Package ‘PResiduals’

March 4, 2015

Type Package
Title Probability-Scale Residuals and Residual Correlations
Version 0.2-1
Date 2015-03-03
Author Charles Dupont, Jeffrey Horner, Chun Li, Qi Liu, Bryan Shepherd
Maintainer Charles Dupont <charles.dupont@vanderbilt.edu>
Description Computes probability-scale residuals and residual correlations
for continuous, ordinal, binary, count, and time-to-event data.
Imports MASS, Formula, rms, sandwich, actuar
Suggests survival
License GPL (>= 2)
‘print.cobot.R’ ‘print.cocobot.R’
NeedsCompilation no
Repository CRAN
Date/Publication 2015-03-04 00:14:10

R topics documented:

PResiduals-package ................................................. 2
cobot ............................................................. 2
cocobot ........................................................... 4
countbot .......................................................... 5
diagn ............................................................... 7
GKGamma .............................................................. 8
presid ............................................................... 8
PResidData .......................................................... 10
print.cobot ........................................................ 11
print.cocobot ...................................................... 11

Index 12
PResiduals-package  Computes probability-scale residuals and residual correlations.

Description

This package outputs probability-scale residuals from multiple models and computes residual correlation. Probability-scale residual can be computed for continuous, ordinal, binary, count, and time-to-event data (although the current implementation is only for ordinal variables). Plots of probability-scale residuals can be useful for model diagnostics. Residual correlation can be used to test for conditional independence between multiple types of variables.

Author(s)

Bryan Shepherd <bryan.shepherd@vanderbilt.edu>
Chun Li <cxl791@case.edu>
Qi Liu <qi.liu.1@vanderbilt.edu>
Charles Dupont <charles.dupont@vanderbilt.edu>
Jeffrey Horner <jeffrey.horner@vanderbilt.edu>

cobot  Conditional ordinal by ordinal tests for association.

Description

cobot tests for independence between two ordered categorical variables, X and Y conditional on other variables, Z. The basic approach involves fitting models of X on Z and Y on Z and determining whether there is any remaining information between X and Y. This is done by computing one of 3 test statistics. T1 compares empirical distribution of X and Y with the joint fitted distribution of X and Y under independence conditional on Z. T2 computes the correlation between ordinal (probability-scale) residuals from both models and tests the null of no residual correlation. T3 evaluates the concordance–disconcordance of data drawn from the joint fitted distribution of X and Y under conditional independence with the empirical distribution. Details are given in Li C and Shepherd BE, Test of association between two ordinal variables while adjusting for covariates. Journal of the American Statistical Association 2010, 105:612-620.

Usage

cobot(formula, link = c("logit", "probit", "cloglog", "cauchit"),
       link.x = link, link.y = link, data, subset, na.action = na.fail,
       fisher = FALSE, conf.int = 0.95)
Arguments

formula  an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under ‘Details’.

link     The link family to be used for ordinal models of both X and Y. Defaults to ‘logit’. Other options are ‘probit’, ‘cloglog’, and ‘cauchit’.

link.x   The link function to be used for a model of the first ordered variable. Defaults to value of link.

link.y   The link function to be used for a model of the second variable. Defaults to value of link.

data     an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which cobot is called.

subset   an optional vector specifying a subset of observations to be used in the fitting process.

na.action how NAs are treated.

fisher   logical; if TRUE, Fisher transformation and delta method a used to compute p value for the test statistic based on correlation of residuals.

conf.int numeric specifying confidence interval coverage.

Details

formula is specified as X | Y ~ Z. This indicates that models of X ~ Z and Y ~ Z will be fit. The null hypothesis to be tested is H_0 : X independant of Y conditional on Z.

Note that \tau_2 can be thought of as an adjusted rank correlation.\cite{Li2012}

Value

object of ‘cobot’ class.

References


See Also

`formula`, `as.data.frame`

Examples

data(PResidData)
cobot(x|y~z, data=PResidData)
cocobot tests for independence between an ordered categorical variable, X, and a continuous variable, Y, conditional on other variables, Z. The basic approach involves fitting an ordinal model of X on Z, a linear model of Y on Z, and then determining whether there is any residual information between X and Y. This is done by computing residuals for both models, calculating their correlation, and testing the null of no residual correlation. This procedure is analogous to test statistic $T_2$ in cobot. Two test statistics (correlations) are currently output. The first is the correlation between probability-scale residuals. The second is the correlation between the observed-minus-expected residual for the continuous outcome model and a latent variable residual for the ordinal model.

