
# independent chains to the same stationary distribution

fit4b = merge_sim(fit4)
str(fit4b$sim)
# fit 4b contains only one element $sim, of length 1500
# (which is the result of concatenating the 4 simulations, after removing the first 25%
# iterations as burn-in and keeping one iteration every two).
```
### plot.poisreg

**Plot Trace and Distribution of Regression Parameters**

**Description**

Plot Trace and Distribution of Regression Parameters

**Usage**

```r
## S3 method for class 'poisreg'
plot(x, ...)
```

**Arguments**

- `x` object of class "poisreg" (usually, the result of a call to `sample_bpr`).
- `...` further arguments passed to or from other methods.

**Value**

The function calls `plot.mcmc` on the matrix of sampled regression coefficients, and returns the trace of the sampled outputs and a density estimate for each variable in the chain.

**See Also**

`sample_bpr, plot.mcmc`

### plot.posterior_check

**Graphical Posterior Predictive Checks**

**Description**

This function is a method for class `posterior_check`. Plot diagnostic statistics for graphical posterior predictive checks.

**Usage**

```r
## S3 method for class 'posterior_check'
plot(x, ...)
```

**Arguments**

- `x` object of class "posterior_check" (usually, the result of a call to `posterior_predictive`).
- `...` other parameters to be passed through to plotting functions. See Details.
posterior_predictive

Details

It is possible to generate additional plots that compare the posterior predictive distribution of a statistic with the observed value. This is done through the parameter `stats`: it is a list with elements the function names of the statistics one wants to compare. Default is `stats = list("mean")`, other possible values are, e.g., "median", "sd", "max" etc.

Value

The function outputs (at least) three plots for graphical posterior predictive check. The first plot compares the empirical cumulative distribution function (ECDF) with the cumulative distribution function obtained with samples from the posterior predictive distribution (median and point-wise 95% credible bands). The second plot compares the distribution of the observed sample with the predictive distribution obtained using the maximum a posteriori (MAP) estimates of the regression parameters. The third plot compares the predictive distribution of a statistic (default is the mean) with the observed value of the same statistics, displayed with a red line.

See Also

posterior_predictive

Examples

library(MASS) # load the data set
code
head(epil)

fit = sample_bpr( y ~ lbase*trt + lage + V4, data = epil,
  iter = 1000)
plot(posterior_predictive(fit), stats = c("mean", "sd", "max"))
# plots for posterior predictive check
Arguments

object object of class "poisreg" (usually, the result of a call to sample_bpr).
new_X (optional) a data frame in which to look for variables with which to predict.

Value

The call to this function returns an object of S3 class posterior_check. The object is a list with the following elements:
data: the component from object (list with covariates X and response variable y).
y_pred: matrix of dimension \([n, \text{iter}]\) (with \(n\) sample size), each column is a draw from the posterior predictive distribution.
y_MAP_pred: vector of length \(n\) containing a draw from the posterior distribution obtained using the maximum a posteriori estimates (MAP) of the parameters.
diagnostics: list containing 2 elements: CPO, i.e. the Conditional Predictive Ordinate (Gelfand et al. 1992); and LPML, i.e. the logarithm of the pseudo-marginal likelihood (Ibrahim et al. 2014).
newdata: if the matrix new_X of new values of the covariates is provided, list of three elements:
  • new_X: the provided matrix of explanatory variables;
  • y_newdata: a matrix of dimension \([\text{nrow(new_X)}, \text{iter}]\), each column is a draw from the posterior predictive distribution using new_X;
  • y_MAP_newdata: vector of length \(\text{nrow(new_X)}\) containing a draw from the posterior distribution obtained using the MAP estimate of the parameters, computed on the new data new_X.

perc_burnin: the component from object.

References


---

sample_bpr Fitting Bayesian Poisson Regression

Description

The function generates draws from the posterior distribution of the coefficients of Poisson regression models. The method allows for Gaussian and horseshoe (Carvalho et al, 2010) prior distributions, and relies on a Metropolis-Hastings or importance sampler algorithm.
Usage

