

Package ‘list’

March 22, 2012

Version 5.1

Date 2012-3-22

Title Statistical Methods for the Item Count Technique and List Experiment

Author Graeme Blair <gblair@Princeton.edu>, Kosuke Imai <kimai@Princeton.edu>

Maintainer Graeme Blair <gblair@Princeton.edu>

Depends R (>= 2.14.0), utils, VGAM (>= 0.8-1), magic (>= 1.4-6), gamlss.dist (>= 4.0-0), MASS (>= 7.3-4), quadprog (>= 1.5-3), arm (>= 1.4-6), corpcor (>= 1.5.6), mvtnorm (>= 0.9-9), sandwich (>= 2.2-6), coda (>= 0.14-2)

Description list is a publicly available R package that allows researchers to conduct multivariate statistical analyses of survey data with list experiments. In addition, the package implements the statistical test that is designed to detect certain failures of list experiments. This survey methodology is also known as the item count technique or the unmatched count technique and is an alternative to the commonly used randomized response method. The package implements the methods developed by Imai (2011) and Blair and Imai (2012), a Bayesian MCMC implementation of regression for the standard and multiple sensitive item list experiment designs including ceiling and floor effects and a random effects setup, and a Bayesian MCMC hierarchical regression model with up to three hierarchical groups.

LazyLoad yes

LazyData yes

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-03-22 21:06:54

R topics documented:

affirm	2
ict.test	3
ictreg	4
ictregBayes	10
ictregBayesHier	18
mis	25
multi	26
plot.predict.ictreg	27
predict.ictreg	29
predict.ictregBayes	33
predict.ictregBayesHier	35
race	37
summary.ictreg	38

Index	41
--------------	-----------

affirm	<i>The 1991 National Race and Politics Survey</i>
--------	---

Description

This dataset is a subset of the 1991 National Race and Politics Survey and contains the item count technique or the list experiment. The main question reads as follows: Now I'm going to read you four things that sometimes make people angry or upset. After I read all (three/four), just tell me HOW MANY of them upset you. (I don't want to know which ones, just how many.) (1) "the federal government increasing the tax on gasoline;" (2) "professional athletes getting million-dollar-plus salaries;" (3) "large corporations polluting the environment;" (4) "black leaders asking the government for affirmative action."

where the last item is presented only with the treatment group and the control list only contains the first three items.

Usage

```
data(race)
```

Format

A data frame containing the following 6 variables for 1171 observations.

y	numeric	the number of items that make respondents angry	0 - 4
south	numeric	whether or not a respondents live in a southern state	0 - 1
male	numeric	whether or not a respondent is male	0 - 1
college	numeric	whether or not a respondent attended some college	0 - 1
age	numeric	age of a respondent divided by 10	
treat	numeric	treatment status	0 - 1

Source

The full data set is available at SDA (Survey Documentation and Analysis; <http://sda.berkeley.edu/D3/Natlrace/Doc/nrac.htm>)

ict.test	<i>Item Count Technique</i>
----------	-----------------------------

Description

Function to conduct a statistical test with the null hypothesis that there is no "design effect" in a list experiment, a failure of the experiment.

Usage

```
ict.test(y, treat, J = NA, alpha = 0.05, n.draws = 250000, gms = TRUE,
         pi.table = TRUE)
```

Arguments

y	A numerical vector containing the response data for a list experiment.
treat	A numerical vector containing the binary treatment status for a list experiment.
J	Number of non-sensitive (control) survey items.
alpha	Confidence level for the statistical test.
n.draws	Number of Monte Carlo draws.
gms	A logical value indicating whether the generalized moment selection procedure should be used.
pi.table	A logical value indicating whether a table of estimated proportions of respondent types with standard errors is displayed.

Details

This function allows the user to perform a statistical test on data from a list experiment or item count technique with the null hypothesis of no design effect. A design effect occurs when an individual's response to the non-sensitive items changes depending upon the respondent's treatment status.

Value

ict.test returns a numerical scalar with the Bonferroni-corrected minimum p-value of the statistical test.

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

See Also

`ictreg` for list experiment regression based on the assumption of no design effect

Examples

```
data(affirm)
data(race)

# Conduct test with null hypothesis that there is no design effect
# Replicates results on Blair and Imai (2010) pg. 30

test.value.affirm <- ict.test(affirm$y, affirm$treat, J = 3, gms = TRUE)

test.value.race <- ict.test(race$y, race$treat, J = 3, gms = TRUE)
```

ictreg

Item Count Technique

Description

Function to conduct multivariate regression analyses of survey data with the item count technique, also known as the list experiment and the unmatched count technique.

Usage

```
ictreg(formula, data = parent.frame(), treat = "treat", J, method = "ml",
       overdispersed = FALSE, constrained = TRUE, floor = FALSE,
       ceiling = FALSE, ceiling.fit = "glm", floor.fit = "glm",
       ceiling.formula = ~ 1, floor.formula = ~ 1, fit.start = "lm",
       fit.nonsensitive = "nls", multi.condition = "none",
       maxIter = 5000, verbose = FALSE, ...)
```

Arguments

<code>formula</code>	An object of class "formula": a symbolic description of the model to be fitted.
<code>data</code>	A data frame containing the variables in the model
<code>treat</code>	Name of treatment indicator as a string. For single sensitive item models, this refers to a binary indicator, and for multiple sensitive item models it refers to a multi-valued variable with zero representing the control condition. This can be an integer (with 0 for the control group) or a factor (with "control" for the control group).

J	Number of non-sensitive (control) survey items.
method	Method for regression, either ml for the Maximum Likelihood (ML) estimation with the Expectation-Maximization algorithm; lm for linear model estimation; or nls for the Non-linear Least Squares (NLS) estimation with the two-step procedure.
overdispersed	Indicator for the presence of overdispersion. If TRUE, the beta-binomial model is used in the EM algorithm, if FALSE the binomial model is used. Not relevant for the NLS or lm methods.
constrained	A logical value indicating whether the control group parameters are constrained to be equal. Not relevant for the NLS or lm methods
floor	A logical value indicating whether the floor liar model should be used to adjust for the possible presence of respondents dishonestly reporting a negative preference for the sensitive item among those who hold negative views of all the non-sensitive items.
ceiling	A logical value indicating whether the ceiling liar model should be used to adjust for the possible presence of respondents dishonestly reporting a negative preference for the sensitive item among those who hold affirmative views of all the non-sensitive items.
ceiling.fit	Fit method for the M step in the EM algorithm used to fit the ceiling liar model. glm uses standard logistic regression, while bayesglm uses logistic regression with a weakly informative prior over the parameters.
floor.fit	Fit method for the M step in the EM algorithm used to fit the floor liar model. glm uses standard logistic regression, while bayesglm uses logistic regression with a weakly informative prior over the parameters.
ceiling.formula	Covariates to include in ceiling liar model. These must be a subset of the covariates used in formula.
floor.formula	Covariates to include in floor liar model. These must be a subset of the covariates used in formula.
fit.start	Fit method for starting values for standard design ml model. The options are lm, glm, and nls, which use OLS, logistic regression, and non-linear least squares to generate starting values, respectively. The default is nls.
fit.nonsensitive	Fit method for the non-sensitive item fit for the nls method and the starting values for the ml method for the modified design. Options are glm and nls, and the default is nls.
multi.condition	For the multiple sensitive item design, covariates representing the estimated count of affirmative responses for each respondent can be included directly as a level variable by choosing level, or as indicator variables for each value but one by choosing indicators. The default is none.
maxIter	Maximum number of iterations for the Expectation-Maximization algorithm of the ML estimation. The default is 5000.
verbose	a logical value indicating whether model diagnostics are printed out during fitting.
...	further arguments to be passed to NLS regression commands.

Details

This function allows the user to perform regression analysis on data from the item count technique, also known as the list experiment and the unmatched count technique.

Three list experiment designs are accepted by this function: the standard design; the multiple sensitive item standard design; and the modified design proposed by Corstange (2009).

For the standard design, three methods are implemented in this function: the linear model; the Maximum Likelihood (ML) estimation for the Expectation-Maximization (EM) algorithm; the nonlinear least squares (NLS) estimation with the two-step procedure both proposed in Imai (2010); and the Maximum Likelihood (ML) estimator in the presence of two types of dishonest responses, "ceiling" and "floor" liars. The ceiling model, floor model, or both, as described in Blair and Imai (2010) can be activated by using the `ceiling` and `floor` options. The constrained and unconstrained ML models presented in Imai (2010) are available through the `constrained` option, and the user can specify if overdispersion is present in the data for the no liars models using the `overdispersed` option to control whether a beta-binomial or binomial model is used in the EM algorithm to model the item counts.

The modified design and the multiple sensitive item design are automatically detected by the function, and only the binomial model without overdispersion is available.

