Package 'EffectLiteR'

January 3, 2019

Version 0.4-3

Date 2019-01-03

Title Average and Conditional Effects

Author Axel Mayer [aut, cre],
Lisa Dietzfelbinger [ctb]

Maintainer Axel Mayer <amayer2010@gmail.com>

Description Use structural equation modeling to estimate average and conditional effects of a treatment variable on an outcome variable, taking into account multiple continuous and categorical covariates.

Depends R (>= 3.1.0), lavaan (>= 0.5-20)

Imports methods, shiny (>= 0.11), foreign, ggplot2, nnet, survey, lavaan.survey, car

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/amayer2010/EffectLiteR

LazyData yes

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-01-03 10:30:03 UTC

R topics documented:

`autoSelectSubset` *Autoselect Subset for Aggregated Effects*

Description

Automatically selects a subset of the original dataset for computing specific aggregated effects. The subset is selected such that it is as close as possible to the user supplied newdata frame. The function uses exact matching for categorical covariates (and the treatment if specified) and matching based on the Mahalanobis distance for continuous covariates.

Usage

```
autoSelectSubset(obj, newdata, nsub = 10)
```

Arguments

`obj`	Object of class `effectlite`.
`newdata`	A data.frame with a single row, containing the same continuous and categorical covariates (and potentially the treatment variable) as used when fitting the EffectLiteR model in obj.
`nsub`	Integer. How many data points should be used for matching the continous covariates. Will be ignored if no values for continuous covariates are specified.

Value

Vector of integers indicating the rows to use for computing the aggregated effects. Can directly be used in `computeAggregatedEffects`

Examples

```
m1 <- effectLite(y="dv", z=c("z1"), k=c("k1"), x="x",
control="control", data=example01, fixed.cell=TRUE, fixed.z=TRUE)
newdata <- data.frame(k1=NA, z1=1)
agg.subset <- autoSelectSubset(m1, newdata)
```

computeAggregatedEffects

Compute Aggregated Effects

Description

Computes aggregates of conditional effects for a subset of the original dataset based on a fitted EffectLiteR model.

Usage

```
computeAggregatedEffects(obj, agg.subset)
```

Arguments

obj	Object of class effectlite.
agg.subset	Vector of integers indicating the row numbers of the original dataset for the subset used to compute the aggregated effect

Value

Object of class "data.frame".

Examples

```
m1 <- effectLite(y="dv", z=c("z1"), k=c("k1"), x="x",
control="control", data=example01, fixed.cell=TRUE, fixed.z=TRUE)
newdata <- data.frame(k1=NA, z1=1)
agg.subset <- autoSelectSubset(m1, newdata)
computeAggregatedEffects(m1, agg.subset)
```

conditionalEffectsPlot

Plot conditional effects

Description

Can be used to make a conditional effects plot with an effect function on the y axis and a covariate on the x axis. ggplot2 is used to create the plot.

Usage

```
conditionalEffectsPlot(obj, zsel = "id", gxsel = "g1", colour = "",
  show.ci = FALSE, regression = "default", regression.ci = FALSE)
```

Arguments

obj	Object of class effectlite obtained from fitting an effect model using effectLite
zsel	Name of a covariate (character string) plotted on the x-axis. If "id" (the default) the subject index is shown on the x-axis, where subjects in the data are enumerated as 1:nrow(data).
gxsel	Name of an effect function (character string) plotted on the y-axis.
colour	Name of a covariate (character string) used as colour variable in the plot.
show.ci	Logical. Should 95 percent confidence intervals around conditional effects be shown in the plot.
regression	Specifies if a regression line should be drawn. Can be one of c("default","smooth","linear","none")
regression.ci	Logical. Will be passed on to geom_smooth and specifies its se argument. Notice that the confidence interval shown by geom_smooth does not take uncertainty into account that comes from estimating the values of the conditional effects on the y axis.

Value

Object of class c("gg", "ggplot").

Examples

```
m1 <- effectLite(y="dv", x="x", k="k1", z="z1", control="control", data=example01)
conditionalEffectsPlot(m1, zsel="z1", gxsel="g1", colour="k1")
```

---

effectLite *Estimate average and conditional effects*

Description

This function is the main funtion of the package and can be used to estimate average and conditional effects of a treatment variable on an outcome variable, taking into account any number of continuous and categorical covariates. It automatically generates lavaan syntax for a multi-group structural equation model, runs the model using lavaan, and extracts various average and conditional effects of interest.

