Package ‘CPMCGLM’

February 19, 2015

Version 1.1
Date 2012-03-20
Title Correction of the p-value after multiple coding
Author Jeremie Riou, Amadou Diakite, and Benoit Liquet
Maintainer Jeremie Riou <jeremie.riou@isped.u-bordeaux2.fr>
LazyLoad yes
Description We propose to determine the correction of the significance level after multiple coding of an explanatory variable in Generalized Linear Model. The different methods of correction of the p-value are the Single step Bonferroni procedure, and resampling based methods developed by P.H. Westfall in 1993. Resampling methods are based on the permutation and the parametric bootstrap procedure. If some continuous, and dichotomous transformations are performed this package offers an exact correction of the p-value developed by B. Liquet & D. Commenges in 2005. The naive method with no correction is also available.
License GPL (> 2)
Depends R (>= 2.10.0), mvtnorm, plyr
URL http://www.r-project.org
Repository CRAN
NeedsCompilation no
Date/Publication 2013-11-06 23:09:38

R topics documented:

CPMCGLM ................................................................. 2
data_sim ................................................................. 6
ForInternalUse ......................................................... 7
print.CPMCGLM ....................................................... 7
summary.CPMCGLM .................................................. 8

Index 9
**CPMCGLM**

Correction of the significance level after multiple coding of an explanatory variable in generalized linear model.

**Description**

We propose to determine the correction of the significance level after multiple coding of an explanatory variable in Generalized Linear Model. The different methods of correction of the p-value are the Single step Bonferroni procedure, and resampling based methods developed by P.H. Westfall in 1993. Resampling methods are based on the permutation and the parametric bootstrap procedure. If some continuous, and dichotomous transformations are performed this package offers an exact correction of the p-value developed by B. Liquet & D. Commenges in 2005. The naive method with no correction is also available.

**Usage**

```r
CPMCGLM(formula, family, link, data, varcod, dicho, nb.dicho, categ, nb.categ, boxcox, nboxcox, N=1000, cutpoint)
```

**Arguments**

- **formula**: an object of class "formula" : a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
- **family**: a description of the error distribution to be used in the model. This should be a character string naming a family function. The possible family functions are: "binomial", "gaussian", and "poisson".
- **link**: a description of the link function to be used in the model. This needs to be a character string naming a link function. For the "gaussian" family, you must use the "identity" link. For the "binomial" family function, "logit" and "probit" link are available. And for "Poisson" family function, you must use the "log" link function.
- **data**: a data frame containing the variables of the model.
- **varcod**: a continuous variable that you want to transform.
- **dicho**: a vector with the order of the quantile which are used for computing the cutpoint of each dichotomous transformation. The length of the vector corresponds to the number of transformation. If you specify this argument, "nb.dicho" must not be present.
- **nb.dicho**: if you do not enter the "dicho" argument, you can enter the number of dichotomous transformations that you want. The strategy of coding is presented in "Details" section.
- **categ**: a matrix with the order of quantile which are used for computing the categorical cutpoints of each transformation. The details of the "categ" specification are given under "Details". If you specify this argument, "nb.categ" must not be present.
if you do not enter the "categ" argument, you can enter the number of categorical
transformations that you want. The strategy of coding is presented in "Details"
section.

a vector of $\lambda$ parameters corresponding to each BoxCox transformation.

if you do not enter the "boxcox" argument, you can enter the number of box-
cox transformations that you want. The maximum number of transformations
that you can enter is 5. For the stage of coding, it seems natural to try the
crude variable ($\lambda_1=1$) and the log transformation. When two transformations
are considered we propose $\lambda_1 =1$ and $\lambda_2 =0$. Further interesting transforma-
tions include the square $\lambda_3 =2$ and the square root $\lambda_4 =0.5$. Finally, the power
$\lambda_5 =3/2$ may be tried.

the number of resampling that you want to do.

a matrix with the different numeric values for the cutpoints. The details of the
cutpoint specification are given under 'Details'.