Value

`ictreg` returns an object of class "ictreg". The function `summary` is used to obtain a table of the results. The object `ictreg` is a list that contains the following components. Some of these elements are not available depending on which method is used (`lm`, `nls` or `ml`), which design is used (`standard`, `modified`), whether multiple sensitive items are include (`multi`), and whether the constrained model is used (`constrained = TRUE`).

<code>par.treat</code>	point estimate for effect of covariate on item count fitted on treatment group
<code>se.treat</code>	standard error for estimate of effect of covariate on item count fitted on treatment group
<code>par.control</code>	point estimate for effect of covariate on item count fitted on control group
<code>se.control</code>	standard error for estimate of effect of covariate on item count fitted on control group
<code>coef.names</code>	variable names as defined in the data frame
<code>design</code>	call indicating whether the standard design as proposed in Imai (2010) or the modified design as proposed in Corstange (2009) is used
<code>method</code>	call of the method used
<code>overdispersed</code>	call indicating whether data is overdispersed
<code>constrained</code>	call indicating whether the constrained model is used
<code>boundary</code>	call indicating whether the floor/ceiling boundary models are used
<code>multi</code>	indicator for whether multiple sensitive items were included in the data frame
<code>call</code>	the matched call
<code>data</code>	the data argument
<code>x</code>	the design matrix

<code>y</code>	the response vector
<code>treat</code>	the vector indicating treatment status
<code>J</code>	Number of non-sensitive (control) survey items set by the user or detected.
<code>treat.labels</code>	a vector of the names used by the <code>treat</code> vector for the sensitive item or items. This is the names from the <code>treat</code> indicator if it is a factor, or the number of the item if it is numeric.
<code>control.label</code>	a vector of the names used by the <code>treat</code> vector for the control items. This is the names from the <code>treat</code> indicator if it is a factor, or the number of the item if it is numeric.

For the floor/ceiling models, several additional output objects are included:

<code>ceiling</code>	call indicating whether the assumption of no ceiling liars is relaxed, and ceiling parameters are estimated
<code>par.ceiling</code>	point estimate for effect of covariate on whether respondents who answered affirmatively to all non-sensitive items and hold a true affirmative opinion toward the sensitive item lied and reported a negative response to the sensitive item
<code>se.ceiling</code>	standard error for estimate for effect of covariate on whether respondents who answered affirmatively to all non-sensitive items and hold a true affirmative opinion toward the sensitive item lied and reported a negative response to the sensitive item
<code>floor</code>	call indicating whether the assumption of no floor liars is relaxed, and floor parameters are estimated
<code>par.ceiling</code>	point estimate for effect of covariate on whether respondents who answered negatively to all non-sensitive items and hold a true affirmative opinion toward the sensitive item lied and reported a negative response to the sensitive item
<code>se.ceiling</code>	standard error for estimate for effect of covariate on whether respondents who answered negatively to all non-sensitive items and hold a true affirmative opinion toward the sensitive item lied and reported a negative response to the sensitive item
<code>coef.names.ceiling</code>	variable names from the ceiling liar model fit, if applicable
<code>coef.names.floor</code>	variable names from the floor liar model fit, if applicable

For the multiple sensitive item design, the `par.treat` and `se.treat` vectors are returned as lists of vectors, one for each sensitive item.

For the unconstrained model, the `par.control` and `se.control` output is replaced by:

<code>par.control.phi0</code>	point estimate for effect of covariate on item count fitted on treatment group
<code>se.control.phi0</code>	standard error for estimate of effect of covariate on item count fitted on treatment group
<code>par.control.phi1</code>	point estimate for effect of covariate on item count fitted on treatment group

`se.control.phi1` standard error for estimate of effect of covariate on item count fitted on treatment group

Depending upon the estimator requested by the user, model fit statistics are also included:

`llik` the log likelihood of the model, if `ml` is used

`resid.se` the residual standard error, if `nls` or `lm` are used. This will be a scalar if the standard design was used, and a vector if the multiple sensitive item design was used

`resid.df` the residual degrees of freedom, if `nls` or `lm` are used. This will be a scalar if the standard design was used, and a vector if the multiple sensitive item design was used

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." Political Analysis. Forthcoming. available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." Journal of the American Statistical Association, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[predict.ictreg](#) for fitted values

Examples

```
data(race)

# Calculate list experiment difference in means

diff.in.means.results <- ictreg(y ~ 1, data = race,
                               treat = "treat", J=3, method = "lm")

summary(diff.in.means.results)

# Fit linear regression
# Replicates Table 1 Columns 1-2 Imai (2011); note that age is divided by 10

lm.results <- ictreg(y ~ south + age + male + college, data = race,
                    treat = "treat", J=3, method = "lm")

summary(lm.results)
```

```
# Fit two-step non-linear least squares regression
# Replicates Table 1 Columns 3-4 Imai (2011); note that age is divided by 10

nls.results <- ictreg(y ~ south + age + male + college, data = race,
                    treat = "treat", J=3, method = "nls")

summary(nls.results)

## Not run:

# Fit EM algorithm ML model with constraint
# Replicates Table 1 Columns 5-6, Imai (2011); note that age is divided by 10

ml.constrained.results <- ictreg(y ~ south + age + male + college, data = race,
                               treat = "treat", J=3, method = "ml",
                               overdispersed = FALSE, constrained = TRUE)

summary(ml.constrained.results)

# Fit EM algorithm ML model with no constraint
# Replicates Table 1 Columns 7-10, Imai (2011); note that age is divided by 10

ml.unconstrained.results <- ictreg(y ~ south + age + male + college, data = race,
                                   treat = "treat", J=3, method = "ml",
                                   overdispersed = FALSE, constrained = FALSE)

summary(ml.unconstrained.results)

# Fit EM algorithm ML model for multiple sensitive items
# Replicates Table 3 in Blair and Imai (2010)

multi.results <- ictreg(y ~ male + college + age + south + south:age, treat = "treat",
                      J = 3, data = multi, method = "ml",
                      multi.condition = "level")

summary(multi.results)

# Fit standard design ML model
# Replicates Table 7 Columns 1-2 in Blair and Imai (2010)

noboundary.results <- ictreg(y ~ age + college + male + south, treat = "treat",
                            J = 3, data = affirm, method = "ml",
                            overdispersed = FALSE)

summary(noboundary.results)

# Fit standard design ML model with ceiling effects alone
# Replicates Table 7 Columns 3-4 in Blair and Imai (2010)

ceiling.results <- ictreg(y ~ age + college + male + south, treat = "treat",
                        J = 3, data = affirm, method = "ml", fit.start = "nls",
                        ceiling = TRUE, ceiling.fit = "bayesglm",
                        ceiling.formula = ~ age + college + male + south)
```

```

summary(ceiling.results)

# Fit standard design ML model with floor effects alone
# Replicates Table 7 Columns 5-6 in Blair and Imai (2010)

floor.results <- ictreg(y ~ age + college + male + south, treat = "treat",
  J = 3, data = affirm, method = "ml", fit.start = "glm",
  floor = TRUE, floor.fit = "bayesglm",
  floor.formula = ~ age + college + male + south)

summary(floor.results)

# Fit standard design ML model with floor and ceiling effects
# Replicates Table 7 Columns 7-8 in Blair and Imai (2010)

both.results <- ictreg(y ~ age + college + male + south, treat = "treat",
  J = 3, data = affirm, method = "ml",
  floor = TRUE, ceiling = TRUE,
  floor.fit = "bayesglm", ceiling.fit = "bayesglm",
  floor.formula = ~ age + college + male + south,
  ceiling.formula = ~ age + college + male + south)

summary(both.results)

## End(Not run)

```

 ictregBayes

Item Count Technique

Description

Function to conduct multivariate regression analyses of survey data with the item count technique, also known as the list experiment and the unmatched count technique.

Usage

```

ictregBayes(formula, data = parent.frame(), treat = "treat", J,
  constrained.single = c("full", "none", "intercept"),
  constrained.multi = TRUE, fit.start = "lm",
  n.draws = 10000, burnin = 5000, thin = 0, delta.start,
  psi.start, Sigma.start, Phi.start, delta.mu0, psi.mu0,
  delta.A0, psi.A0, Sigma.df, Sigma.scale, Phi.df, Phi.scale,
  delta.tune, psi.tune, gamma.tune, zeta.tune,
  formula.mixed, group.mixed,
  verbose = TRUE, robit = FALSE, df = 5, ...)