```r
call = NULL,
data = NULL,
iter,
burnin = NULL,
prior = list(type = "gaussian", b = NULL, B = NULL, tau = NULL),
pars = list(method = "MH", max_dist = 50, max_r = NULL, max_dist_burnin = 1e+06),
state = NULL,
thin = 1,
verbose = TRUE,
seed = NULL,
nchains = 1,
perc_burnin = 0.25
```

Arguments

**formula**
- an object of class "formula": a symbolic description of the model to be fitted.

**data**
- data frame or matrix containing the variables in the model.

**iter**
- number of algorithm iterations.

**burnin**
- (optional) a positive integer specifying the length of the burn-in. If a value > 1 is provided, the first burnin iterations use a different tuning parameter in order to better explore the parameter space.

**prior**
- a named list of parameters to select prior type and parameters, with arguments:
  - **type**
    - string specifying whether an informative Gaussian ("gaussian") or a horseshoe ("horseshoe") prior should be used. Default is "gaussian".
  - **b, B**
    - (optional) if a Gaussian prior is used, the mean and covariance matrix passed as prior parameters. If not specified, the prior on the regression parameters is centered at zero, with independent N(0,2) components.
  - **tau**
    - if a horseshoe prior is used, the global shrinkage parameter tau has to be fixed. This can be seen as an empirical Bayes approach, and allows to speed convergence and avoid potential convergence issues that often occur when it is sampled. In general, the parameter can be interpreted as a measure of sparsity, and it should be fixed to small values. See van der Pas et al. (2017) for a discussion.

**pars**
- a named list of parameters to select algorithm type and tuning parameters, with arguments:
  - **method**
    - the type of algorithm used. Default is a Metropolis-Hastings algorithm ("MH"), the alternative is an importance sampler algorithm ("IS").
  - **max_dist**
    - tuning parameter controlling the "distance" of the approximation to the true target posterior. For the Metropolis-Hastings algorithm, it can be used to balance acceptance rate and autocorrelation of the chains. As a general indication, larger values are needed for increasing size/dimension of the data to obtain good results. #
• \texttt{max_r}: (optional) additional tuning parameter which sets an upper bound for the parameters \( r \) controlling the approximation.

• \texttt{max_dist_burnin}: if \texttt{burnin} is specified, the tuning parameter used for the first part of the chain. A very large value is sometimes useful to explore the parameter space (especially if the chains are initialized very far from their stationary distribution).

\texttt{state} \hspace{1cm} \text{optional vector providing the starting points of the chains.}

\texttt{thin} \hspace{1cm} \text{a positive integer specifying the period for saving samples. The default is 1.}

\texttt{verbose} \hspace{1cm} \text{logical (default = TRUE) indicating whether to print messages on the progress of the algorithm and possible convergence issues.}

\texttt{seed} \hspace{1cm} \text{(optional) positive integer: the seed of random number generator.}

\texttt{nchains} \hspace{1cm} \text{(optional) positive integer specifying the number of Markov chains. The default is 1.}

\texttt{perc_burnin} \hspace{1cm} \text{(default = 0.25) percentage of the chain to be discarded to perform inference. If both burnin and perc_burnin are specified, the most conservative burn-in is considered.}

\textbf{Details}

This function fits a Bayesian Poisson regression model with Gaussian prior distributions on the regression coefficients:

\[
Y \sim \text{Poisson}(\lambda), \quad \lambda = \exp(X\beta)
\]

where \( Y \) is a size \( n \) vector of counts and \( X \) is a \( n \times p \) matrix of coefficients; and \( (\beta|−) \) has a Gaussian distribution (possibly conditionally on some parameters).

Specifically, the function allows for informative Gaussian prior distribution on the parameters, i.e. \( (\beta_1, ..., \beta_p) \sim \text{N}(b, B) \), and for a horseshoe prior distribution (Carvalho et al, 2010). The horseshoe prior is a scale mixture of normals, which is typically used in high-dimension settings to induce sparsity and regularization of the coefficients.

The implemented Metropolis-Hastings and importance sampler exploit as proposal density a multivariate Gaussian approximation of the posterior distribution. Such proposal is based on the convergence of the negative binomial distribution to the Poisson distribution and on the Polya-gamma data augmentation of Polson et al. (2013).

The output of the sampling is an object of class \texttt{poisreg} and admits class-specific methods to perform inference. The function \texttt{summary.poisreg} can be used to obtain or print a summary of the results and of the algorithm diagnostics. The function \texttt{mcmc_diagnostics} can be used to obtain or print convergence diagnostics for the sampled chains. The function \texttt{plot.poisreg} prints the trace of the sampled values and a density estimate of the regression coefficients. See \texttt{plot.mcmc}.

The function \texttt{posterior_predictive} can be used to compute the posterior predictive distributions to check the model. See also the related function \texttt{plot.ppc}. 
Value

An object of S3 class poisreg containing the results of the sampling.

poisreg is a list containing at least the following elements:

- `sim`: list of the results of the sampling. It contains the following elements:
  - `beta`: mcmc object of posterior draws of the regression coefficients.
  - `r`: the sequence of adaptive tuning parameters used in each iteration.
  - `time`: the total amount of time to perform the simulation.

- `formula`: the formula object used.
- `data`: list with elements the matrix of covariates X and response variable y.
- `state`: the starting points of the chain.
- `burnin`: length of the used burn-in.
- `prior`: whether a Gaussian or horseshoe prior was used.
- `prior_pars`: prior parameters.
- `thin`: thinning frequency passed to the thin parameter.
- `nchains`: number of chains. If nchains was chosen >1, the output list will also include additional numbered `sim` elements, one for each sampled chain.
- `perc_burnin`: percentage of the chain used as burn-in.

References


See Also

`summary.poisreg`, `mcmc_diagnostics`, `plot.poisreg`, `merge_sim`, `posterior_predictive`

Examples

