Package ‘micEconAids’

March 16, 2017

Version 0.6-18
Date 2017-03-16
Title Demand Analysis with the Almost Ideal Demand System (AIDS)
Author Arne Henningsen
Maintainer Arne Henningsen <arne.henningsen@gmail.com>
Depends R (>= 2.14.0), lmtest, micEcon (>= 0.6-0)
Imports stats, systemfit (>= 1.1-12), miscTools (>= 0.6-0)
Description Functions and tools for analysing consumer demand with the Almost Ideal Demand System (AIDS) suggested by Deaton and Muellbauer (1980).
License GPL (>= 2)
URL http://www.micEcon.org
NeedsCompilation no
Repository CRAN
Date/Publication 2017-03-16 13:49:05 UTC

R topics documented:

aidsBestA0 ............................................................... 2
aidsCalc ............................................................... 3
aidsConcav ............................................................. 6
aidsConsist ............................................................. 7
aidsElas ................................................................. 9
aidsEst ................................................................. 13
aidsMono ............................................................... 16
aidsPx ................................................................. 18
aidsUtility ............................................................. 20
Blanciforti86 .......................................................... 22
coeff.aidsEst ........................................................ 23
df.residual.aidsEst .................................................. 24
Find 'best' Value for alpha 0 in the AIDS

Description

Search for the intercept of the translog price index ($\alpha_0$) that gives the best fit to the Almost Ideal Demand System (see Michalek and Keyzer, 1992)

Usage

```r
aidsBestA0( priceNames, shareNames, totExpName,
  a0min = -50, a0max = 50, stoprange = 3, stopiter = 10,
  verbose = FALSE, ... )
```

Arguments

- `priceNames`: a vector of strings containing the names of the prices.
- `shareNames`: a vector of strings containing the names of the expenditure shares.
- `totExpName`: a string containing the variable name of total expenditure.
- `a0min`: lower limit of the range for searching for $\alpha_0$.
- `a0max`: upper limit of the range for searching for $\alpha_0$.
- `stoprange`: stop searching when the search interval is smaller than or equal to `stoprange`.
- `stopiter`: maximal number of iterations.
- `verbose`: print each determinant of the residual covariance matrix immediately after its calculation.
- `...`: arguments passed to `aidsEst`.

Details

The demand system is estimated using the Iterative Linear Least Squares Estimator (ILLE) suggested by Blundell and Robin (1999). This iterative procedure is equivalent to the method proposed by Michalek and Keyzer (1992). However, the latter do not correct the coefficient covariance matrix.

The fit of the model is measured in terms of the likelihood value. Since the determinant of the residual covariance matrix is monotonically decreasing with the likelihood value, we search for the smallest determinant of the residual covariance matrix.

Since each call of `aidsEst` generally takes a long time, the search algorithm is constructed to minimize the calls of the function `aidsEst`. 
aidsCalc

Value

a list containing following objects:

alpha0 \( \alpha_0 \) that gives the best fit.

allValues all \( \alpha_0 \) values that have been tested and the determinants of the corresponding residual covariance matrices.

iter number of iterations.

Author(s)

Arne Henningsen

References


See Also

aidsEst

Examples

data( Blanciforti86 )
# Data on food consumption are available only for the first 32 years
Blanciforti86 <- Blanciforti86[ 1:32, ]

bestA0 <- aidsBestA0( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
   c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
   data = Blanciforti86, useMatrix = FALSE )
# may take some time (argument 'useMatrix = FALSE' decreases
# the computation time only if the model and data set are small)
print( bestA0$alpha0 )
plot( bestA0$allValues ) # this should be convex

aidsCalc Shares and Quantities of the Almost Ideal Demand System

Description

Given prices, total expenditure and coefficients this function calculates the demanded quantities and expenditure shares based on the Almost Ideal Demand System.
Usage

aidsCalc( priceNames, totExpName, coef, data, priceIndex = "TL",
       basePrices = NULL, baseShares = NULL )

## S3 method for class 'aidsEst'
predict( object, newdata = NULL,
          observedShares = FALSE, ... )

Arguments

priceNames   a vector of strings containing the names of the prices.
totExpName   a string containing the variable name of total expenditure.
coef         a list containing the coefficients alpha, beta, gamma, and (only for the translog
             price index) alpha0.
data         a data frame containing the data.
priceIndex   a character string specifying the price index (see aidsPx) or a numeric vector
             providing the log values of the price index.
basePrices   a vector specifying the base prices for the Paasche, Laspeyres, and Tornqvist
             price index.
baseShares   a vector specifying the base expenditure shares for the Laspeyres, simplified
             Laspeyres, and Tornqvist index.
object       an object of class aidsEst.
newdata      an optional data frame which should contain the variables for the prediction. If
             omitted, the data frame used for the estimation is used also for the prediction.
observedShares logical. Using observed shares? (see details).
...           currently not used.

Details

The predict method for objects of class aidsEst extracts all relevant elements from an object
returned by aidsEst and passes them as arguments to aidsCalc. The optional argument
observedShares determines whether fitted (default) or observed expenditure shares are used in
the price index of the LA-AIDS.

Value

aidsCalc and the predict method for objects of class aidsEst return a list with following elements:

shares       a data frame containing the calculated expenditure shares.
quantities   a data frame containing the calculated quantities.

