Package 'eive'

August 21, 2023

Type Package

Title An Algorithm for Reducing Errors-in-Variable Bias in Simple and Multiple Linear Regressions

Version 3.1.3

Date 2023-08-20

Author Mehmet Hakan Satman (Ph.D.), Erkin Diyarbakirlioglu (Ph.D.)

Maintainer Mehmet Hakan Satman <mhsatman@istanbul.edu.tr>

Description Performs a compact genetic algorithm search to reduce errors-in-variables bias in linear regression. The algorithm estimates the regression parameters with lower biases and higher variances but mean-square errors (MSEs) are reduced.

License GPL

Imports Rcpp (>= 0.11.1), compiler(>= 2.0.0)

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-08-21 09:12:45 UTC

Suggests testthat

RoxygenNote 7.2.3

Encoding UTF-8

R topics documented:

eive-package *EIVE - Errors-in-Variable estimation*

Description

This package includes functions for compact genetic algorithms and errors-in-variable estimation. The function 'eive' performs a genetic search to reduce the errors-in-variable bias in ordinary least squares estimator.

Change log:

In version 3.1.2

- Add eive.cga.formula for lm() compatible regression settings.

- Implement roxygen type documentation system

In version 3.1.1,

- Enhance document for multivariate eive (eivem)

In version 3.1,

- Errors-in-variables with multiple response variables.

- eive.cga() now returns 2 new items: $cleanx and $measurementerror which are also accessable using the keys $proxy$fitted.values and $proxy$residuals, respectively.

- eivem() added for multiple y values

- new tests added and can be triggered using devtools:::test()

- updated docs

In version 2.1, more speed improvements by using lm.fit instead lm in critical code.

In version 2.0, some routines are rewritten in C++ and wrapped using Rcpp so a substantial speed improvement achieved.

Author(s)

Mehmet Hakan Satman <mhsatman@istanbul.edu.tr> Erkin Diyarbakirlioglu <ediyarbakirlioglu@gmail.com>

Maintainer: Mehmet Hakan Satman <mhsatman@istanbul.edu.tr>

cga *Compact Genetic Algorithm*

Description

Performs a Compact Genetic Algorithm (CGA) search for a given chromosome size, population size (mutation rate), and an objective function.

Usage

```
cga(chsize, popsize, evalFunc)
```

Arguments

chsize	Number of bits.
popsize	Size of population. The value is used for mutating the probability vector by 1/popsize.
evalFunc	Objective function.

Value

Binary vector of size chsize.

---

cga_generate_chromosome

Generate Chromosome

Description

Generate a binary vector using a probability vector This function is not directly called by user. CGAs (Compact genetic algorithms) sample chromosomes using this probability vector. A probability vector contains[P1, P2, ..., PN] and the function generates and returns a chromosome[B1, B2, ..., BN]. The probability of BK having the value of 1 is PK. So, it has more chance to have [1, 1, 1, 0, 0] rather than [0, 0, 0, 1, 1] when the probability vector is [0.9, 0.9, 0.9, 0.1, 0.1].

Usage

```
cga_generate_chromosome(prob_vec, vect)
```

Arguments

prob_vec	Vector of probabilities
vect	Vector of bits.

Value

Mutates the vect. Returns null.

`eive.cga` *Performs CGA based errors-in-variables correction for a given set of variables. A single independent variable is supposed to be measured subject to error.*

Description

Performs CGA based errors-in-variables correction for a given set of variables. A single independent variable is supposed to be measured subject to error.

Usage

```
eive.cga(dirtyx, otherx = NULL, y, numdummies = 10, popsize = 20)
```

Arguments

`dirtyx`	Vector of independent variable that is measured with error.
`otherx`	Matrix of other independent variables. If the model has a single independent variable, it is NULL by default.
`y`	Vector of response variable
`numdummies`	Number of dummy variables used in auxiliary regression.
`popsize`	Population size parameter for compact genetic algorithm. 1/popsize is the mutation rate.

Value

A list() of regression equations.

Slots

`ols` lm object calculated using original values

`eive` lm object calculated using the predicted variable by eive

`proxy` lm object of proxy regression obtained by genetic search.

`cleanedx` Error-free estimate of the x variable (dirtyx) that is measured with error.

`measurementerror` Estimate of the measurement error.

