Package ‘misreport’

February 27, 2017

Title Statistical Analysis of Misreporting on Sensitive Survey Questions

Description Enables investigation of the predictors of misreporting on sensitive survey questions through a multivariate list experiment regression method. The method permits researchers to model whether a survey respondent's answer to the sensitive item in a list experiment is different from his or her answer to an analogous direct question.

Version 0.1.1

Depends R (>= 3.2.0)

Imports numDeriv (>= 2014.2-1), VGAM (>= 1.0-2), mvtnorm (>= 1.0-5)

Suggests knitr, rmarkdown

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Gregory Eady [aut, cre]

Maintainer Gregory Eady <`gregory.eady@gmail.com`>

Repository CRAN

Date/Publication 2017-02-27 08:15:34

R topics documented:

bootListExperiment — *List experiment regression with bootstrapped standard errors*

Description

A wrapper function that makes repeated calls to listExperiment to calculate parameter estimates and standard errors through non-parametric boot-strapping.

Usage

```
bootListExperiment(formula, data, treatment, J, direct = NULL,
  sensitive.response = NULL, outcome = NULL, outcome.trials = NULL,
  outcome.model = "logistic", outcome.constrained = TRUE,
  control.constraint = "partial", misreport.treatment = TRUE,
  weights = NULL, se = TRUE, tolerance = 1e-08, max.iter = 5000,
  n.runs = 1, verbose = TRUE, get.data = FALSE, par.control = NULL,
  par.sensitive = NULL, par.misreport = NULL, par.outcome = NULL,
  par.outcome.aux = NULL, formula.control = NULL,
  formula.sensitive = NULL, formula.misreport = NULL,
  formula.outcome = NULL, boot.iter = 1000, parallel = FALSE,
  n.cores = 2, cluster = NULL)
```

Arguments

`formula`	An object of class "formula": a symbolic description of the model to be fitted.
`data`	A data frame containing the variables to be used in the model.
`treatment`	A string indicating the name of the treatment indicator in the data. This variable must be coded as a binary, where 1 indicates assignment to treatment and 0 indicates assignment to control.
`J`	An integer indicating the number of control items in the list experiment.
`direct`	A string indicating the name of the direct question response in the data. The direct question must be coded as a binary variable. If NULL (default), a misreport sub-model is not fit.
`sensitive.response`	A value 0 or 1 indicating whether the response that is considered sensitive in the list experiment/direct question is 0 or 1.
`outcome`	A string indicating the variable name in the data to use as the outcome in an outcome sub-model. If NULL (default), no outcome sub-model is fit. [*experimental*]
`outcome.trials`	An integer indicating the number of trials in a binomial/betabinomial model if both an outcome sub-model is used and if the argument `outcome.model` is set to "binomial" or "betabinomial". [*experimental*]
`outcome.model`	A string indicating the model type to fit for the outcome sub-model ("logistic", "binomial", "betabinomial"). [*experimental*]

`outcome.constrained`	A logical value indicating whether to constrain U* = 0 in the outcome sub-model. Defaults to TRUE. [*experimental*]
`control.constraint`	A string indicating the constraint to place on Z* and U* in the control-items sub-model: **"none" (default)** Estimate separate parameters for Z* and U*. **"partial"** Constrain U* = 0. **"full"** Constrain U* = Z* = 0.
`misreport.treatment`	A logical value indicating whether to include a parameter for the treatment indicator in the misreport sub-model. Defaults to TRUE.
`weights`	A string indicating the variable name of survey weights in the data (note: standard errors are not currently output when survey weights are used).
`se`	A logical value indicating whether to calculate standard errors. Defaults to TRUE.
`tolerance`	The desired accuracy for EM convergence. The EM loop breaks after the change in the log-likelihood is less than the value of `tolerance`. Defaults to 1e-08.
`max.iter`	The maximum number of iterations for the EM algorithm. Defaults to 10000.
`n.runs`	The total number of times that the EM algorithm is run (can potentially help avoid local maxima). Defaults to 1.
`verbose`	A logical value indicating whether to print information during model fitting. Defaults to TRUE.
`get.data`	For internal use. Used by wrapper function `bootListExperiment`.
`par.control`	A vector of starting parameters for the control-items sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2).
`par.sensitive`	A vector of starting parameters for the sensitive-item sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2).
`par.misreport`	A vector of starting parameters for the misreport sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2).
`par.outcome`	A vector of starting parameters for the outcome sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2). [experimental]
`par.outcome.aux`	A vector of starting parameters for the outcome sub-model in which `outcome.model` is "betabinomial". i.e. c(alpha, beta). If NULL (default), randomly generated starting points are used, drawn from uniform(0, 1). [experimental]
`formula.control`	An object of class "`formula`" used to specify a control-items sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2)