Usage

```
cocobot(formula, data, link = c("logit", "probit", "cloglog", "cauchit"),
       subset, na.action = getOption("na.action"), emp = TRUE, fisher = FALSE,
       conf.int = 0.95)
```

Arguments

- `formula`: an object of class `Formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under ‘Details’.
- `link`: The link family to be used for the ordinal model of X on Z. Defaults to ‘logit’. Other options are ‘probit’, ‘cloglog’, and ‘cauchit’.
- `data`: an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which `cocobot` is called.
- `subset`: an optional vector specifying a subset of observations to be used in the fitting process.
- `na.action`: action to take when NA present in data.
- `emp`: logical indicating whether the residuals from the model of Y on Z are computed based on the assumption of normality (FALSE) or empirically (TRUE).
- `fisher`: logical indicating whether to apply fisher transformation to compute confidence intervals and p-values for the correlation.
- `conf.int`: numeric specifying confidence interval coverage.

Details

Formula is specified as X | Y ~ Z. This indicates that models of X ~ Z and Y ~ Z will be fit. The null hypothesis to be tested is $H_0 : X$ independent of $Y$ conditional on $Z$. The ordinal variable, $X$, must precede the | and be a factor variable, and Y must be continuous.
Value

object of ‘cocobot’ class.

References

Shepherd BE, Li C, Liu Q (submitted) Probability-scale residuals for continuous, discrete, and
censored data.

Examples

data(PResidData)
cocobot(y|w ~ z, data=PResidData)

countbot

*Conditional count by ordinal tests for association.*

Description

countbot tests for independence between an ordered categorical variable, X, and a count variable,
Y, conditional on other variables, Z. The basic approach involves fitting an ordinal model of X on
Z, a Poisson or Negative Binomial model of Y on Z, and then determining whether there is any
residual information between X and Y. This is done by computing residuals for both models, calculat-
ing their correlation, and testing the null of no residual correlation. This procedure is analogous
to test statistic $t_2$ in cobot. Two test statistics (correlations) are currently output. The first is the
correlation between probability-scale residuals. The second is the correlation between the Pearson
residual for the count outcome model and a latent variable residual for the ordinal model.

Usage

countbot(formula, data, link.x = c("logit", "probit", "cloglog", "cauchit"),
family.y = c("poisson", "negative binomial"), subset,
na.action = getOption("na.action"), fisher = FALSE, conf.int = 0.95)

Arguments

formula an object of class Formula (or one that can be coerced to that class): a symbolic
description of the model to be fitted. The details of model specification are given
under ‘Details’.

data an optional data frame, list or environment (or object coercible by as.data.frame
to a data frame) containing the variables in the model. If not found in data,
the variables are taken from environment(formula), typically the environment
from which countbot is called.

link.x The link family to be used for the ordinal model of X on Z. Defaults to ‘logit’.
Other options are ‘probit’, ‘cloglog’, and ‘cauchit’.
family.y  The error distribution for the count model of \( Y \) on \( Z \). Defaults to ‘poisson’. The other option is ‘negative binomial’. If ‘negative binomial’ is specified, \texttt{glm.nb} is called to fit the count model.

subset  an optional vector specifying a subset of observations to be used in the fitting process.

na.action  action to take when NA present in data.

fisher  logical indicating whether to apply fisher transformation to compute confidence intervals and \( p \)-values for the correlation.

conf.int  numeric specifying confidence interval coverage.

Details

Formula is specified as \( X \mid Y \sim Z \). This indicates that models of \( X \sim Z \) and \( Y \sim Z \) will be fit. The null hypothesis to be tested is \( H_0 : X \) independent of \( Y \) conditional on \( Z \). The ordinal variable, \( X \), must precede the \( \mid \) and be a factor variable, and \( Y \) must be an integer.

Value

object of ‘\texttt{countbot}’ class.

References

Shepherd BE, Li C, Liu Q (submitted) Probability-scale residuals for continuous, discrete, and censored data.

Examples

generate.data3 = function(alphax, betax, alphay, betay, eta, N) {
  z = rnorm(N, 0, 1)
  x = v = numeric(N)
  px = (1 + exp(- outer(alphax, betax*z, "*"))) ^ (-1)
  aa = runif(N)
  for(i in 1:N)
    x[i] = sum(aa[i] > px[,i])
  x = as.numeric(as.factor(x))
  v = rpois(N, exp(outer(alphay, betay*z+eta[x], "*")))
  return(list(x=as.factor(x), v=v, z=z))
}

set.seed(13)
alphax = c(-1, 0, 1, 2)
betax = 1
alphay = 1
betay = -.5
diagn

---

Extract or construct a diagonal matrix.

Description

This works like `diag` except when `x` is a single integer value. If `x` is a single integer value then it assumes that you want a 1 by 1 matrix with the value set to `x`.

Usage

```
diagn(x = 1, nrow = length(x), ncol = nrow)
```

Arguments

- `x` a matrix, vector or 1D array, or missing.
- `nrow`, `ncol` optional dimensions for the result when `x` is not a matrix.