Author(s)

Arne Henningsen
References


See Also

`aidsEst`, `aidsPx`

Examples

data( Blancforti86 )
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[1:32,]

priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )

## LA-AIDS
estResult <- aidsEst( priceNames, shareNames, "xFood",
data = Blancforti86, priceIndex = "S" )

# using observed shares in the Stone index
lnp <- aidsPx( "S", priceNames, Blancforti86, shareNames )
fitted <- aidsCalc( priceNames, "xFood", coef = coef( estResult ),
data = Blancforti86, priceIndex = lnp )
fitted$shares # equal to estResult$wFitted
fitted$quant # equal to estResult$qFitted
# now the same with the predict method
fitted2 <- predict( estResult, observedShares = TRUE )
all.equal( fitted, fitted2 )

# using fitted shares in the Stone index
fitted <- aidsCalc( priceNames, "xFood", coef = estResult$coef,
data = Blancforti86, priceIndex = "S" )
# now the same with the predict method
fitted2 <- predict( estResult )
all.equal( fitted, fitted2 )

## AIDS
estResult <- aidsEst( priceNames, shareNames, "xFood",
data = Blancforti86, method = "IL" )

fitted <- aidsCalc( priceNames, "xFood", coef = coef( estResult ),
data = Blancforti86 )
fitted$shares # equal to estResult$wFitted
fitted$quant # equal to estResult$qFitted
fitted2 <- predict( estResult )
all.equal( fitted, fitted2 )
Description

Check whether the expenditure function of the AIDS is concave in prices.

Usage

aidsConcav( priceNames, totExpName, coef, data, shareNames = NULL )

## S3 method for class 'aidsConcav'
print( x, header = TRUE, ... )

Arguments

priceNames a vector of strings containing the names of the prices.
totExpName a string containing the variable name of total expenditure.
coef a list containing the coefficients alpha, beta, gamma, and (only for the translog price index) alpha0.
data a data frame containing the data.
shareNames a vector of strings containing the names of the expenditure shares.
x an object of class aidsConcav.
header logical. Print a header?
... currently unused.

Details

If argument shareNames is specified, observed shares are used for the calculation of the 'C' matrices to check for concavity; if argument shareNames is NULL (e.g., not specified), fitted shares are used for the calculation of the 'C' matrices.

Please note that checking concavity of the expenditure function requires that the expenditure function of the model exists. Therefore, the concavity condition can be checked, only if the symmetry condition is fulfilled and the translog price index is used.

Value

aidsConcav returns a list of class aidsConcav that contains following elements:

concavity a logical vector indicating whether the concavity condition is fulfilled at each observation.
nValidObs number of (valid) observation at which concavity could be checked.
nConcavObs number of observation at which the concavity condition is fulfilled.
concavPercent percent of observations where the concavity condition is fulfilled.
cMatrices a list of the 'C' matrices for each observation to check for concavity (see Deaton and Muellbauer, 1980b, p.76 ).
Check Consistency of the AIDS

Description

Check whether the specification of the AIDS is consistent with microeconomic demand theory (i.e. utility maximisation).

Author(s)

Arne Henningsen

References


See Also

aidsEst, aidsElas

Examples

data( Blanciforti86 )
# Data on food consumption are available only for the first 32 years
Blanciforti86 <- Blanciforti86[ 1:32, ]

priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )

# estimate the AIDS
estResult <- aidsEst( priceNames, shareNames, "xFood",
                      data = Blanciforti86, method = "IL" )

# check concavity with fitted shares
aidsConcav( priceNames, "xFood", coef = estResult$coef,
            data = Blanciforti86 )

# check concavity with observed shares
aidsConcav( priceNames, "xFood", coef = estResult$coef,
            data = Blanciforti86, shareNames = shareNames )
Usage

aidsConsist( priceNames, totExpName, coef, data,
    priceIndex = "TL", basePrices = NULL, baseShares = NULL,
    shareNames = NULL )

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

## S3 method for class 'aidsEst'
checkConsist( object, observedShares = FALSE, ... )

Arguments

priceNames a vector of strings containing the names of the prices.
totExpName a string containing the variable name of total expenditure.
coef a list containing the coefficients alpha, beta, gamma, and (only for the translog price index) alpha0.
data a data frame containing the data.
priceIndex a character string specifying the price index (see aidsPx) or a numeric vector providing the log values of the price index (passed to aidsMono).
basePrices a vector specifying the base prices for the Paasche, Laspeyres, and Tornqvist price index (passed to aidsMono).
baseShares a vector specifying the base expenditure shares for the Laspeyres, simplified Laspeyres, and Tornqvist index (passed to aidsMono).
shareNames a vector of strings containing the names of the expenditure shares (passed to aidsConcav).
x an object of class aidsConsist.
object an object of class aidsEst.
observedShares logical. Using observed shares? (see details).
... currently not used.

Details

The checkConsist method for objects of class aidsEst extracts all relevant elements from an object returned by aidsEst and passes them as arguments to aidsConsist. The optional argument observedShares determines whether fitted (default) or observed expenditure shares are used in the price index of the LA-AIDS to check monotonicity and in the calculation of the substitution matrix to check concavity.

Value

aidsConsist and the checkConsist method return a list of class aidsConsist that contains following elements:

addingUp logical. Is the addinup-up condition fulfilled?
homogeneity logical. Is the homogeneity condition fulfilled?
symmetry logical. Is the symmetry condition fulfilled?
mono an object returned by `aidsMono`.
conca an object returned by `aidsMono` (only if the symmetry condition is fulfilled and the translog price index is used, i.e. argument `priceIndex` is "TL".

