Package ‘DAMisc’

August 29, 2016

Type Package
Title Dave Armstrong’s Miscellaneous Functions
Version 1.4-3
Depends car, effects
Imports lattice, sm, MASS, nnet, splines, pscl, gdata, xtable, QRM, VGAM, boot
Date 2016-07-03
Author Dave Armstrong (UW-Milwaukee, Department of Political Science)
Maintainer Dave Armstrong <dave@quantoid.net>
Description Miscellaneous set of functions I use in my teaching either at the University of Wisconsin-Milwaukee or the Inter-university Consortium for Political and Social Research Summer Program in Quantitative Methods. Broadly, the functions help with presentation and interpretation of GLMs.
License GPL (>= 2)
LazyLoad yes
NeedsCompilation no
Repository CRAN
Date/Publication 2016-07-04 20:29:01

R topics documented:

DAMisc-package .......................................................... 2
aclp ................................................................. 3
aveEffPlot ........................................................... 4
BGMtest ............................................................... 5
binfit ................................................................. 6
btscs ................................................................. 7
cat2Table ............................................................ 8
combTest ............................................................ 9
crSpanTest .......................................................... 10
crTest .............................................................. 11
<table>
<thead>
<tr>
<th>Function name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAintfun</td>
<td>12</td>
</tr>
<tr>
<td>DAintfun2</td>
<td>13</td>
</tr>
<tr>
<td>france</td>
<td>15</td>
</tr>
<tr>
<td>glmChange</td>
<td>16</td>
</tr>
<tr>
<td>glmChange2</td>
<td>17</td>
</tr>
<tr>
<td>intEff</td>
<td>18</td>
</tr>
<tr>
<td>InteractionEx</td>
<td>19</td>
</tr>
<tr>
<td>intQualQuant</td>
<td>20</td>
</tr>
<tr>
<td>logit_cc</td>
<td>22</td>
</tr>
<tr>
<td>mnlAveEffPlot</td>
<td>23</td>
</tr>
<tr>
<td>mnlChange</td>
<td>24</td>
</tr>
<tr>
<td>mnlChange2</td>
<td>25</td>
</tr>
<tr>
<td>mnlfit</td>
<td>26</td>
</tr>
<tr>
<td>mnlSig</td>
<td>27</td>
</tr>
<tr>
<td>NKnots</td>
<td>28</td>
</tr>
<tr>
<td>NKnotsTest</td>
<td>29</td>
</tr>
<tr>
<td>ordAveEffPlot</td>
<td>30</td>
</tr>
<tr>
<td>ordChange</td>
<td>31</td>
</tr>
<tr>
<td>ordChange2</td>
<td>32</td>
</tr>
<tr>
<td>ordfit</td>
<td>33</td>
</tr>
<tr>
<td>outXT</td>
<td>34</td>
</tr>
<tr>
<td>panel.2cat</td>
<td>35</td>
</tr>
<tr>
<td>panel.ci</td>
<td>36</td>
</tr>
<tr>
<td>panel.doublerug</td>
<td>37</td>
</tr>
<tr>
<td>panel.transci</td>
<td>38</td>
</tr>
<tr>
<td>poisGOF</td>
<td>39</td>
</tr>
<tr>
<td>pre</td>
<td>40</td>
</tr>
<tr>
<td>prepanel.ci</td>
<td>41</td>
</tr>
<tr>
<td>print.pre</td>
<td>42</td>
</tr>
<tr>
<td>scaleDataFrame</td>
<td>43</td>
</tr>
<tr>
<td>searchVarLabels</td>
<td>43</td>
</tr>
<tr>
<td>simPredpolr</td>
<td>44</td>
</tr>
<tr>
<td>testLoess</td>
<td>45</td>
</tr>
<tr>
<td>ziChange</td>
<td>45</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Function name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DAMisc-package</strong></td>
<td><strong>47</strong></td>
</tr>
</tbody>
</table>

Dave Armstrong’s Miscellaneous Functions

**Description**

Functions to aid in the presentation of linear model results
Details

Package: DAMisc
Type: Package
Version: 1.3
Date: 2014-06-26
License: GPL (>=2)
LazyLoad: yes

These are functions that help present linear model results. Largely, the represent alternatives in presentation to other R packages. For example, the factorplot function offers an alternative to David Firth’s qvcalc package. This function calculates and presents exact variances of all simple contrasts. Both DAintfun and DAintfun2 are alternative ways of presenting interactions between two continuous variables. DAintfun2 gives results in line with the suggestions in Brambor, Clark and Golder (2006).

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)
Maintainer: Dave Armstrong <davearmstrong.ps@gmail.com>

References


aclp

Example data for btscs function

Description

A subset of data from Alvarez et. al. (1996).

Usage

data(aclp)

Format

A data frame with 4126 observations on the following 7 variables.
cname Country name
country Numeric country identifier
aveEffPlot

year  Year of observation
reg  A dichotomous variable coded 1 for dictatorship, 0 for democracy
gdpw  GDP/worker, 1985 prices
popg  Population growth
democ  A dichotomous variable coded 1 for democracy, 0 for dictatorship, (1-reg)

References

aveEffPlot  Average Effect Plot for Generalized Linear Models

Description
For objects of class glm, it calculates the change the average predicted probability for a hypothetical candidate set of values of a covariate.

Usage
aveEffPlot(obj, varname, data, R=1500, nvals=25, plot=TRUE,...)

Arguments
obj  A model object of class glm.
varname  Character string giving the variable name for which average effects are to be calculated.
data  Data frame used to fit object.
R  Number of simulations to perform.
nvals  Number of evaluation points at which the average probability will be calculated.
plot  Logical indicating whether plot should be returned, or just data (if FALSE).
...  Other arguments to be passed down to xyplot.

Details
The function plots the average effect of a model covariate, for objects of class glm. The function does not work with poly unless the coefficients are provided as arguments to the command in the model (see example below).

Value
A plot or a data frame
Author(s)
Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples
```r
data(france)
p <- poly(france$lrslef, 2)
left.mod <- glm(voteleft ~ male + age + retnat + poly(lrself, 2, coefs=attr(p, "coefs")), data=france, family=binomial)
aveEffPlot(left.mod, "age", data=france, plot=FALSE)
```

Description
This function tests the five hypotheses that Berry, Golder and Milton identify as important when two quantitative variables are interacted in a linear model.

Usage
```r
BGMtest(obj, vars, digits = 3, level = 0.05, two.sided=T)
```

Arguments
- `obj`: An object of class `lm`.
- `vars`: A vector of two variable names giving the two quantitative variables involved in the interaction. These variables must be involved in one, and only one, interaction.
- `digits`: Number of digits to be printed in the summary.
- `level`: Type I error rate for the tests.
- `two.sided`: Logical indicating whether the tests should be two-sided (if TRUE, the default) or one-sided (if FALSE).

Value
A matrix giving five t-tests.

Author(s)
Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples
```r
library(car)
data(Duncan)
mod <- lm(prestige ~ income*education + type, data=Duncan)
BGMtest(mod, c("income", "education"))
```
binfit  
Scalar Measures of Fit for Binary Variable Models

Description
Calculates scalar measures of fit for models with binary dependent variables along the lines described in Long (1997) and Long and Freese (2005).

