Package ‘galts’

February 19, 2015

Type   Package
Title  Genetic algorithms and C-steps based LTS (Least Trimmed Squares)
        estimation
Version 1.3
Date   2013-02-06
Author Mehmet Hakan Satman
Maintainer Mehmet Hakan Satman <mhsatman@istanbul.edu.tr>
Description This package includes the ga.lts function that estimates
        LTS (Least Trimmed Squares) parameters using genetic algorithms
        and C-steps. ga.lts() constructs a genetic algorithm to form a
        basic subset and iterates C-steps as defined in Rousseeuw and
        van-Driessen (2006) to calculate the cost value of the LTS
        criterion. OLS(Ordinary Least Squares) regression is known to
        be sensitive to outliers. A single outlying observation can
        change the values of estimated parameters. LTS is a resistant
        estimator even the number of outliers is up to half of the
        data. This package is for estimating the LTS parameters with
        lower bias and variance in a reasonable time. Version 1.3
        included the function medmad for fast outlier detection in
        linear regression.

Depends genalg, DEoptim
Repository CRAN
License GPL
LazyLoad yes
Date/Publication 2013-02-07 09:27:39
NeedsCompilation no

R topics documented:

galts-package ................................................................. 2
ga.lts ................................................................. 3
medmad ................................................................. 5
medmad.cov ............................................................... 6
Description

This package includes the ga.lts function that estimates LTS (Least Trimmed Squares) parameters using genetic algorithms and C-steps. ga.lts() constructs a genetic algorithm to form a basic subset and iterates C-steps as defined in Rousseeuw and van-Driessen (2006) to calculate the cost value of the LTS criterion. OLS (Ordinary Least Squares) regression is known to be sensitive to outliers. A single outlying observation can change the values of estimated parameters. LTS is a resistant estimator even the number of outliers is up to half of the data. This package is for estimating the LTS parameters with lower bias and variance in a reasonable time.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>gals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.1</td>
</tr>
<tr>
<td>Date:</td>
<td>2011-02-04</td>
</tr>
<tr>
<td>License:</td>
<td>GPL</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

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

References


Examples

```r
# Data generating process
x1<-rnorm(100)
x2<-rnorm(100)
e<-rnorm(100)
```
# Setting betas to 5
y<-5+5*x1+5*x2+e

# Contaminate the data on the dimension of X’s randomly
# This is the maximum contamination rate that the LTS can cope with.
outlyings<-sample(1:100, 40)
x1[outlyings]<-10
x2[outlyings]<-10

# Estimating LTS with ga (Default optimization method)
lts<-ga.lts(y~x1+x2, popsize=40, iters=2, lower=-20, upper=20)
print(lts)

# Estimating LTS with differential evolution
lts<-ga.lts(y~x1+x2, popsize=40, iters=2, lower=-20, upper=20, method="de")
print(lts)

---

**ga.lts**

*Function for estimating the LTS (Least Trimmed Squares) regression parameters using genetic algorithms.*

**Description**

This function estimates the LTS (Least Trimmed Squares) regression parameters using genetic algorithms. LTS is a robust regression estimator with high breakdown property. LTS is a resistant estimator even the number of outliers is up to half of the data. However, calculating LTS estimator is computationally expensive. `ga.lts()` uses evolutionary algorithms (genetic algorithms by default, optionally differential evolution) to construct a basic subset and iterates C-steps as defined in Rousseeuw and van-Driessen (2006). Despite lower time efficiency of the `ga.lts()`, estimations have lower mean square errors, as a result of lower biases and lower variances.

**Usage**

`ga.lts(formula, h = NULL, iters = 2, popsize = 50, lower, upper, csteps = 2, method = "ga", verbose = FALSE)`

**Arguments**

- **formula** Dependent ~ Independents style formula as same in lm() and glm().
- **h** User defined variable to define the majority of the data. Default is floor(n/2)+floor((p+1)/2) where n is the number of observations and p is the number of parameters to estimate.
- **iters** Number of generations of the evolutionary algorithm. This variable can be kept larger if the precision is more important than the computation time. Default is 2.
- **popsize** Number of candidates (chromosomes) in the population of evolutionary algorithm. Default is 50.
- **lower** Lower bound for the initial estimates of the parameters.
- **upper** Upper bound for the initial estimates of the parameters.
csteps  Number of C-steps to be performed for each candidate solution. Default is 2.
method  A string variable for the evolutionary algorithm. 'ga' for the genetic algorithms and 'de' for the differential evolution. Default is 'ga'
verbose  A boolean variable for printing the current status of algorithms to screen. Default is FALSE.