Author(s)
Arne Henningsen

References

See Also
`aidsEst`

Examples
```r
data( Blancforti86 )
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[ 1:32, ]

priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )
estResult <- aidsEst( priceNames, shareNames, "xFood",
        data = Blancforti86, method = "IL" )
aidsConsist( priceNames, "xFood", data = Blancforti86, 
        coef = estResult$coef )

# the same can be obtained in an easier way
checkConsist( estResult )
```

Description
These functions calculate and print the demand elasticities of an AIDS model.
Usage

```r
aidsElas( coef, prices = NULL, shares = NULL, totExp = NULL, 
method = "AIDS", priceIndex = "TL", basePrices = NULL, baseShares = NULL, 
quantNames = NULL, priceNames = NULL, coefCov = NULL, df = NULL )
```

```r
# S3 method for class 'aidsEst'
elas( object, method = NULL, observedShares = FALSE, ... )
```

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

Arguments

- `coef` a list containing the coefficients alpha, beta and gamma.
- `prices` a vector of the prices at which the elasticities should be calculated.
- `shares` a vector of the shares at which the elasticities should be calculated.
- `totExp` total expenditure at which the elasticities should be calculated.
- `method` the elasticity formula to be used (see details).
- `priceIndex` the price index (see details).
- `basePrices` a vector specifying the base prices for the Paasche, Laspeyres, and Tornqvist price index.
- `baseShares` a vector specifying the base expenditure shares for the Laspeyres, simplified Laspeyres, and Tornqvist index.
- `quantNames` an optional vector of strings containing the names of the quantities to label elasticities.
- `priceNames` an optional vector of strings containing the names of the prices to label elasticities.
- `coefCov` variance covariance matrix of the coefficients (optional).
- `df` degrees of freedom to calculate P-values of the elasticities (optional).
- `object` an object of class `aidsEst`.
- `observedShares` logical. Using observed shares for calculating the demand elasticities?
- `x` an object of class `aidsElas`.
- `...` additional arguments of `elas.aidsEst` are passed to `aidsElas`; additional arguments of `print.aidsElas` are currently ignored.

Details

Currently, `aidsElas` and `elas.aidsEst` can calculate elasticities only for models without demand shifters. However, the user can calculate elasticities for models with demand shifters by removing the coefficients of the demand shifters (\(\delta_{ij}\), `coef$delta`), adjusting the coefficients of the demand shifters (\(\alpha_i\), `coef$alpha`) 'by hand', and then calling `aidsElas`. The \(\alpha_i\) coefficients should be adjusted by

```latex
\alpha_i^* = \alpha_i + \sum_{j=1}^{m} \delta_{ij} z_j \forall i = 1, \ldots, n,
```
where \( \alpha_i^* \) are the adjusted \( \alpha_i \) coefficients, \( n \) is the number of goods, \( m \) is the number of demand shifters, \( \delta_{ij} \) are the coefficients of the demand shifters, and \( z_j \) is the \( j \)'s demand shifter. Hence, the adjusted coefficients \( \alpha_i^* \) depend on the values of the demand shifters \( z \); you could, e.g., calculate different sets of elasticities for different values of \( z \) or you could use the means, medians, or modal values of \( z \).

Argument \texttt{priceIndex} has two effects: first it determines the price index that is used for calculating (fitted) expenditure shares, if argument \texttt{shares} is not provided (see \texttt{aidsCalc}); second it determines which version of the formulas for calculating demand elasticities of the LA-AIDS are used, because formulas \texttt{B1/LA}, \texttt{B2}, and \texttt{Go/Ch} have different versions depending on the price index.

e\texttt{las.la} is a wrapper function to \texttt{aidsElas} that extracts the estimated coefficients \( (\texttt{coef}) \), mean expenditure shares \( (\texttt{wMeans}) \), mean prices \( (\texttt{pMeans}) \), names of the prices \( (\texttt{priceNames}) \), estimated coefficient variance covariance matrix \( (\texttt{coef$allcov}) \), and degrees of freedom \( (\texttt{est$df}) \) from the object of class \texttt{aidsEst} and passes them to \texttt{aidsElas}. If argument \texttt{method} in \texttt{elas.la} is not specified, the default value depends on the estimation method. If the demand system was estimated by the linear approximation (LA), the default method is \texttt{`Ch'}. If the demand system was estimated by the iterative linear least squares estimator (ILLE), the default method is \texttt{`AIDS'}.

At the moment the elasticity formulas of the original AIDS (AIDS), the formula of Goddard (1983) or Chalfant (1987) (Go or Ch), the formula of Eales and Unnevehr (1988) (EU), the formula of Green and Alston (1990) or the first of Buse (1994) (GA or B1) and the second formula of Buse (1994) (B2) are implemented.

The variance covariance matrices of the elasticities are calculated using the formula of Klein (1953, p. 258) (also known as the delta method). At the moment this is implemented only for the elasticity formulas of the original AIDS.

**Value**

A list of class \texttt{aidsElas} containing following elements:

- \texttt{method} the elasticity formula used to calculate these elasticities.
- \texttt{priceIndex} the price index used (see details).
- \texttt{df} degrees of freedom to calculate P-values of the elasticities (only if argument \texttt{df} is provided).
- \texttt{exp} vector of expenditure elasticities.
- \texttt{hicks} matrix of Hicksian (compensated) price elasticities.
- \texttt{marshall} matrix of Marshallian (uncompensated) price elasticities.
- \texttt{allvcov} variance covariance matrix of all elasticities.
- \texttt{expvcov} variance covariance matrix of the expenditure elasticities.
- \texttt{hicksvcov} variance covariance matrix of the Hicksian (compensated) price elasticities.
- \texttt{marshallvcov} variance covariance matrix of the Marshallian (uncompensated) price elasticities.
- \texttt{expster} standard errors of the expenditure elasticities.
- \texttt{hicksster} standard errors of the Hicksian (compensated) price elasticities.
- \texttt{marshallster} standard errors of the Marshallian (uncompensated) price elasticities.
- \texttt{expTval} t-values of the expenditure elasticities.
hicksTval  t-values of the Hicksian (compensated) price elasticities.
marshallTval t-values of the Marshallian (uncompensated) price elasticities.
expPval    P-values of the expenditure elasticities.
hicksPval  P-values of the Hicksian (compensated) price elasticities.
marshallPval P-values of the Marshallian (uncompensated) price elasticities.