Usage
`binfit(mod)`

Arguments
- `mod`  
A model of class `glm` with `family=binomial`.

Details
`binfit` calculates scalar measures of fit (many of which are pseudo-R-squared measures) to describe how well a model fits data with a binary dependent variable.

Value
A named vector of scalar measures of fit

Author(s)
Dave Armstrong (UW-Milwaukee, Department of Political Science)

References

Examples
```r
data(france)
left.mod <- glm(voteleft ~ male + age + retnat + poly(lrself, 2), data=france, family=binomial)
binfit(left.mod)
```
**btscs**

*Generate Spells for Binary Variables*

---

**Description**

Beck et al. (1998) identified that binary time-series cross-section data are discrete-time duration data and time dependence can be modeled in a logistic regression by including a flexible function (e.g., cubic spline) of time since the last event as a covariate. This function creates the variable identifying time since last event.

**Usage**

```r
btscs(data, event, tvar, csunit, pad.tS=FALSE)
```

**Arguments**

- `data`: A data frame.
- `event`: Character string giving the name of the dichotomous variable identifying the event (where an event is coded 1 and the absence of an event is coded 0).
- `tvar`: Character string giving the name of the time variable.
- `csunit`: Character string giving the name of the cross-sectional unit.
- `pad.ts`: Logical indicating whether the time-series should be filled in, when panels are unbalanced.

**Value**

The original data frame with one additional variable. The `spell` variable identifies the number of observed periods since the last event.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**References**


```r
library(splines)
# Data from Alvarez et. al. (1996)
data(aclp)
newdat <- btscs(aclp, "democ", "year", "country")

# Estimate Model with and without spell
full.mod <- glm(democ ~ log(gdpw) + popg + bs(spell, df=4), data=newdat, family=binomial)
restricted.mod <- glm(democ ~ log(gdpw) + popg, data=newdat, family=binomial)

# Incremental F-test of time dependence
anova(restricted.mod, full.mod, test='Chisq')
```

---

**cat2Table**

*Fitted Values and CIs for 2-Categorical Interactions*

**Description**

This function makes a table of fitted values and confidence intervals for all of the combinations of two categorical variables in an interaction.

**Usage**

```r
cat2Table(eff.obj, digits, rownames=NULL, colnames=NULL)
```

**Arguments**

- `eff.obj` An object generated by `effect` from the `effects` package where the effect is calculated for two factors involved in an interaction.
- `digits` Number of digits of the fitted values and confidence intervals to print.
- `rownames` An optional vector of row names for the table, if NULL, the levels of the factor will be used.
- `colnames` An optional vector of column names for the table, if NULL, the levels of the factor will be used.

**Value**

A matrix of fitted values and confidence intervals.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)
CombTest

Test for Combining Categories in Multinomial Logistic Regression Models.

Description

Tests the null hypothesis that categories can be combined in Multinomial Logistic Regression Models.

Usage

```
combTest(obj)
```

Arguments

- `obj` An object of class `multinom`.

Value

A matrix of test statistics and p-values.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```
library(nnet)
data(france)
mnl.mod <- multinom(vote ~ age + male + retnat + lrself, data=france)
combTest(mnl.mod)
```
crSpanTest  

Test of Span Parameter in linearity for Component + Residual Plots

Description

This function performs crTest for a user-defined range of span parameters, optionally allowing for multiple testing corrections in the p-values.

Usage

```r
crSpanTest(model, spfromto, n=10, adjust.method="none",
adjust.type=c("none", "across", "within", "both"))
```

Arguments

- **model**: A model object of class `lm`
- **spfromto**: A vector of two values across which a range of `n` span values will be generated and tested.
- **n**: Number of span parameters to test.
- **adjust.method**: Adjustment method for multiple-testing procedure, using `p.adjust` from `stats`.
- **adjust.type**: String giving the values over which the multiple testing correction will be performed. Here, ‘both’ refers to a multiple testing correction done over all span parameters and all variables in the model. ‘within’ means the multiple testing correction should be done within each model, but not across the span parameters and ‘across’ means that the multiple testing correction should be for each variable across the various span parameters, but not across variables within the same model. ‘none’ refers to a pass-through option of no multiple testing procedure.
- **...**: Other arguments to be passed down to the call to `loess`.

Value

A list with two elements:

- **x**: Sequence of span values used in testing
- **y**: p-values for each variable for each span parameter

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```r
library(car)
mod <- lm(prestige ~ income + education + women, data=Prestige)
tmp <- crSpanTest(mod, c(.1, .9), adjust.method="holm",
adjust.type="both")
matplot(tmp$x, tmp$y, type="l")
```
crTest

Test of linearity for Component + Residual Plots

Description

This function estimates a linear model and a loess model on the component-plus-residual plot (i.e., a partial residual plot) for each quantitative variable in the model. The residual sums of squares for each are used to calculate an F-test for each quantitative variable.

Usage

```r
crTest(model, adjust.method="none", ...)
```

Arguments

- `model`: A model object of class `lm`
- `...`: Other arguments to be passed down to the call to `loess`.

Value

A matrix with the following columns for each variable:

