Package ‘BayesLogit’

May 7, 2013

Version 0.2-2

Date 2013-05-06

Title Logistic Regression

Author Nicholas G. Polson, James G. Scott, and Jesse Windle

Maintainer Jesse Windle <jwindle@ices.utexas.edu>

Description The BayesLogit package does posterior simulation for binomial and multinomial logistic regression using the Polya-Gamma latent variable technique. This method is fully automatic, exact, and fast. A routine to efficiently sample from the Polya-Gamma class of distributions is included.

License GPL (>= 3)

Depends R (>= 2.14.0)

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-05-07 19:03:24

R topics documented:

`compute.mixture` *Compute Mixture*

Description

Compute the means, variances, and probabilities to approximate a -log(Ga) distribution or a type III logistic distribution.

Usage

```
compute.mixture(shape, type=c("log.gamma", "logistic.iii"))
```

Arguments

`shape`	The shape parameter.
`type`	The type of distribution to approximate, either a -log(Ga(shape,1)) distribution or a type III logistic(shape) distribution.

Details

Fruhwirth-Schnatter et al. use tables of discrete mixtures of normals to approximate entire families of distributions. These approximations are useful for data augmentation techniques that lead to convenient posterior simulation for certain classes of generalized linear models. In particular, approximations to the -log(Ga) distribution and the type III logistic distribution are useful for negative binomial regression and for binomial logistic regression respectively.

This function generates the means, variances, and probabilities that approximate either a -log(Ga) or type III logistic distribution.

The code that produces these values can be found in the functions `compute.mixture.lg` and `compute.mixture.l3`, which cannot be called directly. `compute.mixture.lg` is an R translation of MATLAB code found in the `bayesf` package. `compute.mixture.l3` is taken from the R package **binomlogit**. The function `compute.mixture.l3` returns a list without an array of means, since they are identically zero.

Value

Returns a list with components `m`, `v`, `p` representing the mean, variance, and probability of the specific discrete mixture of normals.

References

Agnes Fussl. binomlogit: Efficient MCMC for Binomial Logit Models (2012).

Sylvia Fruhwirth-Schnatter. bayesf. URL <http://statmath.wu.ac.at/~fruehwirth/monographie/> (2007).

Sylvia Fruhwirth-Schnatter, Rudolf Fruhwirth, Leonhard Held, Havard Rue. Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data. Statistics and Computing (2009).

Sylvia Fruhwirth-Schnatter and Rudolf Fruhwirth. Data Augmentation and MCMC for Binary and Multinomial Logit Models. Statistical Modelling and Regression Structures (2010), Springer-Verlag.

Agnes Fussl, Sylvia Fruhwirth-Schnatter and Rudolf Fruhwirth. Efficient MCMC for Binomial Logit Models (2013).

See Also

`draw.indicators`

Examples

```
## Approximate - log[Ga] using mixture of normals.
sp = 9
nm = compute.mixture(sp, "log.gamma")
nc = length(nm$m)

r  = sample.int(nc, 10, replace=TRUE, prob=nm$p)
e  = rnorm(10, nm$m[r], sqrt(nm$v[r]))

r.post = draw.indicators(e, nm)

## Approximate type III logistic using mixture of normals.
sp = 9
nm = compute.mixture(sp, "logistic.iii")
nc = length(nm$m)

r  = sample.int(nc, 10, replace=TRUE, prob=nm$p)
e  = rnorm(10, nm$m[r], sqrt(nm$v[r]))

r.post = draw.indicators(e, nm)
```

`draw.indicators` *Draw Indicators*

Description

Draw indicator variables from a normal mixture.

Usage

```
draw.indicators(res, nmix)
draw.indicators.C(res, nmix)
draw.indicators.R(res, nmix)
```

Arguments

`res`	A one-dimensional array of residuals.
`nmix`	A list representing the mixture of normals with components `m`, `v`, `p`, representing the arrays of means, variances, weights of the normal mixture.

Details

Suppose $e_i \sim N(m_{\gamma_i}, v_{\gamma_i})$. Then, given e_i, draw γ_i from $P(\gamma_i = j | e_i)$ for each i.

Value

Returns a one-dimensional array of mixture component identifiers.