Author(s)
Arne Henningsen

References

See Also
aidsEst

Examples

data( Blancforti86 )
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[ 1:32, ]
estResult <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blancforti86 )
wMeans <- colMeans( Blancforti86[ , c( "wFood1", "wFood2", "wFood3", "wFood4" ) ] )
aidsElas( estResult$coef, shares = wMeans, method = "Ch",
priceIndex = "S" )

## Repeating the evaluation of different elasticity formulas of
## Green & Alston (1990)
priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )

# AIDS estimation and elasticities
estResultA <- aidsEst( priceNames, shareNames, "xFood",
data = Blancforti86[ -1, ],
method = "IL", maxiter = 100 )
diag( elas( estResultA, method = "AIDS" )$marshall )
summary( elas( estResultA, method = "AIDS" ) )
aidsEst does an econometric estimation of the Almost Ideal Demand System (AIDS)

Usage

```r
aidsEst( priceNames, shareNames, totExpName, data, 
  method = "LA", priceIndex = "Ls", pxBase = 1, 
  hom = TRUE, sym = TRUE, 
  shifterNames = NULL, instNames = NULL, 
  estMethod = ifelse( is.null( instNames ), "SUR", "3SLS" ), 
  ilmaxiter = 50, iltol = 1e-5, alpha0 = 0, restrict.regMat = FALSE, ... )
```

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

Arguments

- `priceNames` a vector of strings containing the names of the prices.
- `shareNames` a vector of strings containing the names of the expenditure shares.
- `totExpName` a string containing the variable name of total expenditure.
- `data` a data frame containing all required variables.
- `method` character string specifying the method to estimate the AIDS: either 'LA' or 'IL' (see details).
- `priceIndex` character string specifying the price index for the 'Linear Approximation': either 'S', 'SL', 'P', 'L', 'Ls', or 'T' (see details).
pxBase: The base to calculate the LA-AIDS price indices (see `aidsPx`).

hom: logical. Should the homogeneity condition be imposed?

sym: logical. Should the symmetry condition be imposed?

shifterNames: an optional vector of strings containing the names of the demand shifters.

instNames: a vector of strings containing the names of instrumental variables.

estMethod: estimation method (e.g. 'SUR' or '3SLS', see `systemfit`).

ILmaxiter: maximum number of iterations of the 'Iterated Linear Least Squares Estimation'.

ILtol: tolerance level of the 'Iterated Linear Least Squares Estimation'.

alpha0: the intercept of the translog price index ($\alpha_0$).

restrict.regMat: logical. Method to impose homogeneity and symmetry restrictions: either via restrict.matrix (default) or via restrict.regMat (see `systemfit`).

x: An object of class `aidsEst`.

...: additional arguments of `aidsEst` are passed to `systemfit`; additional arguments of `print.aidsEst` are currently ignored.

Details

Argument `method` can specify two different estimation methods: The 'Linear Approximate AIDS' (LA) and the 'Iterative Linear Least Squares Estimator' (IL) proposed by Blundell and Robin (1999).

Argument `priceIndex` can specify six different price indices for the LA-AIDS:

- Stone price index ('S'),
- Stone price index with lagged shares ('SL'),
- loglinear analogue to the Paasche price index ('P'),
- loglinear analogue of the Laspeyres price index ('L'),
- simplified loglinear analogue of the Laspeyres price index ('Ls'), and
- Tornqvist price index ('T').

The 'Iterative Linear Least Squares Estimator' (IL) needs starting values for the (translog) price index. Starting values are taken from an initial estimation of the 'Linear Approximate AIDS' (LA) with the price index specified by argument `priceIndex`.

Value

a list of class `aidsEst` containing following objects:

coeff: a list containing the vectors/matrix of the estimated coefficients (alpha, beta, and gamma).

r2: $R^2$-values of all share equations.

r2q: $R^2$-values of the estimated quantities.

wFitted: fitted expenditure shares.
wResid residuals of the expenditure shares.
qObs observed quantities / quantity indices.
qFitted fitted quantities / quantity indices.
qResid residuals of the estimated quantities.
est estimation result, i.e. the object returned by systemfit.
iter iterations of SUR/3SLS estimation(s). If the AIDS is estimated by the 'Iterated Linear Least Squares Estimator' (ILLE): a vector containing the SUR/3SLS iterations at each iteration.
iliter number of iterations of the 'Iterated Linear Least Squares Estimation'.
method the method used to estimate the aids (see details).
priceIndex the name of the price index (see details).
lnp log of the price index used for estimation.
pMeans means of the prices.
wMeans means of the expenditure shares.
xMean mean of total expenditure.
call the call of aidsEst.
priceNames names of the prices.
shareNames names of the expenditure shares.
totExpName name of the variable for total expenditure.
basePrices the base prices of the Paasche, Laspeyres, or Tornqvist price index.
baseShares the base shares of the Laspeyres, simplified Laspeyres, or Tornqvist price index.

Author(s)
Arne Henningsen

References


See Also
summary.aidsEst, aidsElas, aidsCalc.
Examples

# Using data published in Blanciforti, Green & King (1986)
data( Blanciforti86 )
# Data on food consumption are available only for the first 32 years
Blanciforti86 <- Blanciforti86[ 1:32, ]

## Repeating the demand analysis of Blanciforti, Green & King (1986)
## Note: Blanciforti, Green & King (1986) use scaled data,
## which leads to slightly different results
estResult <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
                      c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blanciforti86, priceIndex = "SL", maxiter = 100 )
print( estResult )
elas( estResult )

## Estimations with a demand shifter: linear trend
priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )
Blanciforti86$trend <- c( 0:( nrow( Blanciforti86 ) - 1 ) )
estResult <- aidsEst( priceNames, shareNames, "xFood",
data = Blanciforti86, shifterNames = "trend" )
print( estResult )

## Estimations with two demand shifters: linear + quadratic trend
Blanciforti86$trend2 <- c( 0:( nrow( Blanciforti86 ) - 1 ) )^2
estResult <- aidsEst( priceNames, shareNames, "xFood",
data = Blanciforti86, shifterNames = c( "trend", "trend2" ) )
print( estResult )

---

aidsMono  

Monotonicity of the AIDS

Description

aidsMono checks whether the expenditure function of an estimated Almost Ideal Demand System (AIDS) is monotonic increasing in prices, which implies that all demanded quantities and expenditure shares are non-negative.