- `RSSp`: Residual sum-of-squares for the parametric (linear) model.
- `RSSnp`: Residual sum-of-squares for the non-parametric (loess) model.
- `DFnum`: Numerator degrees of freedom for the F-test: `tr(S)-(k+1)`.
- `DFdenom`: Denominator degrees of freedom for the F-test: `n-tr(S)`
- `F`: F-statistic
- `p`: p-value, potentially adjusted for multiple comparisons.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```r
library(car)
mod <- lm(prestige ~ income + education + women, data=Prestige)
crTest(mod)
```
DAintfun  

Surface Plots for Two-Way Interactions

Description

Makes surface plots to display interactions between two continuous variables.

Usage

DAintfun(obj, varnames, theta = 45, phi = 10, xlab=NULL, ylab=NULL, zlab=NULL,...)

Arguments

- obj: A model object of class `lm`
- varnames: A two-element character vector where each element is the name of a variable involved in a two-way interaction.
- theta: Angle defining the azimuthal viewing direction to be passed to `persp`.
- phi: Angle defining the colatitude viewing direction to be passed to `persp`.
- xlab: Optional label to put on the x-axis, otherwise if `NULL`, it will take the first element of `varnames`.
- ylab: Optional label to put on the y-axis, otherwise if `NULL`, it will take the second element of `varnames`.
- zlab: Optional label to put on the z-axis, otherwise if `NULL`, it will be ‘Predictions’.
- ...: Other arguments to be passed down to the initial call to `persp`.

Details

This function makes a surface plot of an interaction between two continuous covariates. If the model is

\[ y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i1} \times x_{i2} + \ldots + e_i, \]

this function plots \(b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i1} \times x_{i2}\) for values over the range of \(X_1\) and \(X_2\). The highest 75%, 50% and 25% of the bivariate density of \(X_1\) and \(X_2\) (as calculated by `sm.density` from the `sm` package) are colored in with colors of increasing gray-scale.

Value

- x1: Values of the first element of `varnames` used to make predictions.
- x2: Values of the second element of `varnames` used to make predictions.
- pred: The predictions based on the values `x1` and `x2`.
- graph: A graph is produced, but no other information is returned.
Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

data(InteractionEx)
mod <- lm(y ~ x1*x2 + z, data=InteractionEx)
DAintfun(mod, c("x1", "x2"))

## Make interactive with:
# mypanel <- function(panel){
#   DAintfun(mod, c("x1", "x2"), theta=panel$theta, phi=panel$phi)
#   panel}
# panel <- rp.control(theta=0, phi=25)
# rp.slider(panel, theta, -360, 360, mypanel, showvalue=TRUE)
# rp.slider(panel, phi, 0, 90, mypanel, showvalue=TRUE)

DAintfun2  Conditional Effects Plots for Interactions in Linear Models

Description

Generates two conditional effects plots for two interacted continuous covariates in linear models.

Usage

DAintfun2(obj, varnames, varcov=NULL, rug=TRUE, ticksize=-.03, hist=FALSE, hist.col="gray75", nclass=c(10, 10), scale.hist=.5, border=NA, name.stem = "cond_eff", xlab = NULL, ylab = NULL, plot.type = "screen")

Arguments

obj  A model object of class lm
varnames  A two-element character vector where each element is the name of a variable involved in a two-way interaction.
varcov  A variance-covariance matrix with which to calculate the conditional standard errors. If NULL, it is calculated with vcov(obj).
rug  Logical indicating whether a rug plot should be included.
ticksize  A scalar indicating the size of ticks in the rug plot (if included) positive values put the rug inside the plotting region and negative values put it outside the plotting region.
hist  Logical indicating whether a histogram of the x-variable should be included in the plotting region.
hist.col  Argument to be passed to polygon indicating the color of the histogram bins.
nclass  vector of two integers indicating the number of bins in the two histograms, which will be passed to hist.
scale.hist  A scalar in the range (0,1] indicating how much vertical space in the plotting region the histogram should take up.

border  Argument passed to polygon indicating how the border of the histogram bins should be printed (NA for no border).

name.stem  A character string giving filename to which the appropriate extension will be appended.

xlab  Optional vector of length two giving the x-labels for the two plots that are generated. The first element of the vector corresponds to the figure plotting the conditional effect of the first variable in varnames given the second and the second element of the vector corresponds to the figure plotting the conditional effect of the second variable in varnames conditional on the first.

ylab  Optional vector of length two giving the y-labels for the two plots that are generated. The first element of the vector corresponds to the figure plotting the conditional effect of the first variable in varnames given the second and the second element of the vector corresponds to the figure plotting the conditional effect of the second variable in varnames conditional on the first.

plot.type  One of ‘pdf’, ‘png’, ‘eps’ or ‘screen’, where the one of the first three will produce two graphs starting with name.stem written to the appropriate file type and the third will produce graphical output on the screen.

Details

This function produces graphs along the lines suggested by Brambor, Clark and Golder (2006) and Berry, Golder and Milton (2012), that show the conditional effect of one variable in an interaction given the values of the conditioning variable. This is an alternative to the methods proposed by John Fox in his effects package, upon which this function depends heavily.

Specifically, if the model is

\[ y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i1} \times x_{i2} + \ldots + e_i, \]

this function plots calculates the conditional effect of \( X_1 \) given \( X_2 \)

\[ \frac{\partial y}{\partial X_1} = b_1 + b_3 X_2 \]

and the variances of the conditional effects

\[ V(b_1 + b_3 X_2) = V(b_1 + X_2^2 V(b_3) + 2(1)(X_2) V(b_1, b_3)) \]

for different values of \( X_2 \) and then switches the places of \( X_1 \) and \( X_2 \), calculating the conditional effect of \( X_2 \) given a range of values of \( X_1 \). 95% confidence bounds are then calculated and plotted for each conditional effects along with a horizontal reference line at 0.

Value

graphs  Either a single graph is printed on the screen (using par(mfrow=c(1, 2))) or two figures starting with name.stem are produced where each gives the conditional effect of one variable based on the values of another.
Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

References


Examples

data(interactionex)
mod <- lm(y ~ x1*x2 + z, data=interactionex)
DAintfun2(mod, c("x1", "x2"), hist=TRUE, scale.hist=.3)

 france  
Example data for factorplot function

Description

A subset of data from the 1994 Eurobarometer for France

Usage

data(france)

Format

A data frame with 542 observations on the following 5 variables.

lrself respondent’s left-right self-placement on a 1(left)-10(right) scale
male a dummy variable coded 1 for males and 0 for females
age respondent’s age
vote a factor indicating vote choice with levels PCF, PS, Green, RPR and UDF
retnat a factor indicating the respondent’s retrospective national economic evaluation with levels Better, Same and Worse
voteleft a dichotomous variable where 1 indicates a vote for a left party, 0 otherwise

References

glmChange

Maximal First Differences for Generalized Linear Models

Description

For objects of class glm, it calculates the change in predicted responses, for maximal discrete changes in all covariates holding all other variables constant at typical values.

Usage

glmChange(obj, data, typical.dat=NULL, diffchange=c("range", "sd", "unit"), sim=FALSE, R=1000)

Arguments

obj A model object of class glm.
data Data frame used to fit object.
typical.dat Data frame with a single row containing values at which to hold variables constant when calculating first differences. These values will be passed to predict, so factors must take on a single value, but have all possible levels as their levels attribute.
diffchange A string indicating the difference in predictor values to calculate the discrete change. range gives the difference between the minimum and maximum, sd gives plus and minus one-half standard deviation change around the median and unit gives a plus and minus one-half unit change around the median.
sim Logical indicating whether simulated confidence bounds on the difference should be calculated and presented.
R Number of simulations to perform if sim is TRUE

Details

The function calculates the changes in predicted responses for maximal discrete changes in the covariates, for objects of class glm. This function works with polynomials specified with the poly function. It also works with multiplicative interactions of the covariates by virtue of the fact that it holds all other variables at typical values. By default, typical values are the median for quantitative variables and the mode for factors. The way the function works with factors is a bit different. The function identifies the two most different levels of the factor and calculates the change in predictions for a change from the level with the smallest prediction to the level with the largest prediction.

Value

A list with the following elements:

diffs A matrix of calculated first differences
minmax A matrix of values that were used to calculate the predicted changes
Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

data(france)
leftNmod <- glm(voteleft ~ male + age + retnat +
poly(lrself, 2), data=france, family=binomial)
typicalNfrance <- data.frame(
  retnat = factor(1, levels=1:3, labels=levels(france$retnat)),
  age = 35
)
glmChange(leftNmod, data=france, typical.dat=typicalNfrance)

Description

For objects of class glm, it calculates the change in predicted responses, for maximal discrete changes in all covariates holding all other variables constant at typical values.

Usage

glmChange2(obj, varname, data, change=c("unit", "sd"), R=1500)

Arguments

obj A model object of class glm.
varname Character string giving the variable name for which average effects are to be calculated.
data Data frame used to fit object.
change A string indicating the difference in predictor values to calculate the discrete change. sd gives plus and minus one-half standard deviation change around the median and unit gives a plus and minus one-half unit change around the median.
R Number of simulations to perform.

Details

The function calculates the average effect discrete changes in the covariates, for objects of class glm. This function works with polynomials specified with the poly function.

Value

A vector of values giving the average and 95 percent confidence bounds
intEff

Functions for Estimating Interaction Effects in Logit and Probit Models

Description


Usage

intEff(obj, vars, data)

Arguments

obj A binary logit or probit model estimated with glm.
vars A vector of the two variables involved in the interaction.
data A data frame used in the call to obj.

Value

A data frame with the following variable:

int_eff The correctly calculated marginal effect.
linear The incorrectly calculated marginal effect following the linear model analogy.
phat Predicted Pr(Y=1|X).
se_int_eff Standard error of int_eff.
zstat The interaction effect divided by its standard error.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)
References


Examples

data(france)
mod <- glm(voteleft ~ age*lrself + retnat + male, data=france, family=binomial)
out <- intEff(obj=mod, vars=c("age", "lrself"), data=france)
plot(out$phat, out$int_eff, xlab="Predicted Pr(Y=1|X)", ylab = "Interaction Effect")
ag <- aggregate(out$linear, list(out$phat), mean)
lines(ag[,1], ag[,2], lty=2, col="red", lwd=2)
legend("topright", c("Correct Marginal Effect", "Linear Marginal Effect"), pch=c(1, NA), lty=c(NA, 2), col=c("black", "red"), lwd=c(NA, 2), inset=.01)

Example Data for DAintfun

Description

Data to execute example code for DAintfun

Usage

data(InteractionEx)

Format

A data frame with 500 observations on the following 4 variables.

y a numeric vector
x1 a numeric vector
x2 a numeric vector
z a numeric vector

Details

These are randomly generated data to highlight the functionality of DAintfun
**intQualQuant**  
*Predictions for Factor-Numeric Interactions in Linear Models*

**Description**

This function works on linear models with a single interaction between a continuous (numeric) variable and a factor. The output is a data frame that gives the predicted effect of moving from each category to each other category of the factor over the range of values of the continuous conditioning variable.

**Usage**

