# Package 'brm'

October 12, 2022

**Type** Package

**Title** Binary Regression Model

**Version** 1.1.1

**Date** 2020-06-16

**Author** Linbo Wang, Mark Clements, Thomas Richardson

**Maintainer** Mark Clements <mark.clements@ki.se>

**Description** Fits novel models for the conditional relative risk, risk difference and odds ratio <doi:10.1080/01621459.2016.1192546>.

**License** MIT + file LICENSE

**RoxygenNote** 7.0.2

**NeedsCompilation** no

**Suggests** knitr, rmarkdown

**Imports** stats

**URL** http://github.com/mclements/brm

**BugReports** http://github.com/mclements/brm/issues

**Repository** CRAN

**Date/Publication** 2020-06-17 09:50:09 UTC

## R topics documented:

brm-package . . . . . 2
brm . . . . . 3
getProbRD . . . . . 6
getProbRR . . . . . 7
getProbScalarRD . . . . . 7
getProbScalarRR . . . . . 8
predict.brm . . . . . 9
print.brm . . . . . 10

**Index** **11**

brm-package *Fitting Binary Regression Models*

## Description

The function brm in this package provides an alternative to generalized linear models for fitting binary regression models, in which both the response $y$ and the primary exposure of interest $x$ are binary. This is especially useful if the interest lies in estimating the association between $x$ and $y$, and how that association varies as a function of (other) covariates $v$.

Unlike glm, which uses a single link function for the outcome, brm separates the nuisance model from the target model. This separation provides opportunities to choose nuisance models independently of the target model. To see why this is important, we may contrast it with the use of a GLM to model the log relative risk. In this setting one might use a Poisson regression (with interaction term) $logP(y = 1|x, va, vb) = \alpha * x * va + \beta * vb$ (though such a model ignores the fact that $y$ is binary); here $va$ and $vb$ are subsets of $v$. Such a Poisson model can be seen as a combination of two parts: a target model $logRR(va) = \alpha * va$ and a nuisance model $logP(y = 1|x = 0, vb) = \beta * vb$. However, this nuisance model is variation dependent of the target model so that predicted probabilities may go outside of $[0, 1]$. Furthermore, one cannot solve this problem under a GLM framework as with a GLM, the target model and nuisance model are determined *simultaneously* through a link function.

More specifically, if the target model is a linear model on the conditional log Relative Risk (log RR) or ('logistically' transformed) conditional Risk Difference (atanh RD), brm fits a linear nuisance model for the conditional log Odds Product (log OP). If the target model is a linear model on the conditional log Odds Ratio (log OR), brm fits a linear nuisance model on the conditional logit baseline risk, logit P(y = 1|x = 0, vb). Note in this case the target and nuisance models combine to form a simple logistic regression model (which is fitted using glm).

brm fits the three target models described above as they are simple and the parameter space is unconstrained. brm fits the nuisance models above as they are variation independent of the corresponding target model. This variation independence greatly facilitates parameter estimation and interpretation.

brm also provides doubly robust fitting as an option such that the estimates for $\alpha$ are still consistent and asymptotically normal even when the nuisance model is misspecified, provided that we have a correctly specified logistic model for the exposure probability $P(x = 1|v)$. Such doubly robust estimation is only possible for the Relative Risk and Risk Difference, but not the Odds Ratio.

See Richardson et al. (2017) for more details.

## References

Thomas S. Richardson, James M. Robins and Linbo Wang. "On Modeling and Estimation for the Relative Risk and Risk Difference." Journal of the American Statistical Association: Theory and Methods (2017).

brm *Fitting Binary Regression Models*

## Description

brm is used to estimate the association between two binary variables, and how that varies as a function of other covariates.

## Usage

```
brm(
  y,
  x,
  va,
  vb = NULL,
  param,
  est.method = "MLE",
  vc = NULL,
  optimal = TRUE,
  weights = NULL,
  subset = NULL,
  max.step = NULL,
  thres = 1e-08,
  alpha.start = NULL,
  beta.start = NULL,
  message = FALSE
)
```

## Arguments

| | |
|---|---|
| y | The response vector. Should only take values 0 or 1. |
| x | The exposure vector. Should only take values 0 or 1. |
| va | The covariates matrix for the target model. It can be specified via an object of class "formula" or a matrix. In the latter case, no intercept terms will be added to the covariates matrix. |
| vb | The covariates matrix for the nuisance model. It can be specified via an object of class "formula" or a matrix. In the latter case, no intercept terms will be added to the covariates matrix. (If not specified, defaults to va.) |
| param | The measure of association. Can take value 'RD' (risk difference), 'RR' (relative risk) or 'OR' (odds ratio) |
| est.method | The method to be used in fitting the model. Can be 'MLE' (maximum likelihood estimation, the default) or 'DR' (doubly robust estimation). |
| vc | The covariates matrix for the probability of exposure, often called the propensity score. It can be specified via an object of class "formula" or a matrix. In the latter case, no intercept terms will be added to the covariates matrix. By default |

| | |
|---|---|
| | we fit a logistic regression model for the propensity score. (If not specified, defaults to va.) |
| optimal | Use the optimal weighting function for the doubly robust estimator? Ignored if the estimation method is 'MLE'. The default is TRUE. |
| weights | An optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. |
| subset | An optional vector specifying a subset of observations to be used in the fitting process. |
| max.step | The maximal number of iterations to be passed into the optim function. The default is 1000. |
| thres | Threshold for judging convergence. The default is 1e-6. |
| alpha.start | Starting values for the parameters in the target model. |
| beta.start | Starting values for the parameters in the nuisance model. |
| message | Show optimization details? Ignored if the estimation method is 'MLE'. The default is FALSE. |