Usage

aidsMono( priceNames, totExpName, coef, data,
          priceIndex = "TL", basePrices = NULL, baseShares = NULL )

## S3 method for class 'aidsMono'
print( x, header = TRUE, ... )
Arguments

- `priceNames` a vector of strings containing the names of the prices.
- `totExpName` a string containing the variable name of total expenditure.
- `coef` a list containing the coefficients alpha, beta, gamma, and (only for the translog price index) alpha0.
- `data` a data frame containing the data.
- `priceIndex` a character string specifying the price index (see `aidsPx`) or a numeric vector providing the log values of the price index.
- `basePrices` a vector specifying the base prices for the Paasche, Laspeyres, and Tornqvist price index.
- `baseShares` a vector specifying the base expenditure shares for the Laspeyres, simplified Laspeyres, and Tornqvist index.
- `x` an object of class `aidsMono`.
- `header` logical. Print a header?
- `...` currently unused.

Details

Internally, `aidsMono` passes its arguments to `aidsCalc` and then checks for each observation, whether all expenditure shares are non-negative.

If argument `priceIndex` specifies a price index of the LA-AIDS, 'fitted' values are used for current and lagged expenditure shares in these price indices (see `aidsCalc`). However, if argument `priceIndex` is a numeric vector containing the log values of a price index (e.g. the price index used in the estimation), this price index is used for the calculations.

Value

`aidsMono` returns a list of class `aidsMono` that contains following elements:

- `monotony` a logical vector indicating whether the monotony condition is fulfilled at each observation.
- `nValidObs` number of (valid) observation at which monotonicity could be checked.
- `nMonoObs` number of observation at which the monotonicity condition is fulfilled.
- `monoPercent` percent of observations where the monotony condition is fulfilled.
- `priceIndex` a character string indicating the price index specified by argument `priceIndex` ("numeric" if the price index is specified numerically).

Author(s)

Arne Henningsen
References


See Also

aidsEst, aidsCalc

Examples

data( Blanciforti86 )
   # Data on food consumption are available only for the first 32 years
   Blanciforti86 <- Blanciforti86[ 1:32, ]

   priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
   shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )

   ## AIDS
   estResult <- aidsEst( priceNames, shareNames, "xFood",
                         data = Blanciforti86, method = "IL" )
   aidsMono( priceNames, "xFood", coef = coef( estResult ),
             data = Blanciforti86 )

   ## LA-AIDS with Tornqvist price index
   estResultLaT <- aidsEst( priceNames, shareNames, "xFood",
                            data = Blanciforti86, priceIndex = "T" )
   # with fitted expenditure shares in the price index
   aidsMono( priceNames, "xFood", coef = coef( estResultLaT ),
             data = Blanciforti86, priceIndex = "T",
             basePrices = estResultLaT$basePrices,
             baseShares = estResultLaT$baseShares )
   # with observed expenditure shares in the price index
   aidsMono( priceNames, "xFood", coef = coef( estResultLaT ),
             data = Blanciforti86, priceIndex = estResultLaT$lnp )

---

aidsPx

Price Index for the AIDS

Description

Calculate log price indices for the AIDS and LA-AIDS.

Usage

aidsPx( priceIndex, priceNames, data, shareNames = NULL, base = 1, coef = NULL,
       shifternames = NULL )
Arguments

- **priceIndex**: the price index to be used (see details).
- **priceNames**: a vector of strings containing the names of the prices.
- **data**: a data frame containing the required variables.
- **shareNames**: a vector of strings containing the names of the expenditure shares.
- **base**: the base to calculate the indices (see details).
- **coef**: a list containing the coefficients alpha0, alpha, beta, and gamma (only needed for the translog price index).
- **shifternames**: an optional vector of strings containing the names of the demand shifters that modify the alphas of the Translog price index.

Details

Currently 7 different price indices are implemented:

- translog price index ('TL'),
- Stone price index ('S'),
- Stone price index with lagged shares ('SL'),
- loglinear analogue to the Paasche price index ('P'),
- loglinear analogue of the Laspeyres price index ('L'),
- simplified loglinear analogue of the Laspeyres price index ('Ls'), and
- Tornqvist price index ('T').

The argument base can be either

- a single number: the row number of the base prices and quantities,
- a vector indicating several observations: The means of these observations are used as base prices and quantities, or
- a logical vector with the same length as the data: The means of the observations indicated as 'TRUE' are used as base prices and quantities.
- a list with elements prices and/or shares that are numeric vectors specifying the values of the base prices and/or base expenditure shares.

Value

A vector containing the log price index. If argument priceIndex is "P", "Ls", or "T", the returned vector has the attribute(s) basePrices and/or baseShares that are numeric vectors of the base prices and/or base expenditure shares for calculating the price indices.

Author(s)

Arne Henningsen
References


See Also

`aidsEst`

Examples

```r
data( Blanciforti86 )

# Stone price index
aidsPx( "S", c( "pFood1", "pFood2", "pFood3", "pFood4" ),
        Blanciforti86, c( "wFood1", "wFood2", "wFood3", "wFood4" ) )

# (log-linear analogue of the) Paasche price index
aidsPx( "P", c( "pFood1", "pFood2", "pFood3", "pFood4" ),
        Blanciforti86, c( "wFood1", "wFood2", "wFood3", "wFood4" ),
        base = row.names(Blanciforti86) == "1970" )

# Tornqvist price index
aidsPx( "T", c( "pFood1", "pFood2", "pFood3", "pFood4" ),
        Blanciforti86, c( "wFood1", "wFood2", "wFood3", "wFood4" ),
        base = list( prices = rep( 100, 4 ), shares = rep( 0.25, 4 ) ) )
```

---

**aidsUtility**

*Indirect Utility Function of the Almost Ideal Demand System*

Description

These functions calculate the utility level given prices and total expenditure using the indirect utility function of the Almost Ideal Demand System and the partial derivatives of this indirect utility function with respect to prices and total expenditure.