Usage

```
effectLite(y, x, k = NULL, z = NULL, data, method = "sem",
  control = "default", measurement = character(),
  fixed.cell = "default", fixed.z = "default", missing = "default",
  se = "standard", syntax.only = FALSE, interactions = "all",
  homoscedasticity = "default", test.stat = "default",
  propscore = NULL, ids = ~0, weights = NULL, add = character(),
  ...)
```

Arguments

y	Dependent variable (character string). Can be the name of a manifest variable or of a latent variable.
x	Treatment variable (character string) treated as categorical variable.
k	Vector of manifest variables treated as categorical covariates (character vector).
z	Vector of continuous covariates (character vector). Names of both manifest and latent variables are allowed.
data	A data frame.
method	Can be one of `c("sem","lm")` and indicates which function is used to fit the model.
control	Value of `x` that is used as control group. If "default", takes the first entry of `as.factor(x)`.
measurement	Measurement model. The measurement model is lavaan syntax (character string), that will be appended before the automatically generated lavaan input. It can be used to specify a measurement for a latent outcome variable and/or latent covariates. See also the example and `generateMeasurementModel`.
fixed.cell	logical. If `FALSE` (default), the group sizes are treated as stochastic rather than fixed.
fixed.z	logical. If `FALSE` (default), the continuous covariates are treated as stochastic rather than fixed. fixed.z
missing	Missing data handling. Will be passed on to `sem` or ignored for `method="lm"`.
se	Type of standard errors. Will be passed on to `sem` or ignored for `method="lm"`.
syntax.only	logical. If `TRUE`, only syntax is returned and the model will not be estimated.
interactions	character. Can be one of `c("all","2-way","X:K,X:Z","X:K","X:Z","none","no")` and indicates the type of interaction used in the parameterization of the regression.
homoscedasticity	logical. If `TRUE`, residual variances of the dependent variable are assumed to be homogeneous across cells.
test.stat	character. Can be one of `c("default","Chisq","Ftest")` and indicates the statistic used for the hypothesis tests. The tests are either based on the large sample Chi-Squared statistic (Wald tests) or the finite sample F statistic with approximate F distribution. The default setting for `method="sem"` is `"Chisq"` and the default setting for `method="lm"` is `"Ftest"`.
propscore	Vector of covariates (character vector) that will be used to compute (multiple) propensity scores based on a multinomial regression without interactions. Alternatively, the user can specify a formula with the treatment variable as dependent variable for more control over the propensity score model.
ids	Formula specifying cluster ID variables. Will be passed on to `lavaan.survey`. See `svydesign` for details.
weights	Formula to specify sampling weights. Currently only one weight variable is supported. Will be passed on to `lavaan.survey`. See `svydesign` for details. Note: Only use weights if you know what you are doing. For example, some conditional treatment effects may require different weights than average effects.

add	Character string that will be pasted at the end of the generated lavaan syntax. Can for example be used to add additional (in-) equality constraints or to compute user-defined conditional effects.
...	Further arguments passed to sem.

Value

Object of class effectlite.

Examples

```
## Example with one categorical covariate
m1 <- effectLite(y="y", x="x", k="z", control="0", data=nonortho)
print(m1)

## Example with one categorical and one continuous covariate
m1 <- effectLite(y="dv", x="x", k=c("k1"), z=c("z1"), control="control", data=example01)
print(m1)

## Example with latent outcome and latent covariate
measurement <- '
eta2 =~ 1*CPM12 + 1*CPM22
eta1 =~ 1*CPM11 + 1*CPM21
CPM11 + CPM12 ~ 0*1
CPM21 ~ c(m,m)*1
CPM22 ~ c(p,p)*1'

m1 <- effectLite(y="eta2", x="x", z=c("eta1"), control="0",
                 measurement=measurement, data=example02lv)
print(m1)

## Not run:
## Example with cluster variable and sampling weights
m1 <- effectLite(y="y", x="x", z="z", fixed.cell=TRUE, control="0",
                    syntax.only=F, data=example_multilevel,
                    ids=~cid, weights=~weights)
print(m1)

## End(Not run)
```

`effectLiteGUI` *Shiny interface for effectLite*

Description

This function calls a shiny interface for effectLite.

Usage

```
effectLiteGUI(launch.browser = TRUE)
```

Arguments

launch.browser Option will be passed on to runApp

EffectLiteR *EffectLiteR*

Description

EffectLiteR

---

elrPredict *Predict Conditional Effects*

Description

Predicts conditional treatment effects based on a fitted EffectLiteR model.

Usage

```
elrPredict(obj, newdata = NULL, add.columns = "expected-outcomes")
```

Arguments

obj	Object of class effectlite.
newdata	An optional data.frame, containing the same continuous and categorical covariates as used when fitting the EffectLiteR model in obj. Only covariates (and neither the dependent variable nor indicators for latent variables) should be included.
add.columns	Used to request additional columns. Can be one or several of c("covariates", "modmat", "expected-outcomes", "prop-covariates").

Value

Object of class "data.frame".

Examples

```
m1 <- effectLite(y="dv", z=c("z1"), k=c("k1","kateg2"), x="x",
control="control", data=example01)
newdata <- data.frame(k1="male", kateg2="1", z1=2)
elrPredict(m1, newdata)
```

elrReadData *Read Data File*

Description

Tries to determine the format of the data by the file ending and chooses the appropriate function to read data. Currently supports .csv, .dat, .txt, .sav, and .xpt and calls `read.csv`, `read.csv2`, `read.table`, `read.spss`, `read.xport` accordingly. The default values for arguments depend on the function used to read data.

