Package 'micEcon'

October 13, 2022

Version 0.6-18

Date 2022-09-03

Title Microeconomic Analysis and Modelling

Author Arne Henningsen

Maintainer Arne Henningsen <arne.henningsen@gmail.com>

Depends R (>= 3.5.0)

Imports miscTools (>= 0.6-1), plm (>= 1.1-0)

Suggests Ecdat (>= 0.1-5), systemfit (>= 1.0-0)

Description Various tools for microeconomic analysis and microeconomic modelling,
e.g. estimating quadratic, Cobb-Douglas and Translog functions,
calculating partial derivatives and elasticities of these functions,
and calculating Hessian matrices, checking curvature
and preparing restrictions for imposing monotonicity of Translog functions.

License GPL (>= 2)

URL http://www.micEcon.org

NeedsCompilation no

Repository CRAN

Date/Publication 2022-09-03 21:30:02 UTC

R topics documented:

`appleProdFr86` *Data on French Apple Producers in 1986*

Description

The `appleProdFr86` data frame includes cross-sectional production data of 140 French apple producers from the year 1986. These data have been extracted from a panel data set that was used in Ivaldi et al. (1996).

Usage

```
data(appleProdFr86)
```

Format

This data frame contains the following columns:

vCap costs of capital (including land).

vLab costs of labour (including remuneration of unpaid family labour).

vMat costs of intermediate materials (e.g. seedlings, fertilizer, pesticides, fuel).

qApples quantity index of produced apples.

qOtherOut quantity index of all other outputs.

qOut quantity index of all outputs (not in the original data set, calculated as 580,000 · (`qApples` + `qOtherOut`)).

pCap price index of capital goods

pLab price index of labour.

pMat price index of materials.

pOut price index of the aggregate output (not in the original data set, artificially generated).

adv dummy variable indicating the use of an advisory service (not in the original data set, artificially generated).

Source

This cross-sectional data set has been extracted from a panel data set that is available in the data archive of the Journal of Applied Econometrics: http://qed.econ.queensu.ca/jae/1996-v11.6/ivaldi-ladoux-ossard-simioni/.

References

Ivaldi, M., N. Ladoux, H. Ossard, and M. Simioni (1996) Comparing Fourier and Translog Specifications of Multiproduct Technology: Evidence from an Incomplete Panel of French Farmers. *Journal of Applied Econometrics*, 11(6), p. 649-667.

Bleymueller79E25.1 *Artificial Prices and Quantities*

Description

The Bleymueller251 data frame contains prices and quantities of 4 products for the years 1970, 1974 and 1978. This data are part of Exercise 25.1 of Bleymueller, Gehler und Guetlicher (1979).

Usage

```
data(Bleymueller79E25.1)
```

Format

This data frame contains the following columns:

p.A Price of good A.

p.B Price of good B.

p.C Price of good C.

p.D Price of good D.

q.A Quantity of good A.

q.B Quantity of good B.

q.C Quantity of good C.

q.D Quantity of good D.

Source

Bleymueller, J; G. Gehlert and H. Guelicher (1979) Statistik fuer Wirtschaftswissenschaftler. Verlag Vahlen, Muenchen.

checkConsist *Testing Theoretical Consistency*

Description

Test theoretical consistency of microeconomic models.

Usage

```
checkConsist( object, ... )
```

Arguments

object	a microeconomic model
...	further arguments for methods

Details

This is a generic function.

Author(s)

Arne Henningsen

See Also

checkConsist.aidsEst

cobbDouglasCalc *Calculate dependent variable of a Cobb-Douglas function*

Description

Calculate the dependent variable of a Cobb-Douglas function.