Usage

```r
aidsUtility( priceNames, totExpName, coef, data )
aidsUtilityDeriv( priceNames, totExpName, coef, data, rel = FALSE )
```
aidsUtility

Arguments
priceNames  a vector of strings containing the names of the prices.
totExpName  a string containing the variable name of total expenditure.
coef        a list containing the coefficients in elements alpha0 (scalar), alpha (vector), beta (vector), gamma (matrix), and possibly beta0 (scalar, if not given, it is assumed to be 1).
data        a data frame containing the data.
rel         logical. If TRUE the returned partial derivatives are given in relative terms (like elasticities), i.e., they indicate the percentage change in the utility level when a price or total expenditure is increased by 1%.

Value
aidsUtility returns a numeric vector that contains the utility levels; aidsUtilityDeriv returns a data.frame that contains the partial derivatives of the indirect utility function with respect to prices and total expenditure.

Author(s)
Arne Henningsen

References

See Also
aidsEst, aidsCalc

Examples

data( Blancforti86 )
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[ 1:32, ]

priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )

## estimate the (non-linear) AIDS
estResult <- aidsEst( priceNames, shareNames, "xFood",
data = Blancforti86, method = "IL" )

# calculate the utility levels of each year
utility <- aidsUtility( priceNames, "xFood", coef = coef( estResult ),
data = Blancforti86 )

utilityDeriv <- aidsUtilityDeriv( priceNames, "xFood",
coef = coef( estResult ), data = Blancforti86 )
utilityEla <- aidsUtilityDeriv( priceNames, "xFood", 
  coef = coef( estResult ), data = Blanciforti86, rel = TRUE )

Blanciforti86  U.S. consumption data

Description

The Blanciforti86 data frame contains annual U.S. consumption data from 1947 to 1978 or 1981. These data include eleven aggregate commodity groups: (1) food, (2) alcohol plus tobacco, (3) clothing, (4) housing, (5) utilities, (6) transportation, (7) medical care, (8) durable goods, (9) other nondurable goods, (10) other services, and (11) other miscellaneous goods; four food groups: (1) meats, (2) fruits and vegetables, (3) cereal and bakery products, and (4) miscellaneous foods; and four meat groups: (1) beef and veal, (2) pork, (3) fish, and (4) poultry. For each of these groups the consumption expenditures, price indices, and expenditure shares are available.

This data frame contains the following variables / columns:

- year  The year.
- xAggX  Expenditure on the aggregate commodity group X (in Millions of US-Dollars).
- xAgg  Total expenditure on all eleven aggregate commodity groups (in Millions of US-Dollars).
- xcAggX  Deflated expenditure on the aggregate commodity group X (in Million of 1972 US-Dollars).
- xcAgg  Total deflated expenditure on all eleven aggregate commodity groups (in Million of 1972 US-Dollars).
- pAggX  Price index for the aggregate commodity group X (1972 = 100).
- wAggX  Expenditure share of the aggregate commodity group X.
- xFoodX  Per capita expenditure for food group X (in US-Dollars).
- xFood  Total per capita expenditure for all four food groups (in US-Dollars).
- xcFood  Total deflated per capita expenditure for all four food groups (in 1972 US-Dollars).
- pFoodX  Price index for food group X (1972 = 100).
- wFoodX  Expenditure share of food group X.
- xFoodUsdoc  Expenditure on food published by the US Department of Commerce (USDOC) (in Millions of US-Dollars).
- xFoodUsda  Expenditure on food published by the US Department of Agriculture (USDA) (in Millions of US-Dollars).
- xFoodNew  Expenditure on food (new estimates) (in Millions of US-Dollars).
- xMeatX  Per capita expenditure for meat group X (in US-Dollars).
- xMeat  Total per capita expenditure for all four meat groups (in US-Dollars).
- pMeatX  Price index for meat group X (1972 = 100).
- cMeatX  Per capita consumption of meat group X (in pounds).
**coef.aidsEst**

- **pMeat** Price index for the meat group (1972 = 100).
- **pMeatNew** Price index for the meat group (new estimate) (1972 = 100).
- **population3** Population as presented in Table 5.A.3 (in Millions).
- **population12** Population as presented in Table 5.A.12 (in Millions).

**Usage**

```
data(Blanciforti86)
```

**Source**


---

**coef.aidsEst**

*Coefficients of an Almost Ideal Demand System*

**Description**

These methods return and print the coefficients from an Almost Ideal Demand System.

**Usage**

```
## S3 method for class 'aidsEst'
coef( object, ... )

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

**Arguments**

- **object** an object of class aidsEst.
- **x** an object of class coef.aidsEst.
- **...** further arguments for methods

**Value**

The coef method returns an object of class coef.aidsEst containing following objects:

- **alpha0** a scalar, coefficient alpha0 (only for the AIDS with translog price index).
- **alpha** a vector of the alpha coefficients.
- **beta** a vector of the beta coefficients.
- **gamma** a matrix of the gamma coefficients.
- **delta** a matrix of the delta coefficients (only if the model was estimated with demand shifters).
Author(s)

Arne Henningsen

See Also

aidsEst

---

df.residual.aidsEst  Covariance matrix of an Almost Ideal Demand System

Description

These method returns the covariance matrix of the coefficients from an Almost Ideal Demand System (AIDS).

Usage