```r
intQualQuant(obj, vars, level = .95, varcov=NULL, labs=NULL, n=10, onlySig=FALSE, type=c("facs", "slopes"), plot=TRUE, vals = NULL, rug=TRUE, ci=TRUE, digits=3,...)
```

**Arguments**

- `obj`: An object of class `lm`.
- `vars`: A vector of two variable names giving the two quantitative variables involved in the interaction. These variables must be involved in one, and only one, interaction.
- `level`: Confidence level desired for lower and upper bounds of confidence interval.
- `varcov`: A potentially clustered or robust variance-covariance matrix of parameters used to calculate standard errors. If `NULL`, the `vcov` function will be used.
- `labs`: An optional vector of labels that will be used to identify the effects, if `NULL`, the factor levels will be used.
- `n`: Number of values of the conditioning variable to use.
- `onlySig`: Logical indicating whether only contrasts with significant differences should be returned. Significance is determined to exist if the largest lower bound is greater than zero or the smallest upper bound is smaller than zero.
- `type`: String indicating whether the conditional partial effect of the factors is plotted (if ‘facs’), or the conditional partial effect of the quantitative variable (if ‘slopes’) is produced.
- `plot`: Logical indicating whether graphical results (if `TRUE`) or numerical results (if `FALSE`) are produced.
- `vals`: A vector of values at which the continuous variable will be held constant. If `NULL`, a sequence of length `n` across the variable's range will be used.
- `rug`: Logical indicating whether rug plots should be plotted in the panels.
- `ci`: Logical indicating whether confidence bounds should be drawn.
- `digits`: Number indicating how many decimal places to round the numeric output.
- `...`: Other arguments to be passed down to `effect` if `plot.type = 'slopes'`.
**Value**

For type = ‘facs’ and plot = FALSE, a data frame with the following values:

- **fit**: The expected difference between the two factor levels at the specified value of the conditioning variable.
- **se.fit**: The standard error of the expected differences.
- **x**: The value of the continuous conditioning variable
- **contrast**: A factor giving the two values of the factor being evaluated.
- **lower**: The lower 95% confidence interval for fit
- **upper**: The upper 95% confidence interval for fit

For type = ‘facs’ and plot = TRUE, a lattice display is returned. For type = ‘slopes’ and plot = FALSE, a character matrix with the following columns:

- **B**: The conditional effect of the quantitative variable for each level of the factor.
- **SE(B)**: The standard error of the conditional effect.
- **t-stat**: The t-statistic of the conditional effect.
- **Pr(>|t|)**: The two-sided p-value.

For type = ‘slopes’ and plot = TRUE, a lattice display is returned.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**Examples**

```r
library(car)
data(Prestige)
Prestige$income <- Prestige$income/1000
mod <- lm(prestige ~ income * type + education, data=Prestige)
intQualQuant(mod, c("income", "type"), n=10, plot.type="none")
intQualQuant(mod, c("income", "type"), n=10, plot.type="facs")
intQualQuant(mod, c("income", "type"), n=10, plot.type="slopes")
```
Functions for Estimating Interaction Effects in Logit and Probit Models

Description

Norton and Ai (2003) and Norton, Wang and Ai (2004) discuss methods for calculating the appropriate marginal effects for interactions in binary logit/probit models. These functions are direct translations of the Norton, Wang and Ai (2004) Stata code. These functions are not intended to be called by the user directly, rather they are called as needed by intEff.

Usage

logit_cc(obj=obj, int.var=int.var, vars=vars, b=b, X=X)
logit_cd(obj=obj, int.var=int.var, vars=vars, b=b, X=X)
logit_dd(obj=obj, int.var=int.var, vars=vars, b=b, X=X)
probit_cc(obj=obj, int.var=int.var, vars=vars, b=b, X=X)
probit_cd(obj=obj, int.var=int.var, vars=vars, b=b, X=X)
probit_dd(obj=obj, int.var=int.var, vars=vars, b=b, X=X)

Arguments

obj A binary logit or probit model estimated with glm.
int.var The name of the interaction variable.
vars A vector of the two variables involved in the interaction.
b Coefficients from the glm object.
x Model matrix from the glm object.

Value

A data frame with the following variable:

int_eff The correctly calculated marginal effect.
linear The incorrectly calculated marginal effect following the linear model analogy.
phat Predicted Pr(Y=1|X).
se_int_eff Standard error of int_eff.
zstat The interaction effect divided by its standard error

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)
References


---

**mnlAveEffPlot**

**Average Effects Plot for Multinomial Logistic Regression**

**Description**

Produces a plot of average effects for one variable while holding the others constant at observed values.

**Usage**

```r
mnlAveEffPlot(obj, varname, data, R = 1500, nvals = 25, plot = TRUE, ...)
```

**Arguments**

- `obj` An object of class `multinom`.
- `varname` A string indicating the variable for which the plot is desired.
- `data` The data used to estimate `obj`.
- `R` Number of simulations used to generate confidence bounds.
- `nvals` Number of evaluation points for the predicted probabilities.
- `plot` Logical indicating whether a plot should be produced (if TRUE) or numerical results should be returned (if FALSE).
- `...` Other arguments to be passed down to `xyplot`.

**Value**

Either a plot or a data frame with variables

- `mean` The average effect (i.e., predicted probability)
- `lower` The lower 95% confidence bound
- `upper` The upper 95% confidence bound
- `y` The values of the dependent variable being predicted
- `x` The values of the independent variable being manipulated


**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**References**


**Examples**