## Details

brm contains two parts: the target model for the dependence measure (RR, RD or OR) and the nuisance model; the latter is required for maximum likelihood estimation. If param="RR" then the target model is $logRR(va) = \alpha * va$. If param="RD" then the target model is $atanhRD(va) = \alpha * va$. If param="OR" then the target model is $logOR(va) = \alpha * va$. For RR and RD, the nuisance model is for the log Odds Product: $logOP(vb) = \beta * vb$. For OR, the nuisance model is for the baseline risk: $logit(P(y = 1|x = 0, vb)) = \beta * vb$. In each case the nuisance model is variation independent of the target model, which ensures that the predicted probabilities lie in $[0, 1]$. See Richardson et al. (2016+) for more details.

If est.method="DR" then given a correctly specified logistic regression model for the propensity score $logit(P(x = 1|vc)) = \gamma * vc$, estimation of the RR or RD is consistent, even if the log Odds Product model is misspecified. This estimation method is not available for the OR. See Tchetgen Tchetgen et al. (2014) for more details.

When estimating RR and RD, est.method="DR" is recommended unless it is known that the log Odds Product model is correctly specified. Optimal weights (optimal=TRUE) are also recommended to increase efficiency.

For the doubly robust estimation method, MLE is used to obtain preliminary estimates of $\alpha$, $\beta$ and $\gamma$. The estimate of $\alpha$ is then updated by solving a doubly-robust estimating equation. (The estimate for $\beta$ remains the MLE.)

## Value

A list consisting of

| | |
|---|---|
| param | the measure of association. |
| point.est | the point estimates. |
| se.est | the standard error estimates. |
| cov | estimate of the covariance matrix for the estimates. |

| | |
|---|---|
| conf.lower | the lower limit of the 95% (marginal) confidence interval. |
| conf.upper | the upper limit of the 95% (marginal) confidence interval. |
| p.value | the two sided p-value for testing zero coefficients. |
| coefficients | the matrix summarizing key information: point estimate, 95% confidence interval and p-value. |
| param.est | the fitted RR/RD/OR. |
| va | the matrix of covariates for the target model. |
| vb | the matrix of covariates for the nuisance model. |
| converged | Did the maximization process converge? |

## Author(s)

Linbo Wang <linbo.wang@utoronto.ca>, Mark Clements <mark.clements@ki.se>, Thomas Richardson <thomasr@uw.edu>

## References

Thomas S. Richardson, James M. Robins and Linbo Wang. "On Modeling and Estimation for the Relative Risk and Risk Difference." Journal of the American Statistical Association: Theory and Methods (2017).

Eric J. Tchetgen Tchetgen, James M. Robins and Andrea Rotnitzky. "On doubly robust estimation in a semiparametric odds ratio model." Biometrika 97.1 (2010): 171-180.

## See Also

getProbScalarRD, getProbRD (vectorised), getProbScalarRR and getProbRR (vectorised) for functions calculating risks P(y=1|x=1) and P(y=1|x=0) from (atanh RD, log OP) or (log RR, log OP);

predict.blm for obtaining fitted probabilities from brm fits.

## Examples

```
set.seed(0)
n           = 100
alpha.true  = c(0,-1)
beta.true   = c(-0.5,1)
gamma.true  = c(0.1,-0.5)
params.true = list(alpha.true=alpha.true, beta.true=beta.true,
     gamma.true=gamma.true)
v.1         = rep(1,n)       # intercept term
v.2         = runif(n,-2,2)
v           = cbind(v.1,v.2)
pscore.true = exp(v %*% gamma.true) / (1+exp(v %*% gamma.true))
p0p1.true   = getProbRR(v %*% alpha.true,v %*% beta.true)
x           = rbinom(n, 1, pscore.true)
pA.true       = p0p1.true[,1]
pA.true[x==1] = p0p1.true[x==1,2]
y = rbinom(n, 1, pA.true)
```

```
fit.mle = brm(y,x,v,v,'RR','MLE',v,TRUE)
fit.drw = brm(y,x,v,v,'RR','DR',v,TRUE)
fit.dru = brm(y,x,v,v,'RR','DR',v,FALSE)

fit.mle2 = brm(y,x,~v.2, ~v.2, 'RR','MLE', ~v.2,TRUE) # same as fit.mle
```

---

`getProbRD` *Calculate risks from arctanh RD and log OP (vectorised)*

---

## Description

Calculate risks from arctanh RD and log OP (vectorised)

## Usage

```
getProbRD(atanhrd, logop)
```

## Arguments

| | |
|---|---|
| `atanhrd` | arctanh of risk difference |
| `logop` | log of odds product |

## Details

The $logOP$ is defined as $logOP = log[(P(y = 1|x = 0)/P(y = 0|x = 0)) * (P(y = 1|x = 1)/P(y = 0|x = 1))]$. The inverse hyperbolic tangent function `arctanh` is defined as $arctanh(z) = [log(1 + z) - log(1 - z)]/2$.