```r
## S3 method for class 'aidsEst'
df.residual( object, ... )
```

Arguments

- `object` an object of class `aidsEst`.
- `...` currently not used.

Value

The `df.residual` method for objects of class `aidsEst` returns a symmetric matrix: the covariance matrix of the coefficients.

Author(s)

Arne Henningsen

See Also

`aidsEst`, `coef.aidsEst`
Description

This method extracts the fitted demanded quantities and expenditure shares from an estimated Almost Ideal Demand System.

Usage

```r
## S3 method for class 'aidsEst'
fitted( object, ... )
```

Arguments

- `object` an object of class `aidsEst`.
- `...` currently unused.

Value

The `fitted` method returns a list containing following objects:

- `shares` a data frame for the fitted expenditure shares.
- `quant` a data frame for the fitted demanded quantities.

Author(s)

Arne Henningsen

See Also

`aidsEst`

Examples

```r
# Using data published in Blancforti, Green & King (1986)
data( Blancforti86 )
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[ , 1:32, ]

estResult <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
                       c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
                       data = Blancforti86 )
fitted( estResult )
```
logLik.aidsEst  

*Log-Likelihood value of an object of class aidsEst*

---

**Description**

This method extracts the log-likelihood value of a fitted Almost Ideal Demand System (AIDS).

**Usage**

```r
## S3 method for class 'aidsEst'
logLik( object, ... )
```

**Arguments**

- `object`: an object of class `aidsEst`.
- `...`: currently ignored.

**Value**

A numeric scalar (the log-likelihood value) with 2 attributes: `nobs` (total number of observations in all equations) and `df` (number of free parameters, i.e. coefficients + elements of the residual covariance matrix).

**Author(s)**

Arne Henningsen

**See Also**

`aidsEst`.

**Examples**

```r
# Using data published in Blancforti, Green & King (1986)
data( Blanciforti86 )
# Data on food consumption are available only for the first 32 years
Blanciforti86 <- Blanciforti86[ 1:32, ]

## Repeating the demand analysis of Blancforti, Green & King (1986)
estResult <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blanciforti86, priceIndex = "SL" )
logLik( estResult )
```
lrtest.aidsEst

Likelihood Ratio test for Almost Ideal Demand Systems

Description

Testing hypothesis in Almost Ideal Demand Systems by a Likelihood Ratio test.

Usage

```r
## S3 method for class 'aidsEst'
lrtest( object, ... )
```

Arguments

- `object`: a fitted model object of class `aidsEst`.
- `...`: further fitted model objects of class `aidsEst`.

Details

`lrtest.aidsEst` consecutively compares the fitted model object `object` with the models passed in `...`

Value

An object of class `anova`, which contains the log-likelihood value, degrees of freedom, the difference in degrees of freedom, likelihood ratio Chi-squared statistic and corresponding p value. See documentation of `lrtest` in package "lmtest".

Author(s)

Arne Henningsen

See Also

`aidsEst`, `lrtest` (package "lmtest"), `logLik.aidsEst`

Examples

```r
# Using data published in Blancforti, Green & King (1986)
data( Blancforti86 )
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[ 1:32, ]

# names of prices and expenditure shares
priceNames <- c( "pFood1", "pFood2", "pFood3", "pFood4" )
shareNames <- c( "wFood1", "wFood2", "wFood3", "wFood4" )

# estimation with symmetry and homogeneity imposed
estResult <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
```
c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blanciforti86 )

# estimation with only homogeneity imposed
estResultHom <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blanciforti86, sym = FALSE )

# unrestricted estimation
estResultUnr <- aidsEst( c( "pFood1", "pFood2", "pFood3", "pFood4" ),
c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blanciforti86, sym = FALSE, hom = FALSE )

# LR tests
lrtest( estResult, estResultHom, estResultUnr, estResult )

## Estimation with a demand shifter: linear trend
Blanciforti86$trend <- c( @:( nrow( Blanciforti86 ) - 1 ) )
estResultTrend <- aidsEst( priceNames, shareNames, "xFood",
data = Blanciforti86, shifterNames = "trend" )

# LR tests
lrtest( estResult, estResultTrend )

---

**summary.aidsElas**  
*Summarizing the Elasticities of an Almost Ideal Demand System*

**Description**

These functions summarize and print the estimated elasticities of an Almost Ideal Demand System (AIDS).

**Usage**

```r
## S3 method for class 'aidsElas'
summary( object, ... )

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

**Arguments**

- `object` an object of class `aidsElas`.
- `x` an object of class `summary.aidsElas`.
- `...` currently ignored.
Value

`summary.aidsEst` returns a list of class `summary.aidsElas`. It is identical to the provided object (except for its class), but it contains following additional element:

- **table**
  A matrix with 4 columns: all elasticities, their standard errors (if available), their t-values (if available), and their P-values (if available).

Author(s)

Arne Henningsen

See Also

`aidsElas, aidsEst`

Examples

```r
# Using data published in Blanciforti, Green & King (1986)
data(Blanciforti86)
# Data on food consumption are available only for the first 32 years
Blanciforti86 <- Blanciforti86[ 1:32, ]

## Repeating the demand analysis of Blanciforti, Green & King (1986)
estResult <- aidsEst( c("pFood1", "pFood2", "pFood3", "pFood4" ),
c( "wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
data = Blanciforti86, method = "IL" )
summary( elas( estResult ) )
```

Description

`summary.aidsEst` summarizes the estimation results of an Almost Ideal Demand System (AIDS).