```

Arguments

<code>formula</code>	An object of class "formula": a symbolic description of the model to be fitted.
<code>data</code>	A data frame containing the variables in the model
<code>treat</code>	Name of treatment indicator as a string. For single sensitive item models, this refers to a binary indicator, and for multiple sensitive item models it refers to a multi-valued variable with zero representing the control condition. This can be an integer (with 0 for the control group) or a factor (with "control" for the control group).
<code>J</code>	Number of non-sensitive (control) survey items. This will be set automatically to the maximum value of the outcome variable in the treatment group if no input is sent by the user.
<code>constrained.single</code>	A logical value indicating whether the control group parameters are constrained to be equal in the single sensitive item design.
<code>constrained.multi</code>	A logical value indicating whether the non-sensitive item count is included as a predictor in the sensitive item fits for the multiple sensitive item design.
<code>fit.start</code>	Fit method for starting values. The options are <code>lm</code> , <code>glm</code> , <code>nls</code> , and <code>m1</code> , which use OLS, logistic regression, non-linear least squares, and maximum likelihood estimation to generate starting values, respectively. The default is <code>lm</code> .
<code>n.draws</code>	Number of MCMC iterations after the burnin.
<code>burnin</code>	The number of initial MCMC iterations that are discarded.
<code>thin</code>	The interval of thinning, in which every other (<code>thin = 1</code>) or more iterations are discarded in the output object
<code>delta.start</code>	Optional starting values for the sensitive item fit. This should be a vector with the length of the number of covariates for the single sensitive item design, and either a vector or a list with a vector of starting values for each of the sensitive items. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
<code>psi.start</code>	Optional starting values for the control items fit. This should be a vector of length the number of covariates for the constrained models. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
<code>Sigma.start</code>	Optional starting values for Sigma parameter for mixed effects models.
<code>Phi.start</code>	Optional starting values for the Phi parameter for mixed effects models.
<code>delta.mu0</code>	Optional vector of prior means for the sensitive item fit parameters, a vector of length the number of covariates.
<code>psi.mu0</code>	Optional vector of prior means for the control item fit parameters, a vector of length the number of covariates.
<code>delta.A0</code>	Optional matrix of prior precisions for the sensitive item fit parameters, a matrix of dimension the number of covariates.
<code>psi.A0</code>	Optional matrix of prior precisions for the control items fit parameters, a matrix of dimension the number of covariates.
<code>Sigma.df</code>	Optional prior degrees of freedom parameter for mixed effects models.

<code>Sigma.scale</code>	Optional prior scale parameter for mixed effects models.
<code>Phi.df</code>	Optional prior degrees of freedom parameter for mixed effects models.
<code>Phi.scale</code>	Optional prior scale parameter for mixed effects models.
<code>delta.tune</code>	A required vector of tuning parameters for the Metropolis algorithm for the sensitive item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>psi.tune</code>	A required vector of tuning parameters for the Metropolis algorithm for the control item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>gamma.tune</code>	An optional vector of tuning parameters for the Metropolis algorithm for the control item fit for the random effects. This can be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>zeta.tune</code>	An optional vector of tuning parameters for the Metropolis algorithm for the sensitive item fit for the random effects. This can be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>formula.mixed</code>	To specify a mixed effects model, include this formula object for the group-level fit. <code>~1</code> allows intercepts to vary, and including covariates in the formula allows the slopes to vary also.
<code>group.mixed</code>	A numerical group indicator specifying which group each individual belongs to for a mixed effects model.
<code>verbose</code>	A logical value indicating whether model diagnostics are printed out during fitting.
<code>robit</code>	A logical value indicating whether the robit model is used for the sensitive item fit or logistic regression is used (default).
<code>df</code>	Degrees of freedom for the robit model for the sensitive item fit, only used if <code>robit</code> is set to TRUE.
<code>...</code>	further arguments to be passed to NLS regression commands.

Details

This function allows the user to perform regression analysis on data from the item count technique, also known as the list experiment and the unmatched count technique using a Bayesian MCMC algorithm.

Unlike the maximum likelihood and least squares estimators in the `ictreg` function, the Metropolis algorithm for the Bayesian MCMC estimators in this function must be tuned to work correctly. The `delta.tune` and `psi.tune` are required, and the values, one for each estimated parameter, will need to be manipulated. The output of the `ictregBayes` function, and of the `summary` function run on an `ictregBayes` object display the acceptance ratios from the Metropolis algorithm. If these values are far from 0.4, the tuning parameters should be changed until the ratios approach 0.4.

For the single sensitive item design, the model can constrain all control parameters to be equal (`constrained = "full"`), or just the intercept (`constrained = "intercept"`) or all the control fit parameters can be allowed to vary across the potential sensitive item values (`constrained = "none"`).

For the multiple sensitive item design, the model can include the estimated number of affirmative responses to the control items as a covariate in the sensitive item model fit (constrained set to TRUE) or exclude it (FALSE).

Convergence is at times difficult to achieve, so we recommend running multiple chains from overdispersed starting values by, for example, running an MLE or linear model using the `ictreg()` function, and then generating a set of overdispersed starting values using those estimates and their estimated variance-covariance matrix. An example is provided below for each of the possible designs. Running `summary()` after such a procedure will output the Gelman-Rubin convergence statistics in addition to the estimates. If the G-R statistics are all below 1.1, the model is said to have converged.

Value

`ictregBayes` returns an object of class "ictregBayes". The function `summary` is used to obtain a table of the results, using the `coda` package. Two attributes are also included, the data ("x"), the call ("call"), which can be extracted using the command, e.g., `attr(ictregBayes.object, "x")`.

<code>mcmc</code>	an object of class "mcmc" that can be analyzed using the <code>coda</code> package.
<code>x</code>	the design matrix
<code>multi</code>	a logical value indicating whether the data included multiple sensitive items.
<code>constrained</code>	a logical or character value indicating whether the control group parameters are constrained to be equal in the single sensitive item design, and whether the non-sensitive item count is included as a predictor in the sensitive item fits for the multiple sensitive item design.
<code>delta.start</code>	Optional starting values for the sensitive item fit. This should be a vector with the length of the number of covariates. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
<code>psi.start</code>	Optional starting values for the control items fit. This should be a vector of length the number of covariates. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
<code>delta.mu0</code>	Optional vector of prior means for the sensitive item fit parameters, a vector of length the number of covariates.
<code>psi.mu0</code>	Optional vector of prior means for the control item fit parameters, a vector of length the number of covariates.
<code>delta.A0</code>	Optional matrix of prior precisions for the sensitive item fit parameters, a matrix of dimension the number of covariates.
<code>psi.A0</code>	Optional matrix of prior precisions for the control items fit parameters, a matrix of dimension the number of covariates.
<code>delta.tune</code>	A required vector of tuning parameters for the Metropolis algorithm for the sensitive item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>psi.tune</code>	A required vector of tuning parameters for the Metropolis algorithm for the control item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>J</code>	Number of non-sensitive (control) survey items set by the user or detected.

`treat.labels` a vector of the names used by the `treat` vector for the sensitive item or items. This is the names from the `treat` indicator if it is a factor, or the number of the item if it is numeric.

`control.label` a vector of the names used by the `treat` vector for the control items. This is the names from the `treat` indicator if it is a factor, or the number of the item if it is numeric.

`call` the matched call

If the data includes multiple sensitive items, the following object is also included:

`treat.values` a vector of the values used in the `treat` vector for the sensitive items, either character or numeric depending on the class of `treat`. Does not include the value for the control status

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[predict.ictreg](#) for fitted values

Examples

```
data(race)

## Not run:

## Multiple chain MCMC list experiment regression
## starts with overdispersed MLE starting values

## Standard single sensitive-item design

## Control item parameters fully constrained

mle.estimates <- ictreg(y ~ male + college + age + south, data = race)

draws <- mvrnorm(n = 3, mu = coef(mle.estimates),
  Sigma = vcov(mle.estimates) * 9)
```

```

bayesDraws.1 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[1, 1:5], psi.start = draws[1, 6:10], burnin = 10000,
  n.draws = 100000, delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5),
  constrained.single = "full")

bayesDraws.2 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[2, 1:5], psi.start = draws[2, 6:10], burnin = 10000,
  n.draws = 100000, delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5),
  constrained.single = "full")

bayesDraws.3 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[3, 1:5], psi.start = draws[3, 6:10], burnin = 10000,
  n.draws = 100000, delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5),
  constrained.single = "full")

bayesSingleConstrained <- as.list(bayesDraws.1, bayesDraws.2, bayesDraws.3)

summary(bayesSingleConstrained)

## Control item parameters unconstrained

mle.estimate <- ictreg(y ~ male + college + age + south, data = race,
  constrained = FALSE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimate),
  Sigma = vcov(mle.estimate) * 9)

bayesDraws.1 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[1, 1:5], psi.start = list(psi0 = draws[1, 6:10],
  psi1 = draws[1, 11:15]), burnin = 10000, n.draws = 100000,
  delta.tune = diag(.002, 5),
  psi.tune = list(psi0 = diag(.0017, 5), psi1 = diag(.00005, 5)),
  constrained.single = "none")

bayesDraws.2 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[2, 1:5], psi.start = list(psi0 = draws[2, 6:10],
  psi1 = draws[2, 11:15]), burnin = 10000, n.draws = 100000,
  delta.tune = diag(.002, 5),
  psi.tune = list(psi0 = diag(.0017, 5), psi1 = diag(.00005, 5)),
  constrained.single = "none")

bayesDraws.3 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[3, 1:5], psi.start = list(psi0 = draws[3, 6:10],
  psi1 = draws[3, 11:15]), burnin = 10000, n.draws = 100000,
  delta.tune = diag(.002, 5),
  psi.tune = list(psi0 = diag(.0017, 5), psi1 = diag(.00005, 5)),
  constrained.single = "none")

bayesSingleUnconstrained <- as.list(bayesDraws.1, bayesDraws.2, bayesDraws.3)

summary(bayesSingleUnconstrained)

## Control item parameters constrained except intercept