`formula.sensitive`	An object of class "`formula`" used to specify a sensitive-item sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2)
`formula.misreport`	An object of class "`formula`" used to specify a misreport sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2)
`formula.outcome`	An object of class "`formula`" used to specify an outcome sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2) [*experimental*]
`boot.iter`	The number of boot strap samples to generate.
`parallel`	A logical value indicating whether to run bootstraping in parallel on a multi-core computer.
`n.cores`	The number of cores/threads on which to generate bootstrap samples (when `parallel` = TRUE). Defaults to 2.
`cluster`	An optional cluster object using makeCluster() from the `parallel` package (useful if running on an MPI server).

Details

`bootListExperiment` is a wrapper for the function `listExperiment` that allows researchers to fit a bootstrapped model. The arguments for this function include those for the `listExperiment` function, in addition to a small number of arguments specific to the bootstrap.

Value

`listExperiment` returns an object of class "listExperiment". A summary of this object is given using the `summary.listExperiment` function. All components in the "listExperiment" class are listed below.

Slots

`par.control` A named vector of coefficients from the control-items sub-model.

`par.sensitive` A named vector of coefficients from the sensitive-item sub-model.

`par.misreport` A named vector of coefficients from the misreport sub-model.

`par.outcome` A named vector of coefficients from the outcome sub-model.

`par.outcome.aux` A named vector of (auxiliary) coefficients from the outcome sub-model (if `outcome.model` = "betabinomial").

`df` Degrees of freedom.

`se.sensitive` Standard errors for parameters in the sensitive-item sub-model.

`se.control` Standard errors for parameters in the control-items sub-model.

`se.misreport` Standard errors for parameters in the misreport sub-model.

`se.outcome` Standard errors for parameters in the outcome sub-model.

`se.outcome.aux` Standard errors for the auxiliary parameters in the outcome sub-model (if `outcome.model` = "betabinomial").

`vcov.mle` Variance-covariance matrix.

`w` The matrix of posterior predicted probabilities for each observation in the data used for model fitting.

`data` The data frame used for model fitting.

`direct` The string indicating the variable name of the direct question.

`treatment` The string indicating the variable name of the treatment indicator.

`model.misreport` A logical value indicating whether a misreport sub-model was fit.

`outcome.model` The type of model used as the outcome sub-model.

`outcome.constrained` A logical value indicating whether the parameter U* was constrained to 0 in the outcome sub-model.

`control.constraint` A string indicating the constraints placed on the parameters Z* and U* in the control-items sub-model.

`misreport.treatment` A logical value indicating whether a treatment indicator was included in the misreport sub-model.

`weights` A string indicating the variable name of the survey weights.

`formula` The model formula.

`formula.control` The model specification of the control-items sub-model.

`formula.sensitive` The model specification of the sensitive-item sub-model.

`formula.misreport` The model specification of the misreport sub-model.

`formula.outcome` The model specification of the outcome sub-model.

`sensitive.response` The value 0 or 1 indicating the response to the list experiment/direct question that is considered sensitive.

`xlevels` The factor levels of the variables used in the model.

`llik` The model log-likelihood.

`n` The sample size of the data used for model fitting (this value excludes rows removed through listwise deletion).

`J` The number of control items in the list experiment.

`se` A logical value indicating whether standard errors were calculated.

`runs` The parameter estimates from each run of the EM algorithm (note: the parameters that result in the highest log-likelihood are used as the model solution).

`call` The method call.