Usage

```r
## S3 method for class 'aidsEst'
summary( object, ... )

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

Arguments

- **object**
  An object of class `aidsEst`.

- **x**
  An object of class `summary.aidsEst`.

- **...**
  Currently ignored.
Value

`summary.aidsEst` returns a list of class `summary.aidsEst` that is currently identical to the provided object (except for its class).

Author(s)

Arne Henningsen

See Also

`aidsEst`, `aidsElas`.

Examples

```r
# Using data published in Blancforti, Green & King (1986)
data(Blancforti86)
# Data on food consumption are available only for the first 32 years
Blancforti86 <- Blancforti86[1:32,]

## Repeating the demand analysis of Blancforti, Green & King (1986)
estResult <- aidsEst( c("pFood1", "pFood2", "pFood3", "pFood4"),
  c("wFood1", "wFood2", "wFood3", "wFood4" ), "xFood",
  data = Blancforti86, priceIndex = "SL" )
print( summary(estResult) )
```

---

**USMeatConsump**

*U.S. Meat Consumption Data*

**Description**

The USMeatConsump data set contains quarterly retail prices and consumption quantities for four meat product categories: beef, pork, chicken, and turkey. The data period ranges from the first quarter of 1975 to the third quarter of 1999. Hence, there are 99 observations.

**Usage**

data(USMeatConsump)

**Format**

This data frame contains the following columns:

- year  Year.
- qtr   Quarter of the year.
- t     Time trend.
- pop   Population [million].
- cpi   Consumer price index.
total_exp  Total per capita expenditure.
meat_exp  Per capita expenditure on meat.
beef_q  Per capita consumption of beef [pound].
pork_q  Per capita consumption of pork [pound].
chick_q  Per capita consumption of chicken [pound].
turkey_q  Per capita consumption of turkey [pound].
beef_p  Retail price of beef [cents / pound].
pork_p  Retail price of pork [cents / pound].
chick_p  Retail price of chicken [cents / pound].
turkey_p  Retail price of turkey [cents / pound].
beef_w  Expenditure share of beef (in meat).
pork_w  Expenditure share of pork (in meat).
chick_w  Expenditure share of chicken (in meat).
turkey_w  Expenditure share of turkey (in meat).

Source

Examples

```r
## replicating the LA-AIDS estimation of the SAS example
# loading data set
data(USMeatConsump)

# adding shifter variables for modeling seasonal effects
USMeatConsump$co1 <- cos(1/2*3.14159*USMeatConsump$t)
USMeatConsump$sil <- sin(1/2*3.14159*USMeatConsump$t)

# Scaling prices by their means
USMeatConsump$beef_pm <- USMeatConsump$beef_p/mean(USMeatConsump$beef_p)
USMeatConsump$pork_pm <- USMeatConsump$pork_p/mean(USMeatConsump$pork_p)
USMeatConsump$chick_pm <- USMeatConsump$chick_p/mean(USMeatConsump$chick_p)
USMeatConsump$turkey_pm <- USMeatConsump$turkey_p/mean(USMeatConsump$turkey_p)

# Estimation of the model
meatModel <- aidsEst(c("beef_pm", "pork_pm", "chick_pm", "turkey_pm"),
                     c("beef_w", "pork_w", "chick_w", "turkey_w"),
                     "meat_exp", shifterNames = c("co1", "sil", "t"),
                     priceIndex ="S", data = USMeatConsump, maxiter=1000)
summary(meatModel)
```

**vcov.aidsEst**  
*Covariance matrix of an Almost Ideal Demand System*

**Description**
These method returns the covariance matrix of the coefficients from an Almost Ideal Demand System (AIDS).

**Usage**
```r
## S3 method for class 'aidsEst'
vcov( object, ... )
```

**Arguments**
- `object` an object of class `aidsEst`.
- `...` currently not used.

**Value**
The `vcov` method for objects of class `aidsEst` returns a symmetric matrix: the covariance matrix of the coefficients.

**Author(s)**
Arne Henningsen

**See Also**
`aidsEst`, `coef.aidsEst`
Index

*Topic **datasets**
   Blanciforti86, 22
   USMeatConsump, 30

*Topic **methods**
   coef.aidsEst, 23
   df.residual.aidsEst, 24
   fitted.aidsEst, 25
   vcov.aidsEst, 32

*Topic **models**
   aidsBestA0, 2
   aidsCalc, 3, 11, 15, 17, 18, 21
   aidsConcav, 6, 8
   aidsConsist, 7
   aidsElas, 9
   aidsEst, 13
   aidsMono, 16
   aidsPx, 18
   aidsUtility, 20
   logLik.aidsEst, 26
   lrtest.aidsEst, 27
   summary.aidsElas, 28
   summary.aidsEst, 29

   aidsBestA0, 2
   aidsCalc, 3, 11, 15, 17, 18, 21
   aidsConcav, 6, 8
   aidsConsist, 7
   aidsElas, 9, 15, 29, 30
   aidsEst, 2, 3, 5, 7, 9, 12, 13, 18, 20, 21,
   24–27, 29, 30, 32
   aidsMono, 8, 9, 16
   aidsPx, 4, 5, 8, 14, 17, 18
   aidsUtility, 20
   aidsUtilityDeriv (aidsUtility), 20
   Blanciforti86, 22

   checkConsist.aidsEst (aidsConsist), 7
   coef.aidsEst, 23, 24, 32
   df.residual.aidsEst, 24
   elas.aidsEst (aidsElas), 9
   fitted.aidsEst, 25
   logLik.aidsEst, 26, 27
   lrtest, 27
   lrtest.aidsEst, 27
   predict.aidsEst (aidsCalc), 3
   print.aidsConcav (aidsConcav), 6
   print.aidsConsist (aidsConsist), 7
   print.aidsElas (aidsElas), 9
   print.aidsEst (aidsEst), 13
   print.aidsMono (aidsMono), 16
   print.coef.aidsEst (coef.aidsEst), 23
   print.summary.aidsElas
     (summary.aidsElas), 28
   print.summary.aidsEst
     (summary.aidsEst), 29

   summary.aidsElas, 28
   summary.aidsEst, 15, 29
   systemfit, 14, 15
   USMeatConsump, 30
   vcov.aidsEst, 32