Usage

```
elrReadData(file, name = NULL, header = "default", sep = "default",
  dec = "default", use.value.labels = "default", na.strings = "NA")
```

Arguments

`file`	Name of the file to read.
`name`	Pure file name (without path to file) to read. If `file` includes a lengthy path name with many special characters, specifying this argument in addition to `file` may help the function to find the file ending.
`header`	See `read.table`.
`sep`	See `read.table`.
`dec`	See `read.table`.
`use.value.labels`	See `read.spss`.
`na.strings`	See `read.spss`.

Value

Object of class `"data.frame"`.

example01 *Dataset example01.*

Description

A simulated dataset. The variables are:

Format

A data frame with 2000 rows and 7 variables.

Details

- x. Treatment variable with values control, treat1, and treat2.
- k1. Categorical covariate with values male and female.
- kateg2. Categorical covariate with values 1 and 2.
- z1-z3. Continuous covariates.
- dv. Coninuous dependent variable.

---

example02lv *Dataset example02lv.*

Description

A simulated dataset with latent variables. The variables are:

Format

A data frame with 300 rows and 6 variables.

Details

- CPM11. First indicator of latent covariate.
- CPM21. Second indicator of latent covariate.
- CPM12. First indicator of latent outcome.
- CPM22. Second indicator of latent outcome.
- x. Dichotomous treatment variable with values 0 (control), and 1 (treatment).
- k. Categorical covariate with values first, second, and third.

---

example_multilevel *Dataset example_multilevel.*

Description

A simulated dataset with a cluster ID and sampling weights to test multilevel options. The variables are:

Format

A data frame with 800 rows and 7 variables.

Details

- y. Coninuous dependent variable.
- x. Treatment variable with values 0, 1.
- z. Continuous covariate.
- xz. Product of x and z.
- cid. Cluster ID.
- weights. Sampling weights.
- iptw. Classic inverse probability of treatment weights based on a logistic regression of x on z. Use with care (only for average effects).

`generateMeasurementModel` *Generate measurement model*

Description

This function automatically generates `lavaan` syntax for the measurement model for a call to `effectLite`. It is currently also used in the shiny interface.

Usage

```
generateMeasurementModel(names = NULL, indicators, ncells,
  model = NULL)
```

Arguments

`names`	A vector of character strings with names of latent variables. If not specified, names(indicators) is used.
`indicators`	A list of vectors of character strings to specify indicators of latent variables (see example).
`ncells`	Number of groups/cells.
`model`	A vector of character strings of the same length as names. It is used to specify the type of measurement model for each of the latent variables. Each element can be one of `c("default","parallel","tau-equi","tau-cong")` indicating whether a parallel, essentially tau-equivalent, or tau-congeneric measurement model is used. If "default", the function tries to guess a reasonable measurement model: Congeneric for latent variables with three or more indicators, essentially tau-equivalent for latent variables with less than three indicators and for latent variables with cross-loadings (e.g., method factors), and parallel for single-indicator latent variables. If NULL, "default" is assumed for all latent variables.

Examples

```
## Example with three latent variables
names <- c("eta", "xi1", "xi2")
indicators <- list("eta" = c("y1","y2","y3"),
                   "xi1" = c("z1","z2"),
                   "xi2" = c("z12","z22","z32","z42"))
ncells = 6
model = c("parallel","tau-equi","tau-cong")
cat(generateMeasurementModel(names, indicators, ncells, model))

## Example with method factor
names <- c("eta", "xi", "mf")
indicators <- list("eta" = c("y12","y22"),
                   "xi" = c("y11","y21"),
                   "mf" = c("y12","y22"))
ncells = 2
cat(generateMeasurementModel(names, indicators, ncells))
```

MDRS2016 *Dataset MDRS2016.*

Description

The simulated dataset with latent variables used in Mayer, Dietzfelbinger, Rosseel, and Steyer (2016). The variables are:

Format

A data frame with 1000 rows and 10 variables.

Details

- y11. First indicator of latent covariate (pretest mental health).
- y21. Second indicator of latent covariate (pretest mental health).
- y31. Third indicator of latent covariate (pretest mental health).
- y12. First indicator of latent outcome (posttest mental health).
- y22. Second indicator of latent outcome (posttest mental health).
- y32. Third indicator of latent outcome (posttest mental health).
- x. Categorical treatment variable with values 0 (wait list control group), 1 (conventional therapy), and 2 (innovative therapy).
- k. Categorical covariate with values 0 (male) and 1 (female).
- Ix1. Binary indicator for conventional therapy (X=1).
- Ix2. Binary indicator for innovative therapy (X=2).

nonortho *Dataset nonortho.*

Description

A simulated dataset. The variables are:

Format

A data frame with 500 rows and 3 variables

Details

- y. Continuous dependent variable depression.
- x. Treatment variable with values 0 (control), 1 (treat1), and 2 (treat2).
- z. Categorical covariate with values 0 (low neediness), 1 (medium neediness) and 2 (high neediness).

Index