`boot` A logical value indicating whether non-parametric bootstrapping was used to calculate model parameters and standard errors.

References

Eady, Gregory. 2017 "The Statistical Analysis of Misreporting on Sensitive Survey Questions."

Imai, Kosuke. 2011. "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association* 106 (494): 407-416.

Examples

```
## Simulated list experiment and direct question
n <- 10000
J <- 4

# Covariates
x <- cbind(intercept = rep(1, n), continuous1 = rnorm(n),
           continuous2 = rnorm(n), binary1 = rbinom(n, 1, 0.5))

treatment <- rbinom(n, 1, 0.5)

# Simulate Z*
param_sensitive <- c(0.25, -0.25, 0.5, 0.25)
prob_sensitive <- plogis(x %*% param_sensitive)
true_belief <- rbinom(n, 1, prob = prob_sensitive)

# Simulate whether respondent misreports (U*)
param_misreport <- c(-0.25, 0.25, -0.5, 0.5)
prob_misreport <- plogis(x %*% param_misreport) * true_belief
misreport <- rbinom(n, 1, prob = prob_misreport)

# Simulate control items Y*
param_control <- c(0.25, 0.25, -0.25, 0.25, U = -0.5, Z = 0.25)
prob.control <- plogis(cbind(x, misreport, true_belief) %*% param_control)
control_items <- rbinom(n, J, prob.control)

# List experiment and direct question responses
direct <- true_belief
direct[misreport == 1] <- 0
y <- control_items + true_belief * treatment

A <- data.frame(y, direct, treatment,
                continuous1 = x[, "continuous1"],
                continuous2 = x[, "continuous2"],
                binary1 = x[, "binary1"])

## Not run:
# Note: substantial computation time
model.sim <- bootListExperiment(y ~ continuous1 + continuous2 + binary1,
                                data = A, treatment = "treatment",
                                direct = "direct",
                                J = 4, control.constraint = "none",
                                sensitive.response = 1,
                                boot.iter = 500, parallel = TRUE, n.cores = 2)
summary(model.sim, digits = 3)

## End(Not run)
```

`gender` *List experiment regarding gender and politics*

Description

A dataset containing responses to a list experiment and direct question regarding the statement "women are as competent as men in politics." Data also contain socio-demographics for gender, age, education, region, and political ideology.

Usage

```
gender
```

Format

A data frame with 5000 rows and 7 variables:

y response to the list experiment

treatment treatment assignment

direct response to the direct question

gender gender of respondent {Woman, Man}

age age of respondent {18, 19, ..., 94}

ageGroup age group of respondent {18-29, 30-39, 40-49, 50-64, 65+}

education education of respondent {High school or below, College, University degree}

motherTongue mother tongue of respondent {English, French, Other language}

region region of respondent {Ontario, Atlantic, Quebec, West}

selfPlacement political ideology of respondent (0 = right-wing, 10 = left wing) {0, 1, ..., 10}

weight survey weight

Source

Eady, Gregory. 2016 "The Statistical Analysis of Misreporting on Sensitive Survey Questions."

`listExperiment` *List experiment regression*

Description

Regression analysis for sensitive survey questions using a list experiment and direct question.

Usage

```
listExperiment(formula, data, treatment, J, direct = NULL,
  sensitive.response = NULL, outcome = NULL, outcome.trials = NULL,
  outcome.model = "logistic", outcome.constrained = TRUE,
  control.constraint = "none", misreport.treatment = TRUE, weights = NULL,
  se = TRUE, tolerance = 1e-08, max.iter = 10000, n.runs = 3,
  verbose = TRUE, get.data = FALSE, par.control = NULL,
  par.sensitive = NULL, par.misreport = NULL, par.outcome = NULL,
  par.outcome.aux = NULL, formula.control = NULL,
  formula.sensitive = NULL, formula.misreport = NULL,
  formula.outcome = NULL, get.boot = 0, ...)
```