```

```

mle.estimateds <- ictreg(y ~ male + college + age + south, data = race,
  constrained = TRUE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimateds),
  Sigma = vcov(mle.estimateds) * 9)

bayesDraws.1 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[1, 1:5], psi.start = c(draws[1, 6:10],0),
  burnin = 10000, n.draws = 100000, delta.tune = diag(.002, 5),
  psi.tune = diag(.0004, 6), constrained.single = "intercept")

bayesDraws.2 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[2, 1:5], psi.start = c(draws[2, 6:10],0),
  burnin = 10000, n.draws = 100000, delta.tune = diag(.002, 5),
  psi.tune = diag(.0004, 6), constrained.single = "intercept")

bayesDraws.3 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[3, 1:5], psi.start = c(draws[3, 6:10],0),
  burnin = 10000, n.draws = 100000, delta.tune = diag(.002, 5),
  psi.tune = diag(.0004, 6), constrained.single = "intercept")

bayesSingleInterceptOnly <- as.list(bayesDraws.1, bayesDraws.2, bayesDraws.3)

summary(bayesSingleInterceptOnly)

## Multiple sensitive item design

## Constrained (estimated control item count not included in sensitive fit)

mle.estimateds.multi <- ictreg(y ~ male + college + age + south, data = multi,
  constrained = TRUE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimateds.multi),
  Sigma = vcov(mle.estimateds.multi) * 9)

bayesMultiDraws.1 <- ictregBayes(y ~ male + college + age + south,
  data = multi, delta.start = list(draws[1, 6:10], draws[1, 11:15]),
  psi.start = draws[1, 1:5], burnin = 10000, n.draws = 100000,
  delta.tune = diag(.002, 5), psi.tune = diag(.001, 5),
  constrained.multi = TRUE)

bayesMultiDraws.2 <- ictregBayes(y ~ male + college + age + south,
  data = multi, delta.start = list(draws[2, 6:10], draws[2, 11:15]),
  psi.start = draws[2, 1:5], burnin = 10000, n.draws = 100000,
  delta.tune = diag(.002, 5), psi.tune = diag(.001, 5),
  constrained.multi = TRUE)

bayesMultiDraws.3 <- ictregBayes(y ~ male + college + age + south,
  data = multi, delta.start = list(draws[3, 6:10], draws[3, 11:15]),
  psi.start = draws[3, 1:5], burnin = 10000, n.draws = 100000,
  delta.tune = diag(.002, 5), psi.tune = diag(.001, 5),
  constrained.multi = TRUE)

```

```

bayesMultiConstrained <- as.list(bayesMultiDraws.1, bayesMultiDraws.2,
  bayesMultiDraws.3)

summary(bayesMultiConstrained)

## Unconstrained (estimated control item count is included in sensitive fit)

mle.estimate.multi <- ictreg(y ~ male + college + age + south, data = multi,
  constrained = FALSE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimate.multi),
  Sigma = vcov(mle.estimate.multi) * 9)

bayesMultiDraws.1 <- ictregBayes(y ~ male + college + age + south,
  data = multi, delta.start = list(draws[1, 6:10], draws[1, 11:15]),
  psi.start = draws[1, 1:5], burnin = 50000, n.draws = 300000,
  delta.tune = diag(.0085, 6), psi.tune = diag(.00025, 5),
  constrained.multi = FALSE)

bayesMultiDraws.2 <- ictregBayes(y ~ male + college + age + south,
  data = multi, delta.start = list(draws[2, 6:10], draws[2, 11:15]),
  psi.start = draws[2, 1:5], burnin = 50000, n.draws = 300000,
  delta.tune = diag(.0085, 6), psi.tune = diag(.00025, 5),
  constrained.multi = FALSE)

bayesMultiDraws.3 <- ictregBayes(y ~ male + college + age + south,
  data = multi, delta.start = list(draws[3, 6:10], draws[3, 11:15]),
  psi.start = draws[3, 1:5], burnin = 50000, n.draws = 300000,
  delta.tune = diag(.0085, 6), psi.tune = diag(.00025, 5),
  constrained.multi = FALSE)

bayesMultiUnconstrained <- as.list(bayesMultiDraws.1, bayesMultiDraws.2,
  bayesMultiDraws.3)

summary(bayesMultiUnconstrained)

## Mixed effects models

## Varying intercepts

mle.estimate <- ictreg(y ~ male + college + age + south, data = race)

draws <- mvrnorm(n = 3, mu = coef(mle.estimate),
  Sigma = vcov(mle.estimate) * 9)

bayesDraws.1 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[1, 1:5], psi.start = draws[1, 6:10], burnin = 100,
  n.draws = 1000, delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5),
  constrained.single = "full", group.mixed = "state", formula.mixed = ~ 1)

bayesDraws.2 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[2, 1:5], psi.start = draws[2, 6:10], burnin = 10000,

```

```

n.draws = 100000, delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5),
constrained.single = "full", group.mixed = "state", formula.mixed = ~ 1)

bayesDraws.3 <- ictregBayes(y ~ male + college + age + south, data = race,
  delta.start = draws[3, 1:5], psi.start = draws[3, 6:10], burnin = 10000,
  n.draws = 100000, delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5),
  constrained.single = "full", group.mixed = "state", formula.mixed = ~ 1)

bayesMixed <- as.list(bayesDraws.1, bayesDraws.2, bayesDraws.3)

summary(bayesMixed)

## End(Not run)

```

ictregBayesHier *Item Count Technique*

Description

Function to conduct multilevel, multivariate regression analyses of survey data with the item count technique, also known as the list experiment and the unmatched count technique.

Usage

```

ictregBayesHier(formula, data = parent.frame(),
  group.level.2, group.level.3, group.level.4,
  formula.level.3, formula.level.4,
  treat = "treat", J, fit.start = "lm",
  n.draws = 10000, burnin = 5000, thin = 0,
  delta.start.level.1, delta.mu0.level.1, delta.A0.level.1,
  delta.start.level.2, delta.mu0.level.2, delta.A0.level.2,
  delta.start.level.3, delta.mu0.level.3, delta.A0.level.3,
  delta.start.level.4, delta.mu0.level.4, delta.A0.level.4,
  sigma.start.level.1, sigma.df.level.1, sigma.scale.level.1,
  sigma.start.level.2, sigma.df.level.2, sigma.scale.level.2,
  sigma.start.level.3, sigma.df.level.3, sigma.scale.level.3,
  sigma.start.level.4, sigma.df.level.4, sigma.scale.level.4,
  delta.tune, alpha.tune,
  verbose = TRUE, ...)

```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted.
data	A data frame containing the variables in the model
group.level.2	Name of second level group variable from the data frame indicating which group each individual belongs to as a string

group.level.3	Name of third level group variable from the data frame indicating which group each individual belongs to as a string
group.level.4	Name of fourth level group variable from the data frame indicating which group each individual belongs to as a string
formula.level.2	An object of class "formula" for the second level of the hierarchical model
formula.level.3	An object of class "formula" for the third level of the hierarchical model
formula.level.4	An object of class "formula" for the fourth level of the hierarchical model
treat	Name of treatment indicator as a string. For single sensitive item models, this refers to a binary indicator, and for multiple sensitive item models it refers to a multi-valued variable with zero representing the control condition. This can be an integer (with 0 for the control group) or a factor (with "control" for the control group).
J	Number of non-sensitive (control) survey items. This will be set automatically to the maximum value of the outcome variable in the treatment group if no input is sent by the user.
fit.start	Fit method for starting values. The options are <code>lm</code> , <code>glm</code> , <code>nls</code> , and <code>ml</code> , which use OLS, logistic regression, non-linear least squares, and maximum likelihood estimation to generate starting values, respectively. The default is <code>lm</code> .
n.draws	Number of MCMC iterations after the burnin.
burnin	The number of initial MCMC iterations that are discarded.
thin	The interval of thinning, in which every other (<code>thin = 1</code>) or more iterations are discarded in the output object
delta.start.level.1	Optional starting values for the sensitive item fit. This should be a vector with the length of the number of covariates for the single sensitive item design, and either a vector or a list with a vector of starting values for each of the sensitive items. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
delta.mu0.level.1	Optional vector of prior means for the sensitive item fit parameters, a vector of length the number of covariates.
delta.A0.level.1	Optional matrix of prior precisions for the sensitive item fit parameters, a matrix of dimension the number of covariates.
delta.start.level.2	Optional starting values for the sensitive item fit for the second level of the hierarchical model. This should be a vector with the length of the number of covariates for the single sensitive item design, and either a vector or a list with a vector of starting values for each of the sensitive items. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
delta.mu0.level.2	Optional vector of prior means for the sensitive item fit parameters for the second level of the hierarchical model, a vector of length the number of covariates.