Usage

```
cobbDouglasCalc( xNames, data, coef, coefCov = NULL, dataLogged = FALSE  )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	data frame containing the data.
coef	vector containing the coefficients: if the elements of the vector have no names, the first element is taken as intercept of the *logged* equation and the following elements are taken as coefficients of the independent variables defined in argument xNames (in the same order); if the elements of coef have names, the element named a_0 is taken as intercept of the *logged* equation and the elements named a_1, ..., a_n are taken as coefficients of the independent variables defined in argument xNames (numbered in that order).
coefCov	optional covariance matrix of the coefficients (the order of the rows and columns must correspond to the order of the coefficients in argument coef).
dataLogged	logical. Are the values in data already logged?

Value

A vector containing the endogenous variable. If the inputs are provided as logarithmic values (argument dataLogged is TRUE), the endogenous variable is returned as logarithm; non-logarithmic values are returned otherwise.

If argument coefCov is specified, the returned vector has an attribute "variance" that is a vector containing the variances of the calculated (fitted) endogenous variable.

Author(s)

Arne Henningsen

See Also

translogCalc, cobbDouglasOpt.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a Cobb-Douglas production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, linear = TRUE )

# fitted values
fitted <- cobbDouglasCalc( c( "qLabor", "land", "qVarInput", "time" ), germanFarms,
   coef( estResult )[ 1:5 ] )
#equal to estResult$fitted
```

```
# fitted values and their variances
fitted2 <- cobbDouglasCalc( c( "qLabor", "land", "qVarInput", "time" ), germanFarms,
   coef( estResult )[ 1:5 ], coefCov = vcov( estResult )[ 1:5, 1:5 ] )
# t-values
c( fitted2 ) / attributes( fitted2 )$variance^0.5
```

---

cobbDouglasDeriv *Derivatives of a Cobb-Douglas function*

Description

Calculate the derivatives of a Cobb-Douglas function.

Usage

```
cobbDouglasDeriv( xNames, data, coef, coefCov = NULL,
   yName = NULL, dataLogged = FALSE )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	data frame containing the data.
coef	vector containing the coefficients: if the elements of the vector have no names, the first element is taken as intercept of the *logged* equation and the following elements are taken as coefficients of the independent variables defined in argument xNames (in the same order); if the elements of coef have names, the element named a_0 is taken as intercept of the *logged* equation and the elements named a_1, ..., a_n are taken as coefficients of the independent variables defined in argument xNames (numbered in that order).
coefCov	optional covariance matrix of the coefficients (the order of the rows and columns must correspond to the order of the coefficients in argument coef).
yName	an optional string containing the name of the dependent variable. If it is NULL, the dependent variable is calculated from the independent variables and the coefficients.
dataLogged	logical. Are the values in data already logged?

Value

a list of class cobbDouglasDeriv containing following objects:

deriv	data frame containing the derivatives.
variance	data frame containing the variances of the derivatives (only if argument coefCov is provided). NOTE: if argument yName is specified, the variance of the endogenous variable is currently ignored.

Author(s)

Arne Henningsen

See Also

cobbDouglasCalc, translogDeriv.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a Cobb-Douglas production function
estResult <- translogEst( "qOutput", c( "qLabor", "qVarInput", "land", "time" ),
   germanFarms, linear = TRUE )

# compute the marginal products of the inputs (with "fitted" Output)
margProducts <- cobbDouglasDeriv( c( "qLabor", "qVarInput", "land", "time" ),
   data = germanFarms, coef = coef( estResult )[1:5],
   coefCov = vcov( estResult )[1:5,1:5] )
margProducts$deriv
# t-values
margProducts$deriv / margProducts$variance^0.5

# compute the marginal products of the inputs (with observed Output)
margProductsObs <- cobbDouglasDeriv( c( "qLabor", "qVarInput", "land", "time" ),
   data = germanFarms, coef = coef( estResult )[1:5], yName = "qOutput",
   coefCov = vcov( estResult )[1:5,1:5] )
margProductsObs$deriv
# t-values
margProductsObs$deriv / margProductsObs$variance^0.5
```

---

cobbDouglasOpt *Optimal Values of Independent Variables of a Cobb-Douglas Function*

Description

Calculate the optimal values of the variable independent variables of a Cobb-Douglas function.