Arguments

`formula`	An object of class "`formula`": a symbolic description of the model to be fitted.
`data`	A data frame containing the variables to be used in the model.
`treatment`	A string indicating the name of the treatment indicator in the data. This variable must be coded as a binary, where 1 indicates assignment to treatment and 0 indicates assignment to control.
`J`	An integer indicating the number of control items in the list experiment.
`direct`	A string indicating the name of the direct question response in the data. The direct question must be coded as a binary variable. If NULL (default), a misreport sub-model is not fit.
`sensitive.response`	A value 0 or 1 indicating whether the response that is considered sensitive in the list experiment/direct question is 0 or 1.
`outcome`	A string indicating the variable name in the data to use as the outcome in an outcome sub-model. If NULL (default), no outcome sub-model is fit. [*experimental*]
`outcome.trials`	An integer indicating the number of trials in a binomial/betabinomial model if both an outcome sub-model is used and if the argument `outcome.model` is set to "binomial" or "betabinomial". [*experimental*]
`outcome.model`	A string indicating the model type to fit for the outcome sub-model ("logistic", "binomial", "betabinomial"). [*experimental*]
`outcome.constrained`	A logical value indicating whether to constrain U* = 0 in the outcome sub-model. Defaults to TRUE. [*experimental*]

`control.constraint`
: A string indicating the constraint to place on Z* and U* in the control-items sub-model:

  "none" (default) Estimate separate parameters for Z* and U*.

  "partial" Constrain U* = 0.

  "full" Constrain U* = Z* = 0.

`misreport.treatment`
: A logical value indicating whether to include a parameter for the treatment indicator in the misreport sub-model. Defaults to TRUE.

`weights`
: A string indicating the variable name of survey weights in the data (note: standard errors are not currently output when survey weights are used).

`se`
: A logical value indicating whether to calculate standard errors. Defaults to TRUE.

`tolerance`
: The desired accuracy for EM convergence. The EM loop breaks after the change in the log-likelihood is less than the value of `tolerance`. Defaults to 1e-08.

`max.iter`
: The maximum number of iterations for the EM algorithm. Defaults to 10000.

`n.runs`
: The total number of times that the EM algorithm is run (can potentially help avoid local maxima). Defaults to 1.

`verbose`
: A logical value indicating whether to print information during model fitting. Defaults to TRUE.

`get.data`
: For internal use. Used by wrapper function `bootListExperiment`.

`par.control`
: A vector of starting parameters for the control-items sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2).

`par.sensitive`
: A vector of starting parameters for the sensitive-item sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2).

`par.misreport`
: A vector of starting parameters for the misreport sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2).

`par.outcome`
: A vector of starting parameters for the outcome sub-model. Must be in the order of the parameters in the resulting regression output. If NULL (default), randomly generated starting points are used, drawn from uniform(-2, 2). [experimental]

`par.outcome.aux`
: A vector of starting parameters for the outcome sub-model in which `outcome.model` is "betabinomial". i.e. c(alpha, beta). If NULL (default), randomly generated starting points are used, drawn from uniform(0, 1). [experimental]

`formula.control`
: An object of class "`formula`" used to specify a control-items sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2)

`formula.sensitive`
: An object of class "`formula`" used to specify a sensitive-item sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2)

`formula.misreport`	An object of class "`formula`" used to specify a misreport sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2)
`formula.outcome`	An object of class "`formula`" used to specify an outcome sub-model that is different from that given in `formula`. (e.g. ~ x1 + x2) [*experimental*]
`get.boot`	For internal use. An integer, which if greater than 0 requests that `listExperiment()` generate a non-parametric bootstrap sample and fit a model to that sample. Used by the function `bootListExperiment`.
`...`	Additional options.

Details

The `listExperiment` function allows researchers to fit a model for a list experiment and direct question simultaneously, as described in Eady (2017). The primary aim of the function is to allow researchers to model the probability that respondents provides one response to the sensitive item in a list experiment but respond otherwise when asked about the same sensitive item on a direct question. When a direct question response is excluded from the function, the model is functionally equivalent to that proposed by Imai (2011), as implemented as the `ictreg` function in the `list` package (`https://CRAN.R-project.org/package=list`).

Value

`listExperiment` returns an object of class "listExperiment". A summary of this object is given using the `summary.listExperiment` function. All components in the "listExperiment" class are listed below.