## Value

a matrix $(P(y = 1|x = 0), P(y = 1|x = 1))$ with two columns

## Examples

```
getProbRD(0,0)

set.seed(0)
logrr = rnorm(10,0,1)
logop = rnorm(10,0,1)
probs = getProbRD(logrr, logop)
colnames(probs) = c("P(y=1|x=0)","P(y=1|x=1)")
probs
```

## getProbRR *Calculate risks from log RR and log OP (vectorised)*

### Description

Calculate risks from log RR and log OP (vectorised)

### Usage

```
getProbRR(logrr, logop = NA)
```

### Arguments

| | |
|---|---|
| logrr | log of relative risk |
| logop | log of odds product |

### Details

The $logOP$ is defined as $logOP = log[(P(y = 1|x = 0)/P(y = 0|x = 0)) * (P(y = 1|x = 1)/P(y = 0|x = 1))]$.

### Value

a matrix $(P(y = 1|x = 0), P(y = 1|x = 1))$ with two columns

### Examples

```
getProbRR(0,0)

set.seed(0)
logrr = rnorm(10,0,1)
logop = rnorm(10,0,1)
probs = getProbRR(logrr, logop)
colnames(probs) = c("P(y=1|x=0)","P(y=1|x=1)")
probs
```

## getProbScalarRD *Calculate risks from arctanh RD and log OP*

### Description

Calculate risks from arctanh RD and log OP

### Usage

```
getProbScalarRD(atanhrd, logop)
```

### Arguments

| | |
|---|---|
| atanhrd | arctanh of risk difference |
| logop | log of odds product |

### Details

The $logOP$ is defined as $logOP = log[(P(y = 1|x = 0)/P(y = 0|x = 0)) * (P(y = 1|x = 1)/P(y = 0|x = 1))]$. The inverse hyperbolic tangent function arctanh is defined as $arctanh(z) = [log(1 + z) - log(1 - z)]/2$.

### Value

a vector $(P(y = 1|x = 0), P(y = 1|x = 1))$

### Examples

```
getProbScalarRD(0,0)

set.seed(0)
logrr = rnorm(10,0,1)
logop = rnorm(10,0,1)
probs = mapply(getProbScalarRD, logrr, logop)
rownames(probs) = c("P(y=1|x=0)","P(y=1|x=1)")
probs
```

---

## getProbScalarRR *Calculate risks from log RR and log OP*

---

### Description

Calculate risks from log RR and log OP

### Usage

```
getProbScalarRR(logrr, logop = NA)
```

### Arguments

| | |
|---|---|
| logrr | log of relative risk |
| logop | log of odds product |

### Details

The $logOP$ is defined as $logOP = log[(P(y = 1|x = 0)/P(y = 0|x = 0)) * (P(y = 1|x = 1)/P(y = 0|x = 1))]$.

## Value

a vector $(P(y = 1|x = 0), P(y = 1|x = 1))$

## Examples

```
getProbScalarRR(0,0)

set.seed(0)
logrr = rnorm(10,0,1)
logop = rnorm(10,0,1)
probs = mapply(getProbScalarRR, logrr, logop)
rownames(probs) = c("P(y=1|x=0)","P(y=1|x=1)")
probs
```

---

`predict.brm` *Fitted probabilities from* `brm` *fits*

---

## Description

Calculate fitted probabilities from a fitted binary regression model object.

## Usage

```
## S3 method for class 'brm'
predict(object, x.new = NULL, va.new = NULL, vb.new = NULL, ...)
```

## Arguments

| | |
|---|---|
| `object` | A fitted object from function `brm`. |
| `x.new` | An optional vector of x. |
| `va.new` | An optional covariate matrix to make predictions with. If omitted, the original matrix va is used. |
| `vb.new` | An optional covariate matrix to make predictions with. If vb.new is omitted but va.new is not, then vb.new is set to be equal to va.new. If both vb.new and va.new are omitted, then the original matrix vb is used. |
| `...` | affecting the predictions produced. |

## Value

If x.new is omitted, a matrix consisting of fitted probabilities for p0 = P(y=1|x=0,va,vb) and p1 = P(y=1|x=1,va,vb).

If x.new is supplied, a vector consisting of fitted probabilities px = P(y=1|x=x.new,va,vb).

print.brm *Ancillary function for printing*

## Description

Ancillary function for printing

## Usage

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

## Arguments

| | |
|---|---|
| x | a list obtained with the function 'brm' |
| ... | additional arguments affecting the output |

# Index

brm, 3
brm-package, 2

getProbRD, 6
getProbRR, 7
getProbScalarRD, 7
getProbScalarRR, 8

optim, *4*

predict.brm, 9
print.brm, 10