- `delta.A0.level.2`
Optional matrix of prior precisions for the sensitive item fit parameters for the second level of the hierarchical model, a matrix of dimension the number of covariates.
- `sigma.start.level.1`
Optional list of length the number of sensitive items with the starting values for the sigma parameters.
- `sigma.scale.level.1`
Optional prior scale parameter.
- `sigma.df.level.1`
Optional prior degrees of freedom parameter.
- `sigma.start.level.2`
Optional list of length the number of sensitive items with the starting values for the sigma parameters for the second level of the hierarchical model.
- `sigma.scale.level.2`
Optional prior scale parameter for the second level of the hierarchical model.
- `sigma.df.level.2`
Optional prior degrees of freedom parameter for the second level of the hierarchical model.
- `sigma.start.level.3`
Optional list of length the number of sensitive items with the starting values for the sigma parameters for the third level of the hierarchical model.
- `sigma.scale.level.3`
Optional prior scale parameter for the third level of the hierarchical model.
- `sigma.df.level.3`
Optional prior degrees of freedom parameter for the third level of the hierarchical model.
- `sigma.start.level.4`
Optional list of length the number of sensitive items with the starting values for the sigma parameters for the fourth level of the hierarchical model.
- `sigma.scale.level.4`
Optional prior scale parameter for the fourth level of the hierarchical model.
- `sigma.df.level.4`
Optional prior degrees of freedom parameter for the fourth level of the hierarchical model.
- `delta.start.level.3`
Optional starting values for the sensitive item fit for the third level of the hierarchical model. This should be a vector with the length of the number of covariates for the single sensitive item design, and either a vector or a list with a vector of starting values for each of the sensitive items. The default runs an `ictreg` fit with the method set by the `fit.start` option.
- `delta.mu0.level.3`
Optional vector of prior means for the sensitive item fit parameters for the third level of the hierarchical model, a vector of length the number of covariates.
- `delta.A0.level.3`
Optional matrix of prior precisions for the sensitive item fit parameters for the third level of the hierarchical model, a matrix of dimension the number of covariates.

<code>delta.start.level.4</code>	Optional starting values for the sensitive item fit for the fourth level of the hierarchical model. This should be a vector with the length of the number of covariates for the single sensitive item design, and either a vector or a list with a vector of starting values for each of the sensitive items. The default runs an <code>ictreg</code> fit with the method set by the <code>fit.start</code> option.
<code>delta.mu0.level.4</code>	Optional vector of prior means for the sensitive item fit parameters for the fourth level of the hierarchical model, a vector of length the number of covariates.
<code>delta.A0.level.4</code>	Optional matrix of prior precisions for the sensitive item fit parameters for the fourth level of the hierarchical model, a matrix of dimension the number of covariates.
<code>delta.tune</code>	A required vector of tuning parameters for the Metropolis algorithm for the sensitive item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
<code>alpha.tune</code>	An optional vector of tuning parameters for the Metropolis algorithm for the random effects.
<code>verbose</code>	A logical value indicating whether model diagnostics are printed out during fitting.
<code>...</code>	further arguments to be passed to NLS regression commands.

Details

This function allows the user to perform regression analysis on data from the item count technique, also known as the list experiment and the unmatched count technique using a Bayesian MCMC algorithm.

Unlike the maximum likelihood and least squares estimators in the `ictreg` function, the Metropolis algorithm for the Bayesian MCMC estimators in this function must be tuned to work correctly. The `delta.tune` and `psi.tune` are required, and the values, one for each estimated parameter, will need to be manipulated. The output of the `ictregBayes` function, and of the `summary` function run on an `ictregBayes` object display the acceptance ratios from the Metropolis algorithm. If these values are far from 0.4, the tuning parameters should be changed until the ratios approach 0.4.

For the single sensitive item design, the model can constrain all control parameters to be equal (`constrained = "full"`), or just the intercept (`constrained = "intercept"`) or all the control fit parameters can be allowed to vary across the potential sensitive item values (`constrained = "none"`).

For the multiple sensitive item design, the model can include the estimated number of affirmative responses to the control items as a covariate in the sensitive item model fit (`constrained` set to `TRUE`) or exclude it (`FALSE`).

Convergence is at times difficult to achieve, so we recommend running multiple chains from overdispersed starting values by, for example, running an MLE or linear model using the `ictreg()` function, and then generating a set of overdispersed starting values using those estimates and their estimated variance-covariance matrix. An example is provided below for each of the possible designs. Running `summary()` after such a procedure will output the Gelman-Rubin convergence statistics in addition to the estimates. If the G-R statistics are all below 1.1, the model is said to have converged.

Value

ictregBayes returns an object of class "ictregBayes". The function summary is used to obtain a table of the results, using the coda package. Two attributes are also included, the data ("x"), the call ("call"), which can be extracted using the command, e.g., attr(ictregBayes.object, "x").

mcmc	an object of class "mcmc" that can be analyzed using the coda package.
x	the design matrix
multi	a logical value indicating whether the data included multiple sensitive items.
constrained	a logical or character value indicating whether the control group parameters are constrained to be equal in the single sensitive item design, and whether the non-sensitive item count is included as a predictor in the sensitive item fits for the multiple sensitive item design.
delta.start	Optional starting values for the sensitive item fit. This should be a vector with the length of the number of covariates. The default runs an ictreg fit with the method set by the fit.start option.
psi.start	Optional starting values for the control items fit. This should be a vector of length the number of covariates. The default runs an ictreg fit with the method set by the fit.start option.
delta.mu0	Optional vector of prior means for the sensitive item fit parameters, a vector of length the number of covariates.
psi.mu0	Optional vector of prior means for the control item fit parameters, a vector of length the number of covariates.
delta.A0	Optional matrix of prior precisions for the sensitive item fit parameters, a matrix of dimension the number of covariates.
psi.A0	Optional matrix of prior precisions for the control items fit parameters, a matrix of dimension the number of covariates.
delta.tune	A required vector of tuning parameters for the Metropolis algorithm for the sensitive item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
psi.tune	A required vector of tuning parameters for the Metropolis algorithm for the control item fit. This must be set and refined by the user until the acceptance ratios are approximately .4 (reported in the output).
J	Number of non-sensitive (control) survey items set by the user or detected.
treat.labels	a vector of the names used by the treat vector for the sensitive item or items. This is the names from the treat indicator if it is a factor, or the number of the item if it is numeric.
control.label	a vector of the names used by the treat vector for the control items. This is the names from the treat indicator if it is a factor, or the number of the item if it is numeric.
call	the matched call

If the data includes multiple sensitive items, the following object is also included:

treat.values	a vector of the values used in the treat vector for the sensitive items, either character or numeric depending on the class of treat. Does not include the value for the control status
--------------	---

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[predict.ictreg](#) for fitted values

Examples

```
data(race)

## Not run:

## Multiple chain MCMC list experiment regression
## starts with overdispersed MLE starting values

## Multiple item two level hierarchical model - varying intercepts

mle.estimates.multi <- ictreg(y ~ male + college, data = multi,
  constrained = TRUE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimates.multi),
  Sigma = vcov(mle.estimates.multi) * 9)

bayesDraws.1 <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ 1,
  delta.start.level.1 = list(draws[1, 8:9], draws[1, 2:3], draws[1, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100000, burnin = 50000, thin = 100)

bayesDraws.2 <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ 1,
  delta.start.level.1 = list(draws[2, 8:9], draws[2, 2:3], draws[2, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100000, burnin = 50000, thin = 100)
```

```

bayesDraws.3 <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ 1,
  delta.start.level.1 = list(draws[3, 8:9], draws[3, 2:3], draws[3, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100000, burnin = 50000, thin = 100)

bayesHierTwoLevel <- as.list(bayesDraws.1, bayesDraws.2, bayesDraws.3)

summary(bayesHierTwoLevel)

## Multiple item two level hierarchical model - including covariates

mle.estimate.multi <- ictreg(y ~ male + college, data = multi,
  constrained = TRUE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimate.multi),
  Sigma = vcov(mle.estimate.multi) * 9)

bayesDraws.1 <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ age,
  delta.start.level.1 = list(draws[1, 8:9], draws[1, 2:3], draws[1, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100000, burnin = 50000, thin = 100)

bayesDraws.2 <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ age,
  delta.start.level.1 = list(draws[2, 8:9], draws[2, 2:3], draws[2, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100000, burnin = 50000, thin = 100)

bayesDraws.3 <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ age,
  delta.start.level.1 = list(draws[3, 8:9], draws[3, 2:3], draws[3, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100000, burnin = 50000, thin = 100)

bayesHierTwoLevel <- as.list(bayesDraws.1, bayesDraws.2, bayesDraws.3)

summary(bayesHierTwoLevel)