```r
library(nnet)
data(france)
mnl.mod <- multinom(vote ~ age + male + retnat + lrself, data=france)
## Not run: mnlAveEffPlot(mnl.mod, "lrself", data=france)
```

---

**mnlChange**

*Maximal First Differences for Multinomial Logistic Regression Models*

**Description**

For objects of class `multinom`, it calculates the change in predicted probabilities, for maximal discrete changes in all covariates holding all other variables constant at typical values.

**Usage**

```r
mnlChange(obj, data, typical.dat=NULL, diffchange=c("range", "sd", "unit"),
           sim=TRUE, R=1500)
```

**Arguments**

- `obj`: A model object of class `multinom`.
- `data`: Data frame used to fit object.
- `typical.dat`: Data frame with a single row containing values at which to hold variables constant when calculating first differences. These values will be passed to `predict`, so factors must take on a single value, but have all possible levels as their levels attribute.
- `diffchange`: A string indicating the difference in predictor values to calculate the discrete change. `range` gives the difference between the minimum and maximum, `sd` gives plus and minus one-half standard deviation change around the median and `unit` gives a plus and minus one-half unit change around the median.
- `sim`: Logical indicating whether simulated confidence bounds should be produced.
- `R`: Number of simulations to perform if `sim = TRUE`
Details

The function calculates the changes in predicted probabilities for maximal discrete changes in the covariates for objects of class `multinom`. This function works with polynomials specified with the `poly` function. It also works with multiplicative interactions of the covariates by virtue of the fact that it holds all other variables at typical values. By default, typical values are the median for quantitative variables and the mode for factors. The way the function works with factors is a bit different. The function identifies the two most different levels of the factor and calculates the change in predictions for a change from the level with the smallest prediction to the level with the largest prediction.

Value

A list with the following elements:

- `diffs`: A matrix of calculated first differences
- `minmax`: A matrix of values that were used to calculate the predicted changes
- `minPred`: A matrix of predicted probabilities when each variable is held at its minimum value, in turn.
- `maxPred`: A matrix of predicted probabilities when each variable is held at its maximum value, in turn.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```r
library(nnet)
data(france)
mnl.mod <- multinom(vote ~ age + male + retnat + lrself, data=france)
typical.france <- data.frame(
age = 35,
    retnat = factor(1, levels=1:3, labels=levels(france$retnat))
)
mnlchange(mnl.mod, data=france, typical.dat=typical.france)
```

Description

Calculates average effects of a variable in multinomial logistic regression holding all other variables at observed values.

Usage

```r
mnlChange2(obj, varnames, data, diffchange = c("unit", "sd"), R = 1500)
```
Arguments

obj          An object of class multinom
varnames    A string identifying the variable to be manipulated.
data        Data frame used to fit object.
diffchange  A string indicating the difference in predictor values to calculate the discrete change. sd gives plus and minus one-half standard deviation change around the median and unit gives a plus and minus one-half unit change around the median.
R            Number of simulations.

Value

A list with elements:

mean        Average effect of the variable for each category of the dependent variable.
lower       Lower 95 percent confidence bound
upper       Upper 95 percent confidence bound

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

library(nnet)
data(france)
mnl.mod <- multinom(vote ~ age + male + retnat + lrsel, data=france)
mnlChange2(mnl.mod, "lrsel", data=france, )

Description

Provides fit statistics (pseudo R-squared values) and the Fagerland, Hosmer and Bonfi (2008) specification test for Multinomial Logistic Regression models.

Usage

mnlfit(obj, permute = FALSE)
**Arguments**

- `obj`  
  An object of class `multinom`

- `permute`  
  Logical indicating whether to check all base categories for the Fagerland et al. specification test.

**Value**

A list with elements:

- `result`  
  Fit statistics.

- `permres`  
  The results of the base category permutation exercise.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**References**


**Examples**

```r
library(nnet)
data(france)
mnl.mod <- multinom(vote ~ age + male + retnat + lrself, data=france)
mnlfit(mnl.mod)
```

---

**Description**

By default, the summary for objects of class `multinom` is not particularly helpful. It still requires a lot of work on the part of the user to figure out which coefficients are significantly different from zero and which ones are not. `mnlSig` solves this problem by either flagging significant coefficients with an asterisk or only printing significant coefficients, leaving insignificant ones blank.

**Usage**

```r
mnlSig(obj, pval=.05, two.sided=TRUE, flag.sig=TRUE, insig.blank=FALSE)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>A model object of class multinom.</td>
</tr>
<tr>
<td>pval</td>
<td>The desired Type I error rate to identify coefficients as statistically significant.</td>
</tr>
<tr>
<td>two.sided</td>
<td>Logical indicating whether calculated p-values should be two-sided (if TRUE) or one-sided (if FALSE).</td>
</tr>
<tr>
<td>flag.sig</td>
<td>Logical indicating whether an asterisk should be placed beside coefficients which are significant at the pval level.</td>
</tr>
<tr>
<td>insig.blank</td>
<td>Logical indicating whether coefficients which are not significant at the pval level should be blank in the output.</td>
</tr>
</tbody>
</table>

Value

A data frame suitable for printing with the (optionally significance-flagged) coefficients from a multinomial logit model.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```r
library(nnet)
data(france)
mnl.mod <- multinom(vote ~ retnat + male + retnat + lrself, data=france)
mnlSig(mnl.mod)
```

### Description

Calculates AIC and BIC for the selection of knots in a spline over values (potentially including polynomials) up to a user-defined maximum.

### Usage

```r
Nknots(form, var, data, degree=3, min.knots=1, max.knots=10, includePoly = FALSE, plot=FALSE, criterion=c("AIC", "BIC", "CV"), cvk=10)
```
**NKnotsTest**

**Arguments**

- **form** A formula detailing the model for which smoothing is to be evaluated.
- **var** A character string identifying the variable for which smoothing is to be evaluated.
- **data** Data frame providing values of all variables in form.
- **degree** Degree of polynomial in B-spline basis functions.
- **min.knots** Minimum number of internal B-spline knots to be evaluated.
- **max.knots** Maximum number of internal B-spline knots to be evaluated.
- **includePoly** Include linear and polynomial models up to, and including degree-th order polynomials.
- **plot** Logical indicating whether a plot should be returned.
- **criterion** Statistical criterion to minimize in order to find the best number of knots - AIC, BIC or Cross-validation.
- **cvk** Number of groups for cross-validation

**Value**

A plot, if plot=TRUE, otherwise a data frame with the degrees of freedom and corresponding fit measure.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**Examples**

```r
library(car)
NKnotstest(prestige ~ education + type, var="income", data=na.omit(Prestige), plot=FALSE)
```

---

**Description**

Estimate hypothesis test of lower- and higher-order non-linear relationships against an assumed target relationship.