```
require(MASS) # load the data set
head(epil)

fit = sample_bpr( y ~ lbase*trt + lage + V4, data = epil,
                 iter = 1000)

summary(fit) # summary of posterior inference
mcmc_diagnostics(fit) # summary of MCMC convergence diagnostics
plot(fit)
```

## Examples with different options
# Select prior parameters and set tuning parameter
```r
tf2 = sample_bpr( y ~ lbase*trt + lage + V4, data = epil,
    iter = 1000,
    prior = list( type = "gaussian", b = rep(0, 6),
       B = diag(6) * 3 ),
    pars = list( max_dist = 10 ))
```

# Simulate multiple chains and merge outputs after checking convergence
```r
tf3 = sample_bpr( y ~ lbase*trt + lage + V4, data = epil,
    iter = 1000,
    nchains = 4, thin = 2)
```

# fit3 now contains additional elements ($sim2, $sim3, $sim4)
```r
mcmc_diagnostics(tf3)
```

# the Gelman-Rubin diagnostics confirms convergence of the 4
# independent chains to the same stationary distribution
```r
tf3b = merge_sim(tf3)
```

# fit 3b contains only one MCMC chain of length 1500
# (after thinning and burn-in)
```r
## introduce more variables and use regularization
epil2 <- epil[epil$period == 1, ]
epil2["period"] <- rep(0, 59); epil2["y"] <- epil2["base"]
epil["time"] <- 1; epil2["time"] <- 4
epil2 <- rbind(epil, epil2)
epil2$pred <- unclass(epil2$trt) * (epil2$period > 0)
epil2$subject <- factor(epil2$subject)
epil3 <- aggregate(epil2, list(epil2$subject, epil2$period > 0),
    function(x) if(is.numeric(x)) sum(x) else x[1])
epil3$pred <- factor(epil3$pred,
    labels = c("base", "placebo", "drug"))
contrasts(epil3$pred) <- structure(contr.sdif(3),
    dimnames = list(NULL, c("placebo-base", "drug-placebo")))
```

```r
fit4 = sample_bpr(y ~ pred + factor(subject), data = epil3,
    pars = list(max_dist = 0.3),
    prior = list(type = "horseshoe", tau = 2),
    iter = 3000, burnin = 1000)
```

```r
summary(fit4)
mcmc_diagnostics(fit4)
```

```r
plot(posterior_predictive(fit4), stats = c("mean", "sd", "max"))
```
**Summary**

This function is a method for class `poisreg`. It prints summary statistics and returns posterior estimates of regression quantities.

### Usage

```r
## S3 method for class 'poisreg'
summary(object, ...)  
## S3 method for class 'poisreg'
print(x, ...)
```

### Arguments

- **object**: object of class "poisreg" (usually, the result of a call to `sample_bpr`).
- **...**: further arguments passed to or from other methods.
- **x**: object of class "poisreg" (usually, the result of a call to `sample_bpr`).

### Details

The printed output of `summary.poisreg` summarizes the main quantities of the fit. The first component `Call` recapcs the type of prior and algorithm used.

**Coefficients** is a table of estimated quantities for the regression parameters. The first three columns report the estimated posterior mean, standard errors and medians. The last two columns correspond to the lower and upper bounds of the 0.95 credible intervals. If the credible interval does not include zero, a star is printed in correspondence of each parameter (similarly to the 'significance stars' of `summary.lm`). All the estimates are computed discarding the first part of the chain as burn-in (more details are printed in the `Algorithm` section).

**Algorithm** briefly summarizes the main diagnostics of convergence and efficiency of the algorithm. It prints the number of iterations actually used to obtain the estimates, after removing the burn-in and thinning. If a Metropolis-Hastings algorithm is used, the summary reports the acceptance rate, which is the most commonly used indicator to tune the performance of the algorithm, along with the mean effective sample size (averaged over all parameters). If the importance sampler is used, the summary only reports the effective sample size, which is computed as $\sum_t w_t^2 / (\sum_t w_t)^2$ (where $w_t$ is the sequence of weights) and is a measure of the efficiency of the sampler.

### Value

`summary.poisreg` returns a list with elements:

- **formula**: the component from `object`.
- **data**: list with elements the matrix of covariates `X` and response variable `y`.
- **prior**: `prior$type` from `object`.
- **prior_pars**: prior parameters from `object`.
- **coefficients**: the matrix of coefficient estimantes, standard errors and 95% credible intervals.
- **psi2**: if a horseshoe prior is selected, the estimate of the local shrinkage parameter.
len_burnin : the length of the burn-in used to compute the estimates.
effSize : the mean effective sample size of the chains used to compute the estimates.

Examples

# For examples see example(sample_bpr)
Index

effectiveSize, 2

gelman.diag, 2
geweke.diag, 2

mcmc, 9
mcmc_diagnostics, 2, 8, 9
merge_sim, 2, 3, 9

plot.mcmc, 4, 8
plot.poisreg, 2, 4, 8, 9
plot.posterior_check, 4
posterior_predictive, 4, 5, 5, 8, 9
print.poisreg (summary.poisreg), 10

sample_bpr, 2–4, 6, 6, 11
summary.lm, 11
summary.poisreg, 2, 8, 9, 10