```

```
## End(Not run)
```

 mis

The 1994 Multi-Investigator Survey

Description

This dataset is a subset of the 1994 Multi-Investigator Survey and contains the item count technique or the list experiment. The main question reads as follows: Now I'm going to read you four things that sometimes make people angry or upset. After I read all (three/four), just tell me HOW MANY of them upset you. (I don't want to know which ones, just how many.) (1) "the federal government increasing the tax on gasoline;" (2) "professional athletes getting million-dollar-plus salaries;" (3) "requiring seatbelts be used when driving;" (4) "large corporations polluting the environment;" (5) "black leaders asking the government for affirmative action."

where the last item is presented only with the treatment group and the control list only contains the first three items.

The survey also includes a question in which attitudes toward the sensitive item are asked directly.

Now I'm going to ask you about another thing that sometimes makes people angry or upset. Do you get angry or upset when black leaders ask the government for affirmative action?

Usage

```
data(mis)
```

Format

A data frame containing the following 6 variables for 1171 observations.

y	numeric	the number of items that make respondents angry	0 - 4
sensitive	numeric	whether or not the sensitive item (asked directly) makes respondents angry	0 - 1
south	numeric	whether or not a respondents live in a southern state	0 - 1
male	numeric	whether or not a respondent is male	0 - 1
college	numeric	whether or not a respondent attended some college	0 - 1
age	numeric	age of a respondent divided by 10	
democrat	numeric	whether not a respondent identifies as a Democrat	0 - 1
republican	numeric	whether not a respondent identifies as a Republican	0 - 1
independent	numeric	whether not a respondent identifies as an independent	0 - 1
treat	numeric	treatment status	0 - 1
list.data	numeric	indicator for list experiment subset (treatment and control groups)	0 - 1
sens.data	numeric	indicator for direct sensitive item subset	0 - 1

Source

The full data set is available at SDA (Survey Documentation and Analysis; <http://sda.berkeley.edu/D3/Multi/Doc/mult.htm>)

 multi

The 1991 National Race and Politics Survey

Description

This dataset is a subset of the 1991 National Race and Politics Survey and contains a list experiment with two sensitive items. The main questions read as follows: Now I'm going to read you four things that sometimes make people angry or upset. After I read all (three/four), just tell me HOW MANY of them upset you. (I don't want to know which ones, just how many.) (1) "the federal government increasing the tax on gasoline;" (2) "professional athletes getting million-dollar-plus salaries;" (3) "large corporations polluting the environment;" (4) "a black family moving next door to you."

where the last item is presented only with the treatment group and the control list only contains the first three items.

The second sensitive item replaces item (4) with (4) "black leaders asking the government for affirmative action."

Treatment status one (`treat == 1`) is the "black family" item and status two is the "affirmative action" item.

Usage

```
data(race)
```

Format

A data frame containing the following 6 variables for 1795 observations.

y	numeric	the number of items that make respondents angry	0 - 4
south	numeric	whether or not a respondents live in a southern state	0 - 1
male	numeric	whether or not a respondent is male	0 - 1
college	numeric	whether or not a respondent attended some college	0 - 1
age	numeric	age of a respondent divided by 10	
treat	numeric	treatment status	0 - 2

Source

The full data set is available at SDA (Survey Documentation and Analysis; <http://sda.berkeley.edu/D3/Natlrace/Doc/nrac.htm>)

plot.predict.ictreg *Plot Method for the Item Count Technique*

Description

Function to plot predictions and confidence intervals of predictions from estimates from multivariate regression analysis of survey data with the item count technique.

Usage

```
## S3 method for class 'predict.ictreg'
plot(x, labels = NA, axes.ict = TRUE,
      xlim = NULL, ylim = NULL, xlab = NULL,
      ylab = "Estimated Proportion",
      axes = F, pch = 19, xvec = NULL, ...)
```

Arguments

x	object or set of objects of class inheriting from "predict.ictreg". Either a single object from an ictreg() model fit or multiple predict objects combined with the c() function.
labels	a vector of labels for each prediction, plotted at the x axis.
axes.ict	a switch indicating if custom plot axes are to be used with the user-provided estimate labels.
xlim	a title for the y axis.
ylim	a title for the y axis.
xlab	a title for the x axis.
ylab	a title for the y axis.
axes	an indicator for whether default plot axes are included.
pch	either an integer specifying a symbol or a single character to be used as the default in plotting points.
xvec	a vector of x values at which the proportions will be printed.
...	Other graphical parameters to be passed to the plot() command are accepted.

Details

plot.predict.ictreg produces plots with estimated population proportions of respondents answering the sensitive item in a list experiment in the affirmative, with confidence intervals.

The function accepts a set of predict.ictreg objects calculated in the following manner:

```
predict(ictreg.object, avg = TRUE, interval = "confidence")
```

For each average prediction, a point estimate and its confidence interval is plotted at equally spaced intervals. The x location of the points can be manipulated with the `xvec` option.

Either a single predict object can be plotted, or a group of them combined with `c(predict.object1, predict.object2)`. Predict objects with the `newdata.diff` option, which calculates the mean difference in probability between two datasets, and the `direct.glm` option, which calculates the mean difference between the mean predicted support for the sensitive item in the list experiment and in a direct survey item, can also be plotted in the same way as other predict objects.

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[ictreg](#) for model fitting and [predict.ictreg](#) for predictions based on the model fits.

Examples

```
data(race)
race.south <- race.nonsouth <- race
race.south[, "south"] <- 1
race.nonsouth[, "south"] <- 0

## Not run:

# Fit EM algorithm ML model with constraint
ml.constrained.results <- ictreg(y ~ south + age + male + college,
  data = race, treat = "treat", J=3, method = "ml",
  overdispersed = FALSE, constrained = TRUE)

# Calculate average predictions for respondents in the South
# and the the North of the US for the MLE model, replicating the
# estimates presented in Figure 1, Imai (2011)
avg.pred.south.mle <- predict(ml.constrained.results,
  newdata = race.south, avg = TRUE, interval = "confidence")
avg.pred.nonsouth.mle <- predict(ml.constrained.results,
  newdata = race.nonsouth, avg = TRUE, interval = "confidence")

# A plot of a single estimate and its confidence interval
plot(avg.pred.south.mle, labels = "South")

# A plot of the two estimates and their confidence intervals
```

```
# use c() to combine more than one predict object for plotting
plot(c(avg.pred.south.mle, avg.pred.nonsouth.mle), labels = c("South", "Non-South"))

# The difference option can also be used to simultaneously
# calculate separate estimates of the two sub-groups
# and the estimated difference. This can also be plotted.

avg.pred.diff.mle <- predict(ml.constrained.results,
  newdata = race.south, newdata.diff = race.nonsouth,
  se.fit = TRUE, avg = TRUE)

plot(avg.pred.diff.mle)

# Social desirability bias plots

# Estimate logit for direct sensitive question

data(mis)

mis.list <- subset(mis, list.data == 1)

mis.sens <- subset(mis, sens.data == 1)

# Fit EM algorithm ML model

fit.list <- ictreg(y ~ age + college + male + south,
  J = 4, data = mis.list, method = "ml")

# Fit logistic regression with directly-asked sensitive question

fit.sens <- glm(sensitive ~ age + college + male + south,
  data = mis.sens, family = binomial("logit"))

# Predict difference between response to sensitive item
# under the direct and indirect questions (the list experiment).
# This is an estimate of the revealed social desirability bias
# of respondents. See Blair and Imai (2010).

avg.pred.social.desirability <- predict(fit.list,
  direct.glm = fit.sens, se.fit = TRUE)

plot(avg.pred.social.desirability)

## End(Not run)
```

Description

Function to calculate predictions and uncertainties of predictions from estimates from multivariate regression analysis of survey data with the item count technique.