**Usage**

```r
NKnotsTest(form, var, data, targetdf=1, degree=3, min.knots=1, max.knots=10, adjust="none")
```
**Arguments**

- **form**: A formula detailing the model for which smoothing is to be evaluated.
- **var**: A character string identifying the variable for which smoothing is to be evaluated.
- **data**: Data frame providing values of all variables in `form`.
- **targetdf**: The assumed degrees of freedom against which the tests will be conducted.
- **degree**: Degree of polynomial in B-spline basis functions.
- **min.knots**: Minimum number of internal B-spline knots to be evaluated.
- **max.knots**: Maximum number of internal B-spline knots to be evaluated.
- **adjust**: Method by which p-values will be adjusted (see `p.adjust`)

**Value**

A matrix with F statistics, numerator and denominator DF and p-values for all required tests.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**Examples**

```r
library(car)
NKnotsTest(prestige ~ education + type, var="income", data=na.omit(Prestige), targetdf=3)
```

---

**ordAveEffPlot**

*Plot Average Effects of Variables in Proportional Odds Logistic Regression*

**Description**

For objects of class `polr` the function plots the average effect of a single variable holding all other variables at their observed values.

**Usage**

```r
ordAveEffPlot(obj, varname, data, R = 1500, nvals = 25, plot = TRUE, returnInd=FALSE, returnMprob = FALSE, ...)
```
Arguments

obj    An object of class polr
varname A string providing the name of the variable for which you want the plot to be drawn.
data   Data used to estimate obj.
R      Number of simulations to generate confidence intervals.
nvals  Number of evaluation points of the function
plot   Logical indicating whether or not the result should be plotted (if TRUE) or returned to the console (if FALSE).
returnInd Logical indicating whether average individual probabilities should be returned.
returnMprob Logical indicating whether marginal probabilities, averaged over individuals, should be returned.
...    Arguments passed down to the call to xyplot

Details

Following the advice of Hanmer and Kalkan (2013) the function calculates the average effect of a variable holding all other variables at observed values and then plots the result.

Value

Either a plot or a list with a data frame containing the variables

mean    The average effect (i.e., predicted probability)
lower   The lower 95% confidence bound
upper   The upper 95% confidence bound
y       The values of the dependent variable being predicted
x       The values of the independent variable being manipulated

and the elements Ind or Mprob, as requested.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

References


Examples

call(MASS)
data(france)
polr.mod <- polr(vote ~ age + male + retnat + lrself, data=france)
## Not run: ordAveEffPlot(polr.mod,"lrself",data=france)
ordChange

Maximal First Differences for Proportional Odds Logistic Regression Models

Description

For objects of class polr, it calculates the change in predicted probabilities, for maximal discrete changes in all covariates holding all other variables constant at typical values.

Usage

ordChange(obj, data, typical.dat=NULL, diffchange = c("range", "sd", "unit"),
  sim = TRUE, R=1500)

Arguments

obj A model object of class polr.
data Data frame used to fit object.
typical.dat Data frame with a single row containing values at which to hold variables constant when calculating first differences. These values will be passed to predict, so factors must take on a single value, but have all possible levels as their levels attribute.
diffchange A string indicating the difference in predictor values to calculate the discrete change. range gives the difference between the minimum and maximum, sd gives plus and minus one-half standard deviation change around the median and unit gives a plus and minus one-half unit change around the median.
sim Logical indicating whether or not simulations should be done to generate confidence intervals for the difference.
R Number of simulations.

Details

The function calculates the changes in predicted probabilities for maximal discrete changes in the covariates for objects of class polr. This function works with polynomials specified with the poly function. It also works with multiplicative interactions of the covariates by virtue of the fact that it holds all other variables at typical values. By default, typical values are the median for quantitative variables and the mode for factors. The way the function works with factors is a bit different. The function identifies the two most different levels of the factor and calculates the change in predictions for a change from the level with the smallest prediction to the level with the largest prediction.

Value

A list with the following elements:

diffs A matrix of calculated first differences
minmax A matrix of values that were used to calculate the predicted changes
minPred  A matrix of predicted probabilities when each variable is held at its minimum value, in turn.
maxPred  A matrix of predicted probabilities when each variable is held at its maximum value, in turn.

Author(s)
Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

library(MASS)
data(france)
polr.mod <- polr(vote ~ age + male + retnat + lrself, data=france)
typical.france <- data.frame(
age = 35,
retnat = factor(1, levels=1:3, labels=levels(france$retnat))
)
ordChange(polr.mod, data=france, typical.dat=typical.france, sim=FALSE)

ordChange2  Average Effects for Proportional Odds Logistic Regression Models

Description
For objects of class polr, it calculates the average change in predicted probabilities, for discrete changes in a covariate holding all other variables at their observed values.

Usage

ordChange2(obj, varnames, data, diffchange = c("sd", "unit"), 
R=1500)

Arguments

obj  A model object of class polr.
varnames  A vector of strings identifying the variable to be manipulated.
data  Data frame used to fit object.
diffchange  A string indicating the difference in predictor values to calculate the discrete change.  sd gives plus and minus one-half standard deviation change around the median and unit gives a plus and minus one-half unit change around the median.
R  Number of simulations.
Details

The function calculates the changes in predicted probabilities for maximal discrete changes in the
covariates for objects of class polr. This function works with polynomials specified with the poly
function. It also works with multiplicative interactions of the covariates by virtue of the fact that it
holds all other variables at typical values. By default, typical values are the median for quantitative
variables and the mode for factors. The way the function works with factors is a bit different. The
function identifies the two most different levels of the factor and calculates the change in predictions
for a change from the level with the smallest prediction to the level with the largest prediction.

Value

A list with the following elements:

- **diffs** A matrix of calculated first differences
- **minmax** A matrix of values that were used to calculate the predicted changes
- **minPred** A matrix of predicted probabilities when each variable is held at its minimum
  value, in turn.
- **maxPred** A matrix of predicted probabilities when each variable is held at its maximum
  value, in turn.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```r
library(MASS)
data(france)
polr.mod <- polr(vote ~ age + male + retnat + lrself, data=france)
typical.france <- data.frame(
  age = 35,
  retnat = factor(1, levels=1:3, labels=levels(france$retnat))
)
ordChange2(polr.mod, "age", data=france, diffchange="sd")
```

ordfit  

Fit Statistics for Proportional Odds Logistic Regression Models

Description

For objects of class polr, it calculates a number of fit statistics and specification tests.

Usage

ordfit(obj)
Arguments

obj A model object of class polr.

Value

An object of class ordfit which is a matrix containing statistics and specification tests.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

References


Examples

library(MASS)
data(france)
polr.mod <- polr(vote ~ age + male + retnat + lrsrself, data=france)
ordfit(polr.mod)

outXT Create LaTeX or CSV versions of an Object Produced by CrossTable

Description

outXT takes the output from CrossTable in the gmodels package and produces either LaTeX code or CSV file that can be imported into word processing software.

Usage

outXT(obj, count=TRUE, prop.r = TRUE, prop.c = TRUE, prop.t = TRUE, col.marg=TRUE, row.marg=TRUE, digits = 3, type = "word", file=NULL)

Arguments

obj A list returned by CrossTable from the gmodels package.
count Logical indicating whether the cell frequencies should be returned.
prop.r Logical indicating whether the row proportions should be returned.
prop.c Logical indicating whether the column proportions should be returned.
prop.t Logical indicating whether the cell proportions should be returned.
col.marg Logical indicating whether the column marginals should be printed.
row.marg Logical indicating whether the row marginals should be printed.
digits Number of digits to use in printing the proportions.
type String where `word` indicates a CSV file will be produced and `latex` indicates LaTeX code will be generated.
file Connection where the file will be written, if `NULL` the output will only be written to the console.

Value

A file containing LaTeX Code or CSV data to make a table

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

---

**panel.2cat**  
_Lattice panel function for confidence intervals with capped bars_

**Description**

This panel function is defined to plot confidence intervals in a multi-panel lattice display where the x-variable is categorical. Note, both lower and upper must be passed directly to `xyplot` as they will be passed down to the panel function.

**Usage**