Details

- formula: A typical predictor has the form "response ~ terms" where "response" is the numeric
response (possibly binary "0","1") vector and "terms" is a serie of terms which specifies a linear
predictor for response.

- nb.dicho: Dichotomous transformations include only the categorical transformations in two classes.
The most natural method is to use a transformation based on the quantile. For one transformation,
the median is used as a cutpoint for the dichotomous coding. For two transformations, the first
tercile is used for the first dichotomous transformation, and the second tercile for the second one,
and so on.

- categ: The categ argument needs to be a matrix. You need to have one line per transformation.
Therefore, the dimension of the matrix is $nbq \times maxq$, where $nbq$ is the number of transformations
tried with the categ transformations, and maxq is the maximum of number of quantiles that is used
in one quantile transformation.

For example:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1.]</td>
<td>0.33</td>
<td>0.66</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>[2.]</td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
<td>NA</td>
</tr>
<tr>
<td>[3.]</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
</tr>
</tbody>
</table>

In this example, three transformations are performed, so $nbq=3$. And maxq=4, because the max-
imum of number of quantiles that we used for the quintiles is 4. The first transformation leads
to a categorical transformation in three classes, with cutpoints at the first and the second tercile.
The second transformation allows to obtain a categorical variable in four classes with cutpoints at
the quartile. And the third one allows to obtain a variable in five classes with the cutpoints at the
quintiles.

- nb.categ: This concerns categorical transformations in more than two classes. Considering one of
these transformations, the most intuitive method is to use a transformation in three classes at the
tercile. For two of such transformations, we added the previous coding and a categorical transfor-
mation in four classes based on the quartile, and so on.
- cutpoint: The cutpoint argument needs to be a matrix. The form of this matrix is similar as one of the quantile matrix. The number of rows corresponds to the number of transformations (nbc) tried with this method, and the number of columns corresponds to the maximum of cutpoints (maxc) that is used in one transformation.

For example:

```
[1,]  8  16 NA  NA
[2,]  6  12 18  NA
[3,]  5  10 15  20
```

In this example, one wants to perform three transformations, hence the three rows. The first transformation leads to a categorical variable in three classes, with two cutpoints for the value "8", and the value "16". The second transformation allows to obtain a categorical transformation in four classes, with cutpoints for values: "6", "12", and "18". The last transformation tries to obtain a categorical transformation in five classes with cutpoint for values: "5", "10", "15", and "20". Therefore, we used four columns because four is the maximum of cutpoints used, in the third transformation.

**Value**

- **call**: the code used for the model.
- **n**: the number of subjects in the dataset.
- **N**: the number of resampling.
- **family**: the family function used.
- **link**: the link function used.
- **nbt**: the number of score tests realized.
- **nbb**: the number of score tests realized with BoxCox transformation.
- **nbq**: the number of score tests realized with Quantile transformation.
- **nbc**: the number of score tests realized with Cutpoint transformation.
- **vq**: the vector quantiles’ values for the best coding.
- **adj**: the number of adjustment variables.
- **trans**: the method of transformation for each coding.
- **BC**: the method of the best transformation: "Dichotomous", "Categorical", "Box-Cox", "Continuous", "Cutpoint".
- **bestcod**: the corresponding value of the transformation parameter for the best transformation.
- **naive.p.value**: the Pvalue of the best association without correction.
- **exact.p.value**: the adjusted Pvalue of the best association with an exact correction.
- **bonferroni.adjusted.p.value**: the adjusted Pvalue of the best association with the Bonferroni correction.
- **parametric.bootstrap.adjusted.p.value**: the adjusted Pvalue of the best association with the parametric bootstrap correction.
- **permutation.adjusted.p.value**: the adjusted Pvalue of the best association with the permutation correction.
Author(s)
J.Riou, A.Diakite, and B.Liquet

References

See Also
print.CPMCGLM, summary.CPMCGLM

Examples
## Not run:
# load data
data(data_sim)
#
#Example of quantile matrix definition