Usage

```
## S3 method for class 'ictreg'
predict(object, newdata, newdata.diff, direct.glm, se.fit = FALSE,
        interval = c("none", "confidence"), level = .95, avg = FALSE, sensitive.item, ...)
```

Arguments

<code>object</code>	Object of class inheriting from "ictreg"
<code>newdata</code>	An optional data frame containing data that will be used to make predictions from. If omitted, the data used to fit the regression are used.
<code>newdata.diff</code>	An optional data frame used to compare predictions with predictions from the data in the provided <code>newdata</code> data frame.
<code>direct.glm</code>	A glm object from a logistic binomial regression predicting responses to a direct survey item regarding the sensitive item. The predictions from the <code>ictreg</code> object are compared to the predictions based on this glm object.
<code>se.fit</code>	A switch indicating if standard errors are required.
<code>interval</code>	Type of interval calculation.
<code>level</code>	Significance level for confidence intervals.
<code>avg</code>	A switch indicating if the mean prediction and associated statistics across all observations in the dataframe will be returned instead of predictions for each observation.
<code>sensitive.item</code>	For multiple sensitive item design list experiments, specify which sensitive item fits to use for predictions. Default is the first sensitive item.
<code>...</code>	further arguments to be passed to or from other methods.

Details

`predict.ictreg` produces predicted values, obtained by evaluating the regression function in the frame `newdata` (which defaults to `model.frame(object)`). If the logical `se.fit` is TRUE, standard errors of the predictions are calculated. Setting `interval` specifies computation of confidence intervals at the specified level or no intervals.

If `avg` is set to TRUE, the mean prediction across all observations in the dataset will be calculated, and if the `se.fit` option is set to TRUE a standard error for this mean estimate will be provided. The `interval` option will output confidence intervals instead of only the point estimate if set to TRUE.

Two additional types of mean prediction are also available. The first, if a `newdata.diff` data frame is provided by the user, calculates the mean predicted values across two datasets, as well as the mean difference in predicted value. Standard errors and confidence intervals can also be added. For difference prediction, `avg` must be set to TRUE.

The second type of prediction, triggered if a `direct.glm` object is provided by the user, calculates the mean difference in prediction between predictions based on an `ictreg` fit and a `glm` fit from a direct survey item on the sensitive question. This is defined as the revealed social desirability bias in Blair and Imai (2010).

Value

`predict.ictreg` produces a vector of predictions or a matrix of predictions and bounds with column names `fit`, `lwr`, and `upr` if `interval` is set. If `se.fit` is `TRUE`, a list with the following components is returned:

<code>fit</code>	vector or matrix as above
<code>se.fit</code>	standard error of prediction

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[ictreg](#) for model fitting

Examples

```
data(race)

race.south <- race.nonsouth <- race

race.south[, "south"] <- 1
race.nonsouth[, "south"] <- 0

## Not run:

# Fit EM algorithm ML model with constraint with no covariates

ml.results.south.nocov <- ictreg(y ~ 1,
  data = race[race$south == 1, ], method = "ml", treat = "treat",
  J = 3, overdispersed = FALSE, constrained = TRUE)
ml.results.nonsouth.nocov <- ictreg(y ~ 1,
  data = race[race$south == 0, ], method = "ml", treat = "treat",
  J = 3, overdispersed = FALSE, constrained = TRUE)
```

```
# Calculate average predictions for respondents in the South
# and the the North of the US for the MLE no covariates
# model, replicating the estimates presented in Figure 1,
# Imai (2010)

avg.pred.south.nocov <- predict(ml.results.south.nocov,
  newdata = as.data.frame(matrix(1, 1, 1)), se.fit = TRUE,
  avg = TRUE)
avg.pred.nonsouth.nocov <- predict(ml.results.nonsouth.nocov,
  newdata = as.data.frame(matrix(1, 1, 1)), se.fit = TRUE,
  avg = TRUE)

# Fit linear regression

lm.results <- ictreg(y ~ south + age + male + college,
  data = race, treat = "treat", J=3, method = "lm")

# Calculate average predictions for respondents in the
# South and the the North of the US for the lm model,
# replicating the estimates presented in Figure 1, Imai (2010)

avg.pred.south.lm <- predict(lm.results, newdata = race.south,
  se.fit = TRUE, avg = TRUE)

avg.pred.nonsouth.lm <- predict(lm.results, newdata = race.nonsouth,
  se.fit = TRUE, avg = TRUE)

# Fit two-step non-linear least squares regression

nls.results <- ictreg(y ~ south + age + male + college,
  data = race, treat = "treat", J=3, method = "nls")

# Calculate average predictions for respondents in the South
# and the the North of the US for the NLS model, replicating
# the estimates presented in Figure 1, Imai (2010)

avg.pred.nls <- predict(nls.results, newdata = race.south,
  newdata.diff = race.nonsouth, se.fit = TRUE, avg = TRUE)

# Fit EM algorithm ML model with constraint

ml.constrained.results <- ictreg(y ~ south + age + male + college,
  data = race, treat = "treat", J=3, method = "ml",
  overdispersed = FALSE, constrained = TRUE)

# Calculate average predictions for respondents in the South
# and the the North of the US for the MLE model, replicating the
# estimates presented in Figure 1, Imai (2010)

avg.pred.diff.mle <- predict(ml.constrained.results,
  newdata = race.south, newdata.diff = race.nonsouth,
  se.fit = TRUE, avg = TRUE)
```

```

# Calculate average predictions from the item count technique
# regression and from a direct sensitive item modeled with
# a logit.

# Estimate logit for direct sensitive question

data(mis)

mis.list <- subset(mis, list.data == 1)

mis.sens <- subset(mis, sens.data == 1)

# Fit EM algorithm ML model

fit.list <- ictreg(y ~ age + college + male + south,
  J = 4, data = mis.list, method = "ml")

# Fit logistic regression with directly-asked sensitive question

fit.sens <- glm(sensitive ~ age + college + male + south,
  data = mis.sens, family = binomial("logit"))

# Predict difference between response to sensitive item
# under the direct and indirect questions (the list experiment).
# This is an estimate of the revealed social desirability bias
# of respondents. See Blair and Imai (2010).

avg.pred.social.desirability <- predict(fit.list,
  direct.glm = fit.sens, se.fit = TRUE)

## End(Not run)

```

predict.ictregBayes *Predict Method for the Item Count Technique with Bayesian MCMC*

Description

Function to calculate predictions and uncertainties of predictions from estimates from multivariate regression analysis of survey data with the item count technique.

Usage

```

## S3 method for class 'ictregBayes'
predict(object, newdata, newdata.diff, direct.glm, se.fit = FALSE,
  interval = c("none", "confidence"), level = .95, sensitive.item, ...)

```

Arguments

<code>object</code>	Object of class inheriting from "ictregBayes" or "ictregBayesMulti"
<code>newdata</code>	An optional data frame containing data that will be used to make predictions from. If omitted, the data used to fit the regression are used.
<code>newdata.diff</code>	An optional data frame used to compare predictions with predictions from the data in the provided <code>newdata</code> data frame.
<code>direct.glm</code>	A <code>glm</code> object from a logistic binomial regression predicting responses to a direct survey item regarding the sensitive item. The predictions from the <code>ictreg</code> object are compared to the predictions based on this <code>glm</code> object.
<code>se.fit</code>	A switch indicating if standard errors are required.
<code>interval</code>	Type of interval calculation.
<code>level</code>	Significance level for confidence intervals.
<code>sensitive.item</code>	For the multiple sensitive item design, the integer indicating which sensitive item coefficients will be used for prediction.
<code>...</code>	further arguments to be passed to or from other methods.

Details

`predict.ictregBayes` produces predicted values, obtained by evaluating the regression function in the frame `newdata` (which defaults to `model.frame(object)`). If the logical `se.fit` is `TRUE`, standard errors of the predictions are calculated. Setting `interval` specifies computation of confidence intervals at the specified level or no intervals.

The mean prediction across all observations in the dataset is calculated, and if the `se.fit` option is set to `TRUE` a standard error for this mean estimate will be provided. The `interval` option will output confidence intervals instead of only the point estimate if set to `TRUE`.

Two additional types of mean prediction are also available. The first, if a `newdata.diff` data frame is provided by the user, calculates the mean predicted values across two datasets, as well as the mean difference in predicted value. Standard errors and confidence intervals can also be added. For difference prediction, `avg` must be set to `TRUE`.

The second type of prediction, triggered if a `direct.glm` object is provided by the user, calculates the mean difference in prediction between predictions based on an `ictreg` fit and a `glm` fit from a direct survey item on the sensitive question. This is defined as the revealed social desirability bias in Blair and Imai (2010).

In the multiple sensitive item design, prediction can only be based on the coefficients from one of the sensitive item fits. The `sensitive.item` option allows you to specify which is used, using integers from 1 to the number of sensitive items.