```r
panel.2cat(x,y,subscripts, lower,upper, length=.)
```

**Arguments**

- `x, y`  
  Data from the call to `xyplot`.
- `subscripts`  
  Variable used to created the juxtaposed panels.
- `lower, upper`  
  95% lower and upper bounds of `y`.
- `length`  
  Length of the arrow head lines.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)
Examples

```r
library(car)
library(lattice)
library(effects)
data(Duncan)
Duncan$inc.cat <- cut(Duncan$income, 3)
mod <- lm(prestige ~ inc.cat * type + education, data=Duncan)
e1 <- effect("inc.cat*type", mod)
update(plot(e1), panel=panel.2cat)
```

panel.ci  

**Lattice panel function for confidence intervals**

Description

This panel function is defined to plot confidence intervals in a multi-panel lattice display. Note, both lower and upper must be passed directly to `xyplot` as they will be passed down to the prepanel function.

Usage

```r
panel.ci(x, y, subscripts, lower, upper, zl)
```

Arguments

- `x,y` Data from the call to `xyplot`.
- `subscripts` Variable used to created the juxtaposed panels.
- `lower, upper` 95% lower and upper bounds of `y`.
- `zl` Logical indicating whether or not a horizontal dotted line at zero is desired.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

```r
library(car)
library(lattice)
data(Ornstein)
mod <- lm(interlocks ~ log(assets)*nation, data=Ornstein)
mod.out <- intQualQuant(mod, c("log(assets)", "nation"),
                         m=25, plot=FALSE, type="facs")
xyplot(fit ~ x | contrast, data=mod.out,
xlab = "Assets", ylab = "Difference In Fitted Values",
lower=mod.out$lower, upper=mod.out$upper, zl=TRUE,
prepanel=prepanel.ci, panel=panel.ci)
```
**panel.doublerug**  
*Lattice panel function for two rug plots*

**Description**

This panel function is defined to plot two rugs, one on top of the other in a multi-panel lattice display.

**Usage**

```r
panel.doublerug(xa = NULL, xb = NULL,
regular = TRUE, start = if (regular) 0 else 0.97,
   end = if (regular) 0.03 else 1, x.units = rep("npc", 2),
   lty = 1, lwd = 1)
```

**Arguments**

- `xa, xb`  
  Numeric vectors to be plotted.
- `regular`  
  Logical flag indicating whether rug is to be drawn on the usual side (bottom/left) as opposed to the other side (top/right).
- `start, end`  
  Start and end points for the rug ticks on the y-axis.
- `x.units`  
  Character vectors, replicated to be of length two. Specifies the (grid) units associated with start and end above. `x.units` are for the rug on the x-axis and y-axis respectively (and thus are associated with start and end values on the y and x scales respectively). See `panel.rug` for more details.
- `lty, lwd`  
  Line type and width arguments (see `par` for more details).

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

---

**panel.transci**  
*Lattice panel function for translucent confidence intervals*

**Description**

This panel function is defined to plot translucent confidence intervals in a single-panel, grouped (i.e., superposed) lattice display. Note, both lower and upper must be passed directly to `xyplot` as they will be passed down to the panel function.

**Usage**

```r
panel.transci(x, y, groups, lower, upper, ...)
```
Arguments

x, y  
Data from the call to xyplot.

groups  
Variable used to created the superposed panels.

lower, upper  
95% lower and upper bounds of y.

...  
Other arguments to be passed down to the plotting functions.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Description

Deviance and Chi-squared goodness-of-fit test of the null hypothesis that poisson variance is appropriate to model the conditional dispersion of the data, given a particular model.

Usage

poisGOF(obj)

Arguments

obj  
A model object of class glm (with family=poisson).

Value

A 2x2 data frame with rows representing the different types of statistics (Deviance and Chi-squared) and columns representing the test statistic and p-value.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

References


Examples

```r
## Example taken from MASS help file for glm, identified to be
## Dobson (1990) Page 93: Randomized Controlled Trial:
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family=poisson())
poisGOF(glm.D93)
```
Description

Calculates proportional reduction in error (PRE) and expected proportional reduction in error (epre) from Herron (1999).

Usage

pre(mod1, mod2=NULL, sim=FALSE, R=2500)

Arguments

mod1 A model of class glm (with family binomial), polr or multinom for which (e)PRE will be calculated.
mod2 A model of the same class as mod1 against which proportional reduction in error will be measured. If NULL, the null model will be used.
sim A logical argument indicating whether a parametric bootstrap should be used to calculate confidence bounds for (e)PRE. See Details for more information.
R Number of bootstrap samples to be drawn if sim=TRUE.

Details

Proportional reduction in error is calculated as a function of correct and incorrect predictions (and the probabilities of correct and incorrect predictions for ePRE). When sim=TRUE, a parametric bootstrap will be used that draws from the multivariate normal distribution centered at the coefficient estimates from the model and using the estimated variance-covariance matrix of the estimators as Sigma. This matrix is used to form R versions of XB and predictions are made for each of the R different versions of XB. Confidence intervals can then be created from the bootstrap sampled (e)PRE values.

Value

An object of class pre, which is a list with the following elements:

- pre The proportional reduction in error
- epre The expected proportional reduction in error
- m1form The formula for model 1
- m2form The formula for model 2
- pcp The percent correctly predicted by model 1
- pmc The percent correctly predicted by model 2
- epcp The expected percent correctly predicted by model 1
- epmc The expected percent correctly predicted by model 2
- pre.sim A vector of bootstrapped PRE values if sim=TRUE
- epre.sim A vector of bootstrapped ePRE values if sim=TRUE
Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

References


Examples

data(france)
leftNmod <- glm(voteleft ~ male + age + retnat + poly(lrself, 2), data=france, family=binomial)
pre(leftNmod)

---

prepanel.ci  Lattice prepanel function for confidence intervals

Description

This prepanel function is defined so as to allow room for all confidence intervals plotted in a lattice display. Note, both lower and upper must be passed directly to xyplot as they will be passed down to the prepanel function.

Usage

prepanel.ci(x,y,subscripts, lower,upper)

Arguments

x,y Data from the call to xyplot.
subscripts Variable used to created the juxtaposed panels.
lower, upper 95% lower and upper bounds of y.

Value

A list giving the ranges and differences in ranges of x and the lower and upper bounds of y.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)
print.pre

Description

Prints the output from an object of class pre. The function prints all components of the calculation and optionally simulated confidence bounds.

Usage

## S3 method for class 'pre'
print(x, ..., sim.ci=.95)

Arguments

x An object of class pre.

sim.ci Coverage for the simulated confidence interval, if sim=TRUE in the call to pre.

... Other arguments passed to print, currently not implemented

Author(s)

Dave Armstrong (Department of Political Science, UW-Milwaukee)

See Also

pre
scaleDataFrame

Standardize quantitative variables in a data frame

Description

This function standardizes quantitative variables in a data frame while leaving the others untouched. This leaves not only factors, but also binary variables (those with values 0, 1, or NA).

Usage

scaleDataFrame(data)

Arguments

data A data frame.

Value

A data frame with standardized quantitative variables

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

searchVarLabels

Search Variable Labels Attribute

Description

Data imported from SPSS or Stata comes with the variable labels set (if they were set in the original dataset) as one of the dataframe’s attributes. This allows you to search the variable labels and returns the variable column number, name and label for all variables that have partially match the search term either in their labels or names.

Usage

searchVarLabels(dat, str)

Arguments

dat a data frame whose variable labels you want to search.
str string used to search variable labels.
Details

For an imported Stata dataset, variable labels are in the var.labels attribute of the dataset and in an SPSS dataset, they are in the variable.labels attribute. These are searched, ignoring case, for the desired string.

Value

matrix A matrix of dimensions n-matches x 2 is returned, where the first column is the column number of the matching variable and the second column is the variable label. The row names of the matrix are the variable names.

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

simPredpolr

Calculate Predictions for Proportional Odds Logistic Regression

Description

Calculates predicted probabilities from models of class polr from a model object and a vector of coefficient values. This is an auxiliary function used in pre if sim=TRUE.

Usage

simPredpolr(object, coefs, n.coef)

Arguments

object An object of class polr.
n.coef Number of coefficients (minus intercepts) for the polr model.
coefs A vector of coefficients where elements 1 to n.coef give model coefficients and elements n.coef+1 to k have intercepts.

Value

An n x m-category matrix of predicted probabilities

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)
testLoess  

*Significance Test for Loess vs. LM*

**Description**

Calculates an F test to evaluate significant differences between a LOESS model and a parametric alternative estimated with `lm`.

**Usage**