Slots

`par.control` A named vector of coefficients from the control-items sub-model.

`par.sensitive` A named vector of coefficients from the sensitive-item sub-model.

`par.misreport` A named vector of coefficients from the misreport sub-model.

`par.outcome` A named vector of coefficients from the outcome sub-model.

`par.outcome.aux` A named vector of (auxiliary) coefficients from the outcome sub-model (if `outcome.model` = "betabinomial").

`df` Degrees of freedom.

`se.sensitive` Standard errors for parameters in the sensitive-item sub-model.

`se.control` Standard errors for parameters in the control-items sub-model.

`se.misreport` Standard errors for parameters in the misreport sub-model.

`se.outcome` Standard errors for parameters in the outcome sub-model.

`se.outcome.aux` Standard errors for the auxiliary parameters in the outcome sub-model (if `outcome.model` = "betabinomial").

`vcov.mle` Variance-covariance matrix.

`w` The matrix of posterior predicted probabilities for each observation in the data used for model fitting.

`data` The data frame used for model fitting.

`direct` The string indicating the variable name of the direct question.

`treatment` The string indicating the variable name of the treatment indicator.

`model.misreport` A logical value indicating whether a misreport sub-model was fit.

`outcome.model` The type of model used as the outcome sub-model.

`outcome.constrained` A logical value indicating whether the parameter U* was constrained to 0 in the outcome sub-model.

`control.constraint` A string indicating the constraints placed on the parameters Z* and U* in the control-items sub-model.

`misreport.treatment` A logical value indicating whether a treatment indicator was included in the misreport sub-model.

`weights` A string indicating the variable name of the survey weights.

`formula` The model formula.

`formula.control` The model specification of the control-items sub-model.

`formula.sensitive` The model specification of the sensitive-item sub-model.

`formula.misreport` The model specification of the misreport sub-model.

`formula.outcome` The model specification of the outcome sub-model.

`sensitive.response` The value 0 or 1 indicating the response to the list experiment/direct question that is considered sensitive.

`xlevels` The factor levels of the variables used in the model.

`llik` The model log-likelihood.

`n` The sample size of the data used for model fitting (this value excludes rows removed through listwise deletion).

`J` The number of control items in the list experiment.

`se` A logical value indicating whether standard errors were calculated.

`runs` The parameter estimates from each run of the EM algorithm (note: the parameters that result in the highest log-likelihood are used as the model solution).

`call` The method call.

`boot` A logical value indicating whether non-parametric bootstrapping was used to calculate model parameters and standard errors.

References

Eady, Gregory. 2017 "The Statistical Analysis of Misreporting on Sensitive Survey Questions."

Imai, Kosuke. 2011. "Multivariate Regression Analysis for the Item Count Technique." *Journal of the American Statistical Association* 106 (494): 407-416.

Examples

```
## EXAMPLE 1: Simulated list experiment and direct question
n <- 10000
J <- 4

# Covariates
x <- cbind(intercept = rep(1, n), continuous1 = rnorm(n),
           continuous2 = rnorm(n), binary1 = rbinom(n, 1, 0.5))

treatment <- rbinom(n, 1, 0.5)

# Simulate Z*
param_sensitive <- c(0.25, -0.25, 0.5, 0.25)
prob_sensitive <- plogis(x %*% param_sensitive)
true_belief <- rbinom(n, 1, prob = prob_sensitive)

# Simulate whether respondent misreports (U*)
param_misreport <- c(-0.25, 0.25, -0.5, 0.5)
prob_misreport <- plogis(x %*% param_misreport) * true_belief
misreport <- rbinom(n, 1, prob = prob_misreport)

# Simulate control items Y*
param_control <- c(0.25, 0.25, -0.25, 0.25, U = -0.5, Z = 0.25)
prob.control <- plogis(cbind(x, misreport, true_belief) %*% param_control)
control_items <- rbinom(n, J, prob.control)

# List experiment and direct question responses
direct <- true_belief
direct[misreport == 1] <- 0
y <- control_items + true_belief * treatment

A <- data.frame(y, direct, treatment,
                continuous1 = x[, "continuous1"],
                continuous2 = x[, "continuous2"],
                binary1 = x[, "binary1"])

## Not run:
model.sim <- listExperiment(y ~ continuous1 + continuous2 + binary1,
                            data = A, treatment = "treatment", direct = "direct",
                            J = 4, control.constraint = "none",
                            sensitive.response = 1)
summary(model.sim, digits = 3)

## End(Not run)


## EXAMPLE 2: Data from Eady (2017)
data(gender)

## Not run:
# Note: substantial computation time
```

```
model.gender <- listExperiment(y ~ gender + ageGroup + education +
                                    motherTongue + region + selfPlacement,
                                data = gender, J = 4,
                                treatment = "treatment", direct = "direct",
                                control.constraint = "none",
                                sensitive.response = 0,
                                misreport.treatment = TRUE)
summary(model.gender)

## End(Not run)
```