Usage

```
cobbDouglasOpt( pyName, pxNames, data, coef,
   zNames = NULL, zCoef = NULL, xNames = NULL, dataLogged = FALSE  )
```

Arguments

pyName	character string containing the name of the price of the dependent variable.
pxNames	a vector of strings containing the names of the prices of the variable independent variables.
data	data frame containing the data.
coef	vector containing the intercept and the coefficients of the variable independent variables: if the elements of the vector have no names, the first element is taken as intercept of the *logged* equation and the following elements are taken as coefficients of the variable independent variables with corresponding prices defined in argument pxNames (in the same order); if the elements of coef have names, the element named a_0 is taken as intercept of the *logged* equation and the elements named a_1, ..., a_n are taken as coefficients of the variable independent variables with corresponding prices defined in argument xNames (numbered in that order).
zNames	optional vector of strings containing the names of the fixed independent variables.
zCoef	vector containing the coefficients of the fixed independent variables: if the elements of the vector have no names, they are taken as coefficients of the fixed independent variables defined in argument zNames (in the same order); if the elements of coef have names, the elements named d_1, ..., d_m are taken as coefficients of the fixed independent variables with corresponding prices defined in argument zNames (numbered in that order).
xNames	optional vector of strings containing the names that should be assigned to the returned variable independent variables.
dataLogged	logical. Are the prices and fixed independent variables in data with names defined in pyName, pxNames, and zNames already logged?

Value

A data frame containing the optimal values of the variable independent variables. If the prices and fixed independent variables are provided as logarithmic values (argument dataLogged is TRUE), the optimal values of the variable independent variables are returned as logarithms, too; non-logarithmic values are returned otherwise.

Author(s)

Arne Henningsen

See Also

cobbDouglasCalc.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
```

```
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a Cobb-Douglas production function
estResult <- translogEst( yName = "qOutput",
   xNames = c( "qLabor", "qVarInput", "land", "time" ),
   data = germanFarms, linear = TRUE )

# calculate optimal quantities of variable inputs
xCoef <- coef( estResult )[ 1:3 ]
zCoef <- coef( estResult )[ 4:5 ]
names( zCoef ) <- c( "d_1", "d_2" )
optInput <- cobbDouglasOpt( pyName = "pOutput",
   pxNames = c( "pLabor", "pVarInput" ), coef = xCoef,
   data = germanFarms, xNames = c( "qLabor", "qVarInput" ),
   zNames = c( "land", "time" ), zCoef = zCoef )

# compare observed with optimal input quantities
plot( germanFarms$qLabor, optInput$qLabor )
plot( germanFarms$qVarInput, optInput$qVarInput )
```

`coef.quadFuncEst` *Coefficients of a Quadratic Function*

Description

These methods return the coefficients and their covariance matrix from an estimated quadratic function.

Usage

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

   ## S3 method for class 'quadFuncEst'
vcov( object, ... )
```

Arguments

`object`	an object of class `quadFuncEst`.
`...`	currently ignored.

Value

The `coef` method returns a vector containing all (linearly independent) coefficients of a quadratic function.

The `vcov` method returns the covariance matrix of all (linearly independent) coefficients of a quadratic function.

Author(s)

Arne Henningsen

See Also

quadFuncEst

---

coef.translogEst *Coefficients of a Translog Function*

Description

These methods return the coefficients and their covariance matrix from an estimated translog function.

Usage

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

   ## S3 method for class 'translogEst'
vcov( object, ... )
```

Arguments

object	an object of class translogEst.
...	currently ignored.

Value

The coef method returns a vector containing all (linearly independent) coefficients of a translog function.

The vcov method returns the covariance matrix of all (linearly independent) coefficients of a translog function.

Author(s)

Arne Henningsen

See Also

translogEst

elas *Calculating and returning elasticities*

Description

These functions calculate and return elasticities of microeconomic models. `elasticities` is an alias for `elas`.

Usage