Value

matrix with diagonal elements set to `x`

See Also

- `diag`

Examples

```
diag(5)
diagn(5)
```
**GKGamma**  
*Goodman-Kruskal’s $\gamma$*

**Description**
Computes Goodman-Kruskal’s $\gamma$

**Usage**
```
GKGamma(M)
```

**Arguments**
- **M**: a matrix

**Value**
- **scon**: concordance
- **sdis**: disconcordance
- **gamma**: a real number between -1 and 1. calculated as $\gamma = \frac{scon - sdis}{scon + sdis}$

**References**

**presid**  
*Probability-scale residual*

**Description**
`presid` calculates the probability-scale residual for various model function objects. Currently supported models include `glm` (Poisson, binomial, and gaussian families), `lm` in the `stats` library; `survreg` (Weibull, exponential, gaussian, logistic, and lognormal distributions) and `coxph` in the `survival` library; `polr` and `glm.nb` in the `MASS` library; and `ols`, `cph`, `lrm`, `orm`, `psm`, and `Glm` in the `rms` library.

**Usage**
```
presid(object, ...)
```

**Arguments**
- **object**: The model object for which the probability-scale residual is calculated
- **...**: Additional arguments passed to methods
Details

Probability-scale residual is \( P(Y < y) - P(Y > y) \) where \( y \) is the observed outcome and \( Y \) is a random variable from the fitted distribution.

Value

The probability-scale residual for the model

References


Examples

```r
library(survival)
library(stats)

set.seed(100)
n <- 1000
x <- rnorm(n)
t <- rweibull(n, shape=1/3, scale=exp(x))
c <- rexp(n, 1/3)
y <- pmin(t, c)
d <- ifelse(t<=c, 1, 0)

mod.survreg <- survreg(Surv(y, d) ~ x, dist="weibull")
summary(presid(mod.survreg))
plot(x, presid(mod.survreg))

##### example for proportional hazards model
n <- 1000
x <- rnorm(n)
beta0 <- 1
beta1 <- 0.5
t <- rexp(n, rate = exp(beta0 + beta1*x))
c <- rexp(n, rate=1)
y <- ifelse(t<=c, t, c)
delta <- as.integer(t<=c)

mod.coxph <- coxph(Surv(y, delta) ~ x)
presid <- presid(mod.coxph)
plot(x, presid, cex=0.4, col=delta+2)

##### example for Negative Binomial regression
library(MASS)
n <- 1000
beta0 <- 1
beta1 <- 0.5
```
x <- runif(n, min=-3, max=3)
y <- rbinom(n, mu=exp(beta0 + beta1*x), size=3)

mod.glm.nb <- glm.nb(y~x)
presid <- presid(mod.glm.nb)
summary(presid)
plot(x, presid, cex=0.4)

##### example for proportional odds model
library(MASS)
n <- 1000
x <- rnorm(n)
y <- numeric(n)
alpha = c(-1, 0, 1, 2)
beta <- 1
py <- (1 + exp(- outer(alpha, beta*x, "+")) ) ^ (-1)
aa = runif(n)
for(i in 1:n)
  y[i] = sum(aa[i] > py[,i])
y <- as.factor(y)

mod.polr <- polr(y~x, method="logistic")
summary(mod.polr)
presid <- presid(mod.polr)
summary(presid)
plot(x, presid, cex=0.4)

---

**Description**

An example dataset for use with PResidual

**Usage**

data(PResidData)

**Format**

A data frame with 500 observations on the following 5 variables.

- **x** an ordered factor with levels `1 < 2 < 3 < 4 < 5`
- **y** an ordered factor with levels `1 < 2 < 3 < 4`
- **z** a numeric vector
- **w** a numeric vector
- **c** a numeric vector of non-negative integers
print.cobot

Examples

```r
data(PResidData)
```

Description

cobot class print method

Usage

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

Arguments

- `x` : cobot object
- `...` : arguments passed to `print.default`

print.cocobot

cocobot class print method

Description

cocobot class print method

Usage

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

Arguments

- `x` : cocobot object
- `...` : arguments passed to `print.default`
Index

*Topic array
  diagn, 7
*Topic datasets
  PResidData, 10
*Topic package
  PResiduals-package, 2
*Topic print
  print.cobot, 11
  print.cocobot, 11

as.data.frame, 3–5

cobot, 2
cocobot, 4
countbot, 5
coxph, 8
cph, 8
diag, 7
diagn, 7

Formula, 3–5

GKGamma, 8
Glm, 8
glm, 8
glm.nb, 6, 8

lm, 8
lrm, 8

ols, 8
orm, 8

polr, 8
presid, 8
PResidData, 10
PResiduals (PResiduals-package), 2
PResiduals-package, 2
print.cobot, 11
print.cocobot, 11

psm, 8
survreg, 8