#Linear Gaussian Model
fit1 <- CPMCGLM(formula= Weight~Age+as.factor(Sport)+Desease+Height, family="gaussian",link="identity",data=data_sim, varcod="Age",N=1000, boxcox=c(0,1,2,3),nb.dicho=3,nb.categ=4)
### print fit1
fit1
### summary fit1
summary(fit1)

#Loglinear Poisson Model
fit2 <- CPMCGLM(formula= Stroke~Age+as.factor(Sport)+Height+Weight, family="poisson",link="log",data=data_sim, varcod="Age",N=1000, boxcox=c(0,1,2,3))
### print fit2
fit2
### summary fit2
summary(fit2)

#Logit Model
fit3 <- CPMCGLM(formula= Parameter~Age+as.factor(Sport)+Height+Weight,
family="binomial",link="logit",data=data_sim,varcod="Age",N=1000, boxcox=c(0,1,2,3),nb.dicho=3)
### print fit3
fit3
### summary fit3
summary(fit3)

# Probit Model
fit4 <- CPMCGLM(formula= Parameter~Age+as.factor(Sport)+Height+Weight, family="binomial",link="probit",data=data_sim,varcod="Age",N=1000, nboxcox=2,nb.categ=4)
### print fit4
fit4
### summary fit4
summary(fit4)

## End(Not run)

data_sim                   dataset for CPMCGLM package.

Description
This simulated dataset contains 100 subjects and 8 variables.

Usage
data(data_sim)

Format
A data frame with 100 observations on the following 8 variables.

Height  a numeric vector
Weight  a numeric vector
Age     a numeric vector
Smoke   a binary vector
Sport   a numeric vector
Desease a numeric vector
Parameter a binary vector
Stroke  a categorical vector

Examples
data(data_sim)
head(data_sim)
Description

The function provides the output of a CPMCGLM correction of pvalue.

Usage

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

Arguments

- `x` an object inheriting from classes CPMCGLM.
- `...` other parameters.

Author(s)

J.Riou, A.Diakite, and B.Liquet

See Also

- `CPMCGLM`
- `summary.CPMCGLM`

Examples

```r
# load data
# Not run:
data(data_sim)

#Linear Gaussian Model
fit1 <- CPMCGLM(formula= Weight~Age+as.factor(Sport)+Desease+Height, family="gaussian",link="identity",data=data_sim, varcod="Age",N=1000, boxcox=c(0,1,2,3))
### print fit1
fit1

### End(Not run)
```
**Summary**

A short summary of a CPMCGLM object.

**Description**

The function provides the summary of a CPMCGLM correction of p-value.

**Usage**

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

**Arguments**

- `object`: an object inheriting from classes CPMCGLM.
- `...`: other parameters.

**Author(s)**

J.Riou, A.Diakite, and B.Liquet

**See Also**

`CPMCGLM`, `print.CPMCGLM`

**Examples**

```r
## Not run:
load data
data(data_sim)

#Linear Gaussian Model
fit1 <- CPMCGLM(formula= Weight~Age+as.factor(Sport)+Desease+Height, family="gaussian",link="identity",data=data_sim,varcod="Age",N=1000, boxcox=c(0,1,2,3))
### summary fit1
summary(fit1)

## End(Not run)
```
Index

*Topic datasets
  data_sim, 6
*Topic print
  print.CPMGGLM, 7
*Topic summary
  summary.CPMGGLM, 8

bestcod (ForInternalUse), 7
Boxcox (ForInternalUse), 7
Codage (ForInternalUse), 7
codcut (ForInternalUse), 7
CPMGLM, 2, 7, 8
data_sim, 6
ForInternalUse, 7
permut (ForInternalUse), 7
print.CPMGGLM, 5, 7, 8
summary.CPMGGLM, 5, 7, 8
test.liquet (ForInternalUse), 7
test.score (ForInternalUse), 7
tmultboot (ForInternalUse), 7
transf (ForInternalUse), 7