Examples

```
# Creating an artificial data
# Loading required package
require("eive")

# Setting random number generator seed to 12345
# so each time the script runs, same numbers will
# be generated
set.seed(12345)
```

```
# Number of observations is set to 30
n <- 30

# Unobserved X values are drawn from a Normal distribution
# with mean 10 and variance 7
clean.x <- rnorm(n, mean = 10, sd = sqrt(7))

# Measurement error values are dranw from a Normal distribution
# with mean 0 and variance 3
delta.x <- rnorm(n, mean = 0, sd = sqrt(3))

# Error term of regression. Normally distributed with mean 0 and
# variance 5
e <- rnorm(n, mean = 0, sd = sqrt(5))

# Generating Y values using the linear model
# In this model, intercept is 20 and slope is 10.
y <- 20 + 10 * clean.x + e

# Generating observed X values by adding measurement errors
# to unobserved X
dirty.x <- clean.x + delta.x

# Performs a genetic search to find dummy variables that
# used in two stage least squares.
# Please un-comment the line below
# result <- eive.cga (dirtyx=dirty.x, y=y, numdummies=10)

# Print the result
# Please un-comment the line below
# print(result)

############################################ OUTPUT ############################################
# $ols
#
# Call:
# lm(formula = y ~ dirtyx)
#
# Coefficients:
# (Intercept)       dirtyx
#     63.590        5.533
#
#
# $eive
#
# Call:
# lm(formula = y ~ ols.proxy$fitted.values)
#
# Coefficients:
#            (Intercept)  ols.proxy$fitted.values
#                 23.863                    9.229
#
```

```
#
# $proxy
#
# Call:
# lm(formula = dirtyx ~ matrix(best, nrow = n))
#
# Coefficients:
#              (Intercept)   matrix(best, nrow = n)1   matrix(best, nrow = n)2
#                  12.9321                   -0.6252                   -1.9923
# matrix(best, nrow = n)3   matrix(best, nrow = n)4   matrix(best, nrow = n)5
#                   0.7537                   -0.7076                   -0.5247
# matrix(best, nrow = n)6   matrix(best, nrow = n)7   matrix(best, nrow = n)8
#                  -0.9196                   -2.0802                   -0.9246
# matrix(best, nrow = n)9  matrix(best, nrow = n)10
#                  -0.6164                    1.9694
######################################### END OF OUTPUT ##########################################
```

`eive.cga.formula` *Performs CGA based errors-in-variables correction for given formula and data. A single independent variable is supposed to be measured subject to error.*

Description

Performs CGA based errors-in-variables correction for given formula and data. A single independent variable is supposed to be measured subject to error.

Usage

```
eive.cga.formula(formula, data, dirtyx.varname, numdummies = 10, popsize = 20)
```

Arguments

`formula`	Formula object.
`data`	data.frame that holds the regression data.
`dirtyx.varname`	String key value of the erroneous independent variable.
`numdummies`	Number of dummy variables used in auxiliary regression.
`popsize`	Population size parameter for compact genetic algorithm. 1/popsize is the mutation rate.

Value

A list() of regression equations.

Slots

`ols` lm object calculated using original values

`eive` lm object calculated using the predicted variable by eive

`proxy` lm object of proxy regression obtained by genetic search.

`cleanedx` Error-free estimate of the x variable (dirtyx) that is measured with error.

`measurementerror` Estimate of the measurement error.

See Also

eive.cga

Examples

```
set.seed(12345)
n <- 30
clean_x <- rnorm(n, mean = 10, sd = sqrt(7))
delta_x <- rnorm(n, mean = 0, sd = sqrt(3))

e <- rnorm(n, mean = 0, sd = sqrt(5))
y <- 20 + 10 * clean_x + e

dirty_x <- clean_x + delta_x

mydata <- data.frame(y = y, dirtyx = dirty_x)

result <- eive.cga.formula(
     formula = y ~ dirtyx,
     dirtyx.varname = "dirtyx",
     data = mydata,
     numdummies = 10
)
```

`eivem` *Performs CGA based errors-in-variables correction for a given set of variables in case of multiple Y variables are provided.*

Description

A single independent variable is supposed to be measured subject to error. This functions is the multivariate version of the classical algorithm. Additional response variables are used to get better estimates.

Usage

```
eivem(dirtyx, otherx = NULL, y, numdummies = 10, popsize = 20)
```

Arguments

dirtyx	Vector of independent variable that is measured with error.
otherx	Matrix of other independent variables. If the model has a single independent variable, it is NULL by default.
y	Matrix of response variables. Y_i is placed in the ith row of the matrix.
numdummies	Number of dummy variables used in auxiliary regression. Default is 10.
popsize	Population size parameter for compact genetic algorithm. Default is 20. 1/popsize is the mutation rate.

Value

A list() of regression equations.

Slots

`ols` List of lm objects calculated using original values

`eive` List of lm objects calculated using the predicted variable by eive

`proxy` lm object of proxy regression obtained by genetic search.

`cleanedx` Error-free estimate of the x variable (dirtyx) that is measured with error.

`measurementerror` Estimate of the measurement error.