See Also

`compute.mixture`

Examples

```
## Approximate - log[Ga] using mixture of normals.
sp = 9
nm = compute.mixture(sp, "log.gamma")
nc = length(nm$m)

r  = sample.int(nc, 10, replace=TRUE, prob=nm$p)
e  = rnorm(10, nm$m[r], sqrt(nm$v[r]))

r.post = draw.indicators(e, nm)

## Approximate type III logistic using mixture of normals.
sp = 9
nm = compute.mixture(sp, "logistic.iii")
nc = length(nm$m)

r  = sample.int(nc, 10, replace=TRUE, prob=nm$p)
e  = rnorm(10, nm$m[r], sqrt(nm$v[r]))

r.post = draw.indicators(e, nm)
```

`logit` *Default Bayesian Logistic Regression*

Description

Run a Bayesian logistic regression.

Usage

```
logit(y, X, n=rep(1,length(y)),
      m0=rep(0, ncol(X)), P0=matrix(0, nrow=ncol(X), ncol=ncol(X)),
      samp=1000, burn=500)
```

Arguments

`y`	An N dimensional vector; y_i is the average response at x_i.
`X`	An N x P dimensional design matrix; x_i is the ith row.
`n`	An N dimensional vector; n_i is the number of observations at each x_i.
`m0`	A P dimensional prior mean.
`P0`	A PxP dimensional prior precision.
`samp`	The number of MCMC iterations saved.
`burn`	The number of MCMC iterations discarded.

Details

Logistic regression is a classification mechanism. Given the binary data $\{y_i\}$ and the p-dimensional predictor variables $\{x_i\}$, one wants to forecast whether a future data point y* observed at the predictor x* will be zero or one. Logistic regression stipulates that the statistical model for observing a success=1 or failure=0 is governed by

$$P(y^* = 1|x^*, \beta) = (1 + \exp(-x^* \beta))^{-1}.$$

Instead of representing data as a collection of binary outcomes, one may record the average response y_i at each unique x_i given a total number of n_i observations at x_i. We follow this method of encoding data.

Value

`logit` returns a list.

`beta`	A samp x P array; the posterior sample of the regression coefficients.
`w`	A samp x N' array; the posterior sample of the latent variable. WARNING: N' may be less than N if data is combined.
`y`	The response matrix–different than input if data is combined.
`X`	The design matrix–different than input if data is combined.
`n`	The number of samples at each observation–different than input if data is combined.

References

Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using Polya-Gamma latent variables. http://arxiv.org/abs/1205.0310

Nicholas Poslon and James G. Scott. Default Bayesian analysis for multi-way tables: a data-augmentation approach. http://arxiv.org/pdf/1109.4180

See Also

rpg, logit.EM, mlogit

Examples

```
## From UCI Machine Learning Repository.
data(spambase);

## A subset of the data.
sbase = spambase[seq(1,nrow(spambase),10),];

X = model.matrix(is.spam ~ word.freq.free + word.freq.1999, data=sbase);
y = sbase$is.spam;

## Run logistic regression.
output = logit(y, X, samp=1000, burn=100);
```

---

logit.combine *Collapse Data for Binomial Logistic Regression*

Description

Collapse data for binomial logistic regression.

Usage

```
logit.combine(y, X, n=rep(1,length(y)))
```

Arguments

y	An N dimensional vector; y_i is the average response at x_i.
X	An N x P dimensional design matrix; x_i is the ith row.
n	An N dimensional vector; n_i is the number of observations at each x_i.

Details

Logistic regression is a classification mechanism. Given the binary data $\{y_i\}$ and the p-dimensional predictor variables $\{x_i\}$, one wants to forecast whether a future data point y* observed at the predictor x* will be zero or one. Logistic regression stipulates that the statistical model for observing a success=1 or failure=0 is governed by

$$P(y^* = 1 | x^*, \beta) = (1 + \exp(-x^* \beta))^{-1}.$$

Instead of representing data as a collection of binary outcomes, one may record the average response y_i at each unique x_i given a total number of n_i observations at x_i.

Thus, when a predictor is repeated the two reponses may be collapsed into a single observation representing multiple trials. This function collapses data in this way.

Value

`logit.combine` returns a list.

`y`	The new response.
`X`	The new design matrix.
`n`	The number of samples at each revised observation.

See Also

`logit`, `logit.EM`, `mlogit`

Examples

```
## From UCI Machine Learning Repository.
data(spambase);

## A subset of the data.
sbase = spambase[seq(1,nrow(spambase),10),];

X = model.matrix(is.spam ~ word.freq.free + word.freq.1999, data=sbase);
y = sbase$is.spam;

## Actually unnecessary as logit.EM automatically tries to compress.
new.data = logit.combine(y, X)
mode.spam = logit.EM(new.data$y, new.data$X, new.data$n)
mode.spam
```

`logit.EM` *Logistic Regression Expectation Maximization*

Description

Expectation maximization for logistic regression.