predict.listExperiment
Predict method for the list experiment

Description

Obtains predictions from a fitted list experiment model of the class `listExperiment`.

Usage

```
## S3 method for class 'listExperiment'
predict(object, newdata = NULL,
  treatment.misreport = 0, par.control = NULL, par.sensitive = NULL,
  par.misreport = NULL, ...)
```

Arguments

`object`	Object of class "listExperiment"
`newdata`	An optional data frame from which to calculate predictions.
`treatment.misreport`	Value of the treatment variable covariate in the misreport sub-model (if included in the model). **0** treatment indicator in the misreport sub-model is set to 0 for all individuals (default). **1** treatment indicator in the misreport sub-model is set to 1 for all individuals. **"observed"** treatment indicator in the misreport sub-model is set to the observed treatment value.
`par.control`	An optional set of control-items sub-model parameters to use in place of those from the fitted model.
`par.sensitive`	An optional set of sensitive-item sub-model parameters to use in place of those from the fitted model.
`par.misreport`	An optional set of misreport sub-model parameters to use in place of those from the fitted model.
`...`	Additional arguments

Details

If newdata is omitted, predictions will be made with the data used for model fitting.

Slots

z.hat	Predicted probability of answering affirmatively to the sensitive item in the list experiment.
u.hat	Predicted probability of misreporting (assuming respondent holds the sensitive belief).

References

Eady, Gregory. 2017 "The Statistical Analysis of Misreporting on Sensitive Survey Questions."

Examples

```

data(gender)

## Not run:
# Note: substantial computation time
model.gender <- listExperiment(y ~ gender + ageGroup + education +
                                   motherTongue + region + selfPlacement,
                               data = gender, J = 4,
                               treatment = "treatment", direct = "direct",
                               control.constraint = "none",
                               sensitive.response = 0,
                               misreport.treatment = TRUE)
predict(model.gender, treatment.misreport = 0)

## End(Not run)
```

---

print.listExperiment *Print object summary of listExperiment class*

Description

Calls summary.listExperiment.

Usage

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

Arguments

x	Object of class "listExperiment".
...	Additional arguments.

Details

Prints the object summary of the listExperiment class by calling the `summary.listExperiment` function.

`summary.listExperiment` *Object summary of the listExperiment class*

Description

Summarizes results from a list experiment regression fit using `listExperiment` or `bootListExperiment`.

Usage

```
## S3 method for class 'listExperiment'
summary(object, digits = 4, ...)
```

Arguments

`object`	Object of class "listExperiment".
`digits`	Number of significant digits to print.
`...`	Additional arguments.

Details

`summary.listExperiment` summarizes the information contained in a listExperiment object for each list experiment regression sub-model.

References

Eady, Gregory. 2017 "The Statistical Analysis of Misreporting on Sensitive Survey Questions."

Examples

```
data(gender)

## Not run:
# Note: substantial computation time
model.gender <- listExperiment(y ~ gender + ageGroup + education +
                                   motherTongue + region + selfPlacement,
                               data = gender, J = 4,
                               treatment = "treatment", direct = "direct",
                               control.constraint = "none",
                               sensitive.response = 0,
                               misreport.treatment = TRUE)
summary(model.gender)

## End(Not run)
```

Index