Examples

```
# Creating an artificial data

# Loading required package
require("eive")

# Setting random number generator seed to 12345
# so each time the script runs, same numbers will
# be generated
set.seed(12345)

# Number of observations is set to 30
n <- 30

# Unobserved X values are drawn from a Normal distribution
# with mean 10 and variance 7
clean_x1 <- rnorm(n, mean = 10, sd = sqrt(7))
clean_x2 <- rnorm(n, mean = 10, sd = sqrt(7))

# Measurement error values are dranw from a Normal distribution
# with mean 0 and variance 3
delta_x1 <- rnorm(n, mean = 0, sd = sqrt(3))

# Error term of regression. Normally distributed with mean 0 and
# variance 5
e1 <- rnorm(n, mean = 0, sd = sqrt(5))
```

```
e2 <- rnorm(n, mean = 0, sd = sqrt(5))

# Generating Y values using the linear model
# In this model, intercept is 20 and slope is 10.
y1 <- 20 + 10 * clean_x1 + 10 * clean_x2 + e1
y2 <- 10 + 5 * clean_x1 + 5 * clean_x2 + e2

# Generating observed X values by adding measurement errors
# to unobserved X
dirty_x1 <- clean_x1 + delta_x1

# Performs a genetic search to find dummy variables that
# used in two stage least squares.
# Please un-comment the line below
result <- eivem(dirtyx = dirty_x1, otherx = clean_x2, y = cbind(y1, y2), numdummies = 10)

# Print the result
# Please un-comment the line below
# print(result)

########################################### OUTPUT ############################################
#> result
# $ols
# $ols[[1]]
#
# Call:
# lm(formula = y[, reg.index] ~ dirtyx + otherx)
#
# Coefficients:
# (Intercept)       dirtyx       otherx
#      54.141        6.067       10.137
#
#
# $ols[[2]]
#
# Call:
# lm(formula = y[, reg.index] ~ dirtyx + otherx)
#
# Coefficients:
# (Intercept)       dirtyx       otherx
#      24.814        3.205        5.089
#
#
#
# $eive
# $eive[[1]]
#
# Call:
# lm(formula = y[, reg.index] ~ ols_proxy$fitted.values + otherx)
#
# Coefficients:
#             (Intercept)  ols_proxy$fitted.values                   otherx
#                  24.737                    9.727                    9.147
```

```
#
#
# $eive[[2]]
#
# Call:
# lm(formula = y[, reg.index] ~ ols_proxy$fitted.values + otherx)
#
# Coefficients:
#             (Intercept)  ols_proxy$fitted.values                   otherx
#                   8.313                    5.240                    4.552
#
#

# $proxy
#
# Call:
# lm(formula = dirtyx ~ matrix(best, nrow = n))
#
# Coefficients:
#              (Intercept)   matrix(best, nrow = n)1   matrix(best, nrow = n)2
#                 6.314397                 -0.211580                  1.729143
#  matrix(best, nrow = n)3   matrix(best, nrow = n)4   matrix(best, nrow = n)5
#                 1.994915                  0.947531                 -0.363107
#  matrix(best, nrow = n)6   matrix(best, nrow = n)7   matrix(best, nrow = n)8
#                 0.001768                  1.742553                 -0.023750
#  matrix(best, nrow = n)9  matrix(best, nrow = n)10
#                 0.134750                  2.324853
#
#
# $cleanedx
#         1         2         3         4         5         6         7         8
# 12.730307 12.130102 11.065586  9.795474 12.697138  6.450915 12.673388 10.516553
#         9        10        11        12        13        14        15        16
# 11.095771  7.981887 11.694464 14.841812 11.098755 12.290371  8.988344 12.704789
#        17        18        19        20        21        22        23        24
#  7.671861  9.477178 13.458999 10.964004 11.465852 14.591473  9.771724  6.239335
#        25        26        27        28        29        30
#  6.425397 15.031410  8.992839 12.808138 13.435249  9.799758
#
# $measurementerror
#          1          2          3          4          5          6          7
# -0.9220426 -2.5783644 -0.3964263  1.7585818 -2.1106159 -4.4345451  0.5319987
#          8          9         10         11         12         13         14
#  1.5127360 -0.9523682 -2.6583539 -1.9074299 -1.3927085 -1.9356982  3.1225578
#         15         16         17         18         19         20         21
#  1.4554922  1.0891572  1.4141792 -1.7600789  0.3310142  1.5952156  1.7146703
#         22         23         24         25         26         27         28
#  1.0669497 -2.0036393  3.9419318  1.0296643  2.9783401  0.8968531 -1.7001587
#         29         30
# -0.8864360  1.1995241
#
########################################## END OF OUTPUT ##########################################
```

`generate.eive.data` *Generates simulated errors-in-variables regression data*

Description

Generates simulated errors-in-variables regression data

Usage

```
generate.eive.data(n, e.sd, delta.sd, seed = 12345, useotherx = FALSE)
```

Arguments

`n`	Number of observations
`e.sd`	Standard deviation of error term of regression
`delta.sd`	Standard deviation of error in exploratory variable
`seed`	Seed for random number generator. 12345 by default
`useotherx`	Logical. If TRUE, an additional independent variable is added.

Value

A matrix of variables.

Slots

`xdelta` Errorenous X variable

`otherx` Other X variable

`y` Response variable

Index