Usage

```
logit.EM(y, X, n=rep(1,length(y)), tol=1e-9, max.iter=100)
```

Arguments

`y`	An N dimensional vector; y_i is the average response at x_i.
`X`	An N x P dimensional design matrix; x_i is the ith row.
`n`	An N dimensional vector; n_i is the number of observations at each x_i.
`tol`	Threshold at which algorithm stops.
`max.iter`	Maximum number of iterations.

Details

Logistic regression is a classification mechanism. Given the binary data $\{y_i\}$ and the p-dimensional predictor variables $\{x_i\}$, one wants to forecast whether a future data point y* observed at the predictor x* will be zero or one. Logistic regression stipulates that the statistical model for observing a success=1 or failure=0 is governed by

$$P(y^* = 1 | x^*, \beta) = (1 + \exp(-x^* \beta))^{-1}.$$

Instead of representing data as a collection of binary outcomes, one may record the average response y_i at each unique x_i given a total number of n_i observations at x_i. We follow this method of encoding data.

A non-informative prior is used.

Value

`beta`	The posterior mode.
`iter`	The number of iterations.

References

Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using Polya-Gamma latent variables. http://arxiv.org/abs/1205.0310

Nicholas G. Poslon and James G. Scott. Default Bayesian analysis for multi-way tables: a data-augmentation approach. http://arxiv.org/pdf/1109.4180

See Also

`rpg`, `logit`, `mlogit`

Examples

```
## From UCI Machine Learning Repository.
data(spambase);

## A subset of the data.
sbase = spambase[seq(1,nrow(spambase),10),];

X = model.matrix(is.spam ~ word.freq.free + word.freq.1999, data=sbase);
```

```
y = sbase$is.spam;

## Run logistic regression.
output = logit.EM(y, X);
```

---

mlogit *Bayesian Multinomial Logistic Regression*

Description

Run a Bayesian multinomial logistic regression.

Usage

```
mlogit(y, X, n=rep(1,nrow(as.matrix(y))),
       m.0=array(0, dim=c(ncol(X), ncol(y))),
       P.0=array(0, dim=c(ncol(X), ncol(X), ncol(y))),
       samp=1000, burn=500)
```

Arguments

`y`	An N x J-1 dimensional matrix; y_{ij} is the average response for category j at x_i.
`X`	An N x P dimensional design matrix; x_i is the ith row.
`n`	An N dimensional vector; n_i is the total number of observations at each x_i.
`m.0`	A P x J-1 matrix with the $beta_j$'s prior means.
`P.0`	A P x P x J-1 array of matrices with the $beta_j$'s prior precisions.
`samp`	The number of MCMC iterations saved.
`burn`	The number of MCMC iterations discarded.

Details

Multinomial logistic regression is a classifiction mechanism. Given the multinomial data $\{y_i\}$ with J categories and the p-dimensional predictor variables $\{x_i\}$, one wants to forecast whether a future data point y* at the predictor x*. Multinomial Logistic regression stiuplates that the statistical model for observing a draw category j after rolling the multinomial die $n^* = 1$ time is governed by

$$P(y^* = j | x^*, \beta, n^* = 1) = e^{x^* \beta_j} / \sum_{k=1}^{J} e^{x^* \beta_k}.$$

Instead of representing data as the total number of responses in each category, one may record the average number of responses in each category and the total number of responses n_i at x_i. We follow this method of encoding data.

We assume that $\beta_J = 0$ for purposes of identification!

You may use mlogit for binary logistic regression with a normal prior.

Value

mlogit returns a list.

beta	A samp x P x J-1 array; the posterior sample of the regression coefficients.
w	A samp x N' x J-1 array; the posterior sample of the latent variable. WARNING: N' may be less than N if data is combined.
y	The response matrix–different than input if data is combined.
X	The design matrix–different than input if data is combined.
n	The number of samples at each observation–different than input if data is combined.

References

Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using Polya-Gamma latent variables. http://arxiv.org/abs/1205.0310

See Also

rpg, logit.EM, logit

Examples

```
## Use the iris dataset.
data(iris)
N = nrow(iris)
P = ncol(iris)
J = nlevels(iris$Species)

X     = model.matrix(Species ~ ., data=iris);
y.all = model.matrix(~ Species - 1, data=iris);
y     = y.all[,-J];

out = mlogit(y, X, samp=1000, burn=100);
```

---

rain *Tokyo Rainfall*

Description

The rain data has 366 real valued binomial responses representing days on which it rained over the course of two years in Tokyo.