Value

coefficients  A vector of estimated parameters
crit  LTS criterion value for the reported coefficients
method  Name of the method used in the optimization process

Author(s)

Mehmet Hakan Satman

References


Examples

#Data generating process
x1<-rnorm(100)
x2<-rnorm(100)
e<-rnorm(100)
#Setting betas to 5
y<-5+5*x1+5*x2+e
#Contaminate the data on the dimension of X's randomly
#This is the maximum contamination rate that the LTS can cope with.
outlyings<-sample(1:100,48)
x1[outlyings]<-10
x2[outlyings]<-10
#Estimating LTS with ga (Default optimization method)
lts<-ga.lts(y~x1+x2, popsize=40, iters=2, lower=-20, upper=20)
print(lts)

#Estimating LTS with differential evolution
lts<-ga.lts(y~x1+x2, popsize=40, iters=2, lower=-20, upper=20, method="de")
print(lts)
medmad

*Function for detecting regression outliers*

**Description**

A method for detecting regression outliers.

**Usage**

```r
medmad(formula, h=NULL, csteps=20)
```

**Arguments**

- `formula`: Dependent ~ Independents style formula as same in `lm()` and `glm()`.
- `h`: User defined variable to define the majority of the data. Default is `floor(n/2)+floor((p+1)/2)` where `n` is the number of observations and `p` is the number of parameters to estimate.
- `csteps`: Number of C-steps to be performed for each candidate solution. Default is 2.

**Value**

- `coefficients`: A vector of estimated parameters.
- `crit`: LTS criterion value for the reported coefficients.
- `residuals`: Calculated residuals from the final estimate of model.

**Author(s)**

Mehmet Hakan Satman

**Examples**

```r
n <- 100;
x1 <- rnorm(n, 0, 10);
x2 <- rnorm(n, 0, 10);
x3 <- rnorm(n, 0, 10);
x4 <- rnorm(n, 0, 10);
e <- rnorm(n);
x <- cbind(1, x1, x2, x3, x4);
p <- 5;
betas <- rep(5, p);
c <- 0.20
h <- n - n*c
y <- 5 + 5*x1 + 5*x2 + 5*x3 + 5*x4 + e;
x1[(h+1):n] <- rnorm(n-h, 100, 10);
x2[(h+1):n] <- rnorm(n-h, 100, 10);
x3[(h+1):n] <- rnorm(n-h, 100, 10);
x4[(h+1):n] <- rnorm(n-h, 100, 10);

mm <- medmad(formula=y~x1+x2+x3+x4, csteps=10)
```
medmad.cov  
Function for robust covariance matrix estimation.

Description
Function for robust covariance matrix estimation.

Usage
medmad.cov(data)

Arguments
data  Row matrix of data

Value
varcov  Covariance matrix

Author(s)
Mehmet Hakan Satman

Examples
n<-100;
c <- 0.20
h <- n - n*c
x1<-rnorm(n,0,10);
x2<-rnorm(n,0,10);
x3<-rnorm(n,0,10);
x4<-rnorm(n,0,10);
x1[(h+1):n]<-rnorm(n-h,100,10);
x2[(h+1):n]<-rnorm(n-h,100,10);
x3[(h+1):n]<-rnorm(n-h,100,10);
x4[(h+1):n]<-rnorm(n-h,100,10);
mat <- medmad.cov( cbind (x1, x2, x3, x4 ) )
print(mat)
Index

*Topic package
  galts-package, 2

ga.lts, 3
galts (galts-package), 2
galts-package, 2

medmad, 5
medmad_cov, 6