```r
testloess(lmobj, loessobj, alpha=.05)
```

**Arguments**

- `lmobj`: An object of class `lm`.
- `loessobj`: An object of class `loess`.
- `alpha`: Desired Type I error rate of test.

**Value**

Printed output describing the results of the test.

**Author(s)**

Dave Armstrong (UW-Milwaukee, Department of Political Science)

**Examples**

```r
library(car)
linmod <- lm(prestige ~ income, data=Prestige)
lomod <- loess(prestige ~ income, data=Prestige)
testloess(linmod, lomod)
```

---

ziChange  

*Maximal First Differences for Zero-Inflated Models*

**Description**

Calculates the change in predicted counts or optionally the predicted probability of being in the zero-count group, for maximal discrete changes in all covariates holding all other variables constant at typical values.

**Usage**

```r
ziChange(obj, data, typical.dat=NULL, type="count")
```
Arguments

obj A model object of class zeroinfl.
data Data frame used to fit object.
typical.dat Data frame with a single row containing values at which to hold variables constant when calculating first differences. These values will be passed to predict, so factors must take on a single value, but have all possible levels as their levels attribute.
type Character string of either ‘count’ (to obtain changes in predicted counts) or ‘zero’ (to obtain changes in the predicted probability of membership in the zero group).

Details

The function calculates the changes in predicted counts, or optionally the predicted probability of being in the zero group, for maximal discrete changes in the covariates. This function works with polynomials specified with the poly function. It also works with multiplicative interactions of the covariates by virtue of the fact that it holds all other variables at typical values. By default, typical values are the median for quantitative variables and the mode for factors. The way the function works with factors is a bit different. The function identifies the two most different levels of the factor and calculates the change in predictions for a change from the level with the smallest prediction to the level with the largest prediction.

Value

A list with the following elements:

diffs A matrix of calculated first differences
minmax A matrix of values that were used to calculate the predicted changes

Author(s)

Dave Armstrong (UW-Milwaukee, Department of Political Science)

Examples

library(pscl)
## Example from the help file for zeroinfl in the pscl package
data("bioChemists", package = "pscl")
fm_zinb <- zeroinfl(art ~ fem + mar + kid5 + phd + ment |
                    fem + mar + kid5 + phd + ment, data = bioChemists, dist = "negbin")
typical.bioChem <- data.frame(
  kid5 = 2,
  mar = factor(1, levels=1:2, labels=levels(bioChemists$mar))
)
zichange(fm_zinb, data=bioChemists, typical.dat=typical.bioChem, type="zero")
zichange(fm_zinb, data=bioChemists, typical.dat=typical.bioChem, type="count")
# Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>datasets</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>aclp, 3</td>
<td>france, 15</td>
<td></td>
</tr>
<tr>
<td>InteractionEx, 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aclp, 3</td>
<td>aveEffPlot, 4</td>
<td></td>
</tr>
<tr>
<td>BGMtest, 5</td>
<td>binfit, 6</td>
<td></td>
</tr>
<tr>
<td>btscs, 7</td>
<td>cat2Table, 8</td>
<td></td>
</tr>
<tr>
<td>combTest, 9</td>
<td>crSpanTest, 10</td>
<td></td>
</tr>
<tr>
<td>crTest, 11</td>
<td>Daintfun, 12</td>
<td></td>
</tr>
<tr>
<td>DAMisc (DAMisc-package), 2</td>
<td>DAMisc-package, 2</td>
<td></td>
</tr>
<tr>
<td>france, 15</td>
<td>glmChange, 16</td>
<td></td>
</tr>
<tr>
<td>glmChange2, 17</td>
<td>intEff, 18</td>
<td></td>
</tr>
<tr>
<td>InteractionEx, 19</td>
<td>intQualQuant, 20</td>
<td></td>
</tr>
<tr>
<td>logit_cc, 22</td>
<td>mnlAveEffPlot, 23</td>
<td></td>
</tr>
<tr>
<td>logit_cd (logit_cc), 22</td>
<td>mnlChange, 24</td>
<td></td>
</tr>
<tr>
<td>logit_dd (logit_cc), 22</td>
<td>mnlChange2, 25</td>
<td></td>
</tr>
<tr>
<td>mnlfit, 26</td>
<td>mnlSig, 27</td>
<td></td>
</tr>
<tr>
<td>NKnotsTest, 29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordAveEffPlot, 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordChange, 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordChange2, 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordfit, 34</td>
<td>outXT, 35</td>
<td></td>
</tr>
<tr>
<td>p.adjust, 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>panel.2cat, 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>panel.ci, 37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>panel.doublerug, 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>panel.transci, 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>poisGOF, 39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pre, 40</td>
<td>prepanel.ci, 41</td>
<td></td>
</tr>
<tr>
<td>print.mnlfit (mnlfit), 26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>print.ordfit (ordfit), 34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>print.pre, 42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>probit_cc (logit_cc), 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>probit_cd (logit_cc), 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>probit_dd (logit_cc), 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scaleDataFrame, 43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>searchVarLabels, 43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>simpPredpolr, 44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testLoess, 45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zicChange, 45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>