Value

`predict.ictreg` produces a vector of predictions or a matrix of predictions and bounds with column names `fit`, `lwr`, and `upr` if `interval` is set. If `se.fit` is `TRUE`, a list with the following components is returned:

<code>fit</code>	vector or matrix as above
<code>se.fit</code>	standard error of prediction

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[ictreg](#) for model fitting

Examples

```
data(race)

## Not run:

bayes.fit <- ictregBayes(y ~ age + college + male + south, data = multi,
  treat = "treat", delta.tune = diag(.002, 5), psi.tune = diag(.00025, 5))

bayes.predict <- predict(bayes.fit, interval = "confidence", se.fit = TRUE)

## End(Not run)
```

predict.ictregBayesHier

Predict Method for the Item Count Technique with Bayesian Hierarchical Regression

Description

Function to calculate predictions and uncertainties of predictions from estimates from hierarchical multivariate regression analysis of survey data with the item count technique.

Usage

```
## S3 method for class 'ictregBayesHier'
predict(object, newdata, se.fit = FALSE,
  interval = c("none", "confidence"), level = .95, sensitive.item, ...)
```

Arguments

<code>object</code>	Object of class inheriting from "ictregBayes" or "ictregBayesMulti"
<code>newdata</code>	An optional data frame containing data that will be used to make predictions from. If omitted, the data used to fit the regression are used.
<code>se.fit</code>	A switch indicating if standard errors are required.
<code>interval</code>	Type of interval calculation.
<code>level</code>	Significance level for confidence intervals.
<code>sensitive.item</code>	For the multiple sensitive item design, the integer indicating which sensitive item coefficients will be used for prediction.
<code>...</code>	further arguments to be passed to or from other methods.

Details

`predict.ictregBayesHier` produces predicted values, obtained by evaluating the regression function in the frame `newdata` (which defaults to `model.frame(object)`). If the logical `se.fit` is `TRUE`, standard errors of the predictions are calculated. Setting `interval` specifies computation of confidence intervals at the specified level or no intervals.

The mean prediction across all observations in the dataset is calculated, and if the `se.fit` option is set to `TRUE` a standard error for this mean estimate will be provided. The `interval` option will output confidence intervals instead of only the point estimate if set to `TRUE`.

In the multiple sensitive item design, prediction can only be based on the coefficients from one of the sensitive item fits. The `sensitive.item` option allows you to specify which is used, using integers from 1 to the number of sensitive items.

Value

`predict.ictreg` produces a vector of predictions or a matrix of predictions and bounds with column names `fit`, `lwr`, and `upr` if `interval` is set. If `se.fit` is `TRUE`, a list with the following components is returned:

<code>fit</code>	vector or matrix as above
<code>se.fit</code>	standard error of prediction

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

- Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>
- Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[ictreg](#) for model fitting

Examples

```
data(race)

## Not run:

mle.estimateds.multi <- ictreg(y ~ male + college, data = multi,
  constrained = TRUE)

draws <- mvrnorm(n = 3, mu = coef(mle.estimateds.multi),
  Sigma = vcov(mle.estimateds.multi) * 9)

bayes.fit <- ictregBayesHier(y ~ male + college,
  formula.level.2 = ~ 1,
  ceiling = rep(FALSE, 2), floor = rep(FALSE, 2),
  delta.start.level.1 = list(draws[1, 8:9], draws[1, 2:3], draws[1, 5:6]),
  data = multi, treat = "treat",
  delta.tune = list(rep(0.005, 2), rep(0.05, 2), rep(0.05, 2)),
  alpha.tune = rep(0.001, length(unique(multi$state))),
  J = 3, group.level.2 = "state",
  n.draws = 100, burnin = 10, thin = 1)

bayes.predict <- predict(bayes.fit, interval = "confidence", se.fit = TRUE)

## End(Not run)
```

race

The 1991 National Race and Politics Survey

Description

This dataset is a subset of the 1991 National Race and Politics Survey and contains the item count technique or the list experiment. The main question reads as follows: Now I'm going to read you four things that sometimes make people angry or upset. After I read all (three/four), just tell me HOW MANY of them upset you. (I don't want to know which ones, just how many.) (1) "the federal government increasing the tax on gasoline;" (2) "professional athletes getting million-dollar-plus salaries;" (3) "large corporations polluting the environment;" (4) "a black family moving next door to you."

where the last item is presented only with the treatment group and the control list only contains the first three items.

Usage

```
data(race)
```

Format

A data frame containing the following 6 variables for 1213 observations.

y	numeric	the number of items that make respondents angry	0 - 4
south	numeric	whether or not a respondents live in a southern state	0 - 1
male	numeric	whether or not a respondent is male	0 - 1
college	numeric	whether or not a respondent attended some college	0 - 1
age	numeric	age of a respondent divided by 10	
treat	numeric	treatment status	0 - 1

Source

The full data set is available at SDA (Survey Documentation and Analysis; <http://sda.berkeley.edu/D3/Natlrace/Doc/nrac.htm>)

summary.ictreg	<i>Summary Method for the Item Count Technique</i>
----------------	--

Description

Function to summarize results from list experiment regression based on the ictreg() function, and to produce proportions of liars estimates.

Usage

```
## S3 method for class 'ictreg'
summary(object, boundary.proportions = FALSE, n.draws = 10000, ...)
```

Arguments

object	Object of class inheriting from "ictreg"
boundary.proportions	A switch indicating whether, for models with ceiling effects, floor effects, or both (indicated by the floor = TRUE, ceiling = TRUE options in ictreg), the conditional probability of lying and the population proportions of liars are calculated.
n.draws	For quasi-Bayesian approximation based predictions, specify the number of Monte Carlo draws.
...	further arguments to be passed to or from other methods.

Details

`predict.ictreg` produces a summary of the results from an `ictreg` object. It displays the coefficients, standard errors, and fit statistics for any model from `ictreg`.

`predict.ictreg` also produces estimates of the conditional probability of lying and of the population proportion of liars for boundary models from `ictreg()` if `ceiling = TRUE` or `floor = TRUE`.

The conditional probability of lying for the ceiling model is the probability that a respondent with true affirmative views of all the sensitive and non-sensitive items lies and responds negatively to the sensitive item. The conditional probability for the floor model is the probability that a respondent lies to conceal her true affirmative views of the sensitive item when she also holds true negative views of all the non-sensitive items. In both cases, the respondent may believe her privacy is not protected, so may conceal her true affirmative views of the sensitive item.

Author(s)

Graeme Blair, Princeton University, <gblair@princeton.edu> and Kosuke Imai, Princeton University, <kimai@princeton.edu>

References

Blair, Graeme and Kosuke Imai. (2012) "Statistical Analysis of List Experiments." *Political Analysis*, Vol. 20, No 1 (Winter). available at <http://imai.princeton.edu/research/listP.html>

Imai, Kosuke. (2011) "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association*, Vol. 106, No. 494 (June), pp. 407-416. available at <http://imai.princeton.edu/research/list.html>

See Also

[ictreg](#) for model fitting

Examples

```
data(race)
## Not run:
# Fit standard design ML model with ceiling effects
# Replicates Table 7 Columns 3-4 in Blair and Imai (2012)

ceiling.results <- ictreg(y ~ age + college + male + south, treat = "treat",
  J = 3, data = affirm, method = "ml", fit.start = "nls",
  ceiling = TRUE, ceiling.fit = "bayesglm",
  ceiling.formula = ~ age + college + male + south)

# Summarize fit object and generate conditional probability
# of ceiling liars the population proportion of ceiling liars,
# both with standard errors.
# Replicates Table 7 Columns 3-4 last row in Blair and Imai (2012)

summary(ceiling.results, boundary.proportions = TRUE)
```

```
## End(Not run)
```

Index

*Topic **datasets**

affirm, 2
mis, 25
multi, 26
race, 37

*Topic **models**

ict.test, 3
ictreg, 4
ictregBayes, 10
ictregBayesHier, 18
plot.predict.ictreg, 27
predict.ictreg, 29
predict.ictregBayes, 33
predict.ictregBayesHier, 35
summary.ictreg, 38

*Topic **regression**

ict.test, 3
ictreg, 4
ictregBayes, 10
ictregBayesHier, 18
plot.predict.ictreg, 27
predict.ictreg, 29
predict.ictregBayes, 33
predict.ictregBayesHier, 35
summary.ictreg, 38

affirm, 2

ict (ictreg), 4
ict.test, 3
ictreg, 4, 4, 28, 31, 35, 37, 39
ictregBayes, 10
ictregBayesHier, 18

list (ictreg), 4

mis, 25
multi, 26

plot.predict.ictreg, 27
predict.ictreg, 8, 14, 23, 28, 29

predict.ictregBayes, 33
predict.ictregBayesHier, 35

race, 37

summary.ictreg, 38