Format

A list with three components: `y`, the repsonse representing the number of times it has rained on the ith day of the year in Tokyo from 1983 through 1984, `X`, a matrix of ones, and `n`, the number of observations for each day. `n` is two for all days except Feb. 29.

Details

This is the infamous Tokyo rainfall data from Kitagawa (1987).

References

Genshiro Kitagawa. Non-Gaussian State-Space Modeling of Non-stationary Time Series. Journal of the American Statistical Association (1987).

`rks` *The Kolmogorov-Smirnov distribution*

Description

Generate a random variate from the Kolmogorov-Smirnov distribution.

This is not directly related to the Polya-Gamma technique, but it is a nice example of using an alternating sum to generate a random variate.

Usage

```
rks(N=1)
```

Arguments

`N`	The number of random variates to generate.

Details

The density function of the KS distribution is

$$f(x) = 8 \sum_{i=1}^{\infty} (-1)^{n+1} n^2 x e^{-2n^2x^2}.$$

We follow Devroye (1986) p. 161 to generate random draws from KS.

References

L. Devroye. Non-Uniform Random Variate Generation, 1986.

Examples

```
X = rks(1000)
```

`rpg` *The Polya-Gamma Distribution*

Description

Generate a random variate from the Polya-Gamma distribution.

Usage

```
rpg(num=1, h=1, z=0.0)

rpg.gamma(num=1, h=1, z=0.0, trunc=200)

rpg.devroye(num=1, n=1, z=0.0)

rpg.alt(num=1, h=1, z=0.0)

rpg.sp(num=1, h=1, z=0.0, track.iter=FALSE)
```

Arguments

	You may call rpg when n and z are vectors.
	The number of random variates to simulate.
`num`	Shape parameter, a positive integer
`h`	Shape parameter. codeh >= 1 if not using sum of gammas method.
`z`	Parameter associated with tilting.
`trunc`	The number of elements used in sum of gammas approximation.
`track.iter`	The number of proposals made before accepting.

Details

A random variable X with distribution PG(n,z) is generated by

$$X \sim \sum_{k=1}^{\infty} G(n,1)/(2\pi^2(k-1/2)^2 + z^2/2).$$

The density for X may be derived from Z and PG(n,0) as

$$p(x|n,z) \propto \exp(-xz^2/2)p(x|n,0).$$

Thus PG(n,z) is an exponentially tilted PG(n,0).

Two different methods for generating this random variable are implemented. In general, you may use rpg.gamma to generate an approximation of PG(n,z) using the sum of Gammas representation above. When n is a natural number you may use `rpg.devroye` to sample PG(n,z). The later method is fast.

Value

This function returns `num` Polya-Gamma samples.

References

Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using Polya-Gamma latent variables. `http://arxiv.org/abs/1205.0310`

See Also

`logit.EM`, `logit`, `mlogit`

Examples

```
h = c(1, 2, 3);
z = c(4, 5, 6);

## If h contains small integers, e.g. {1,2,3}, use Devroye-like method.
X = rpg.devroye(100, h, z);

h = c(1.2, 2.3, 3.2);
z = c(4, 5, 6);

## Else if h contains scalars >=1 or larger integers use alternate Devroye-like method.
X = rpg(100, h, z);

## Else sum of gammas method -- this is slow.
X = rpg.gamma(100, h, z);
```

`spambase` *Spambase Data*

Description

The spambase data has 57 real valued explanatory variables which characterize the contents of an email and and one binary response variable indicating if the email is spam. There are 4601 observations.

Format

A data frame: the first column is a binary response variable indicating if the email is spam. The remaining 57 columns are real valued explanatory variables.

Details

Of the 57 explanatory variables, 48 describe word frequency, 6 describe character frequency, and 3 describe sequences of capital letters.

word.freq.<word> A continuous explanatory variable describing the frequency with which the word <word> appears; measured in percent.

char.freq.<char> A continuous explanatory variable describing the frequency with which the character <char> appears; measured in percent.

capital.run.length.<oper> A statistic involving the length of consecutive capital letters.

Use `names` to see the specific words, characters, or statistics for each respective class of variable.

References

Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt of Hewlett-Packard Labs (1999). Spambase Data Set. `http://archive.ics.uci.edu/ml/datasets/Spambase`

Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Index