```
   elas( object, ... )
   elasticities( object, ... )
   ## Default S3 method:
elas( object, ... )
```

Arguments

`object`	a microeconomic model
`...`	further arguments for methods

Details

This is a generic function. The default method just returns the element `elas` from `object`.

Author(s)

Arne Henningsen

See Also

`elas.aidsEst`

germanFarms *Output and Inputs of Farms in West-Germany*

Description

The `germanFarms` data frame contains annual data of an average full-time farm in West-Germany. Additionally, the price indices for agricultural output and agricultural variable input are included. 20 book-keeping years are included - starting in 1975/76 and ending in 1994/95.

Usage

```
data(germanFarms)
```

Format

This data frame contains the following columns:

year the book-keeping year.

vCrop the value of crop outputs (in current Deutschmark).

vAnimal the value of animal outputs (in current Deutschmark).

vOutput the value of outputs (in current Deutschmark).

pOutput price index of agricultural outputs (1980/81 = 100).

vVarInput the value of variable inputs (in current Deutschmark).

pVarInput price index of variable agricultural inputs (1980/81 = 100).

qLabor the number of full-time worker equivalents.

pLabor costs of an agricultural worker (Deutschmarks per year).

land land used for agricultural production (in ha).

Source

Bundesministerium für Ernährung, Landwirtschaft und Forsten (Federal Department for Food, Agriculture and Forests), *Agrarbericht der Bundesregierung (Agricultural Report of the Federal Government)*, Jahrgänge 1977-1996 (years 1977-1996).

---

logDataSet *Creating a Data Set with the Logarithms of the Original Variables*

Description

This function creates a data set with the logarithms of the original variables.

Usage

```
logDataSet( data, varNames, varNamesNum = NULL  )
```

Arguments

`data`	a data frame containing the data (possibly a panel data frame created with `pdata.frame`).
`varNames`	vector of character strings that indicates names of variables in the data frame. The logarithm of these variables are included in the returned data frame.
`varNamesNum`	optional vector of character strings that indicates names of further variables in tha data frame. In case of numeric variables, the logarithms of these variabes are included in the returned data frame. In case of factor of logical variables, these variabes are included in the returned data frame without any transformation.

Author(s)

Arne Henningsen

Examples

```
data( "germanFarms" )
datLog <- logDataSet( germanFarms, c( "vAnimal", "vOutput", "vVarInput" ) )
summary( datLog )
```

`Missong03E7.7` *Meat Prices and Quantities in Germany*

Description

The `Missong03E7.7` data frame contains meat prices and demanded quantities of a representative (West-)German household for the years 1986 to 1989. This data are part of Exercise 7.7 of Missong (2003).

Usage

```
data(Missong03E7.7)
```

Format

This data frame contains the following columns:

p.beef Average price of beef (DM/kg).

q.beef Demanded Quantity of beef (kg).

p.veal Average price of veal (DM/kg).

q.veal Demanded Quantity of veal (kg).

p.pork Average price of pork (DM/kg).

q.pork Demanded Quantity of pork (kg).

Source

Missong, M. (2003) Aufgabensammlung zur deskriptiven Statistik, Oldenbourg, Muenchen.

Statistisches Bundesamt (1989) Fachserie 15, Reihe 1, p. 76f.

quadFuncCalc *Calculate dependent variable of a quadratic function*

Description

Calculate the dependent variable of a quadratic function.

Usage

```
quadFuncCalc( xNames, data, coef, shifterNames = NULL,
   homWeights = NULL )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe or a vector with named elements containing the data.
coef	vector containing all coefficients: if there are n exogenous variables in xNames and m shifter variables in shifterNames, the n+1 alpha coefficients must have names a_0, ..., a_n, the n*(n+1)/2 beta coefficients must have names b_1_1, ..., b_1_n, ..., b_n_n, and the m delta coefficients must have names d_1, ..., d_m (only the elements of the upper right triangle of the beta matrix are directly obtained from coef; the elements of the lower left triangle are obtained by assuming symmetry of the beta matrix).
shifterNames	a vector of strings containing the names of the independent variables that should be included as shifters only (not in quadratic or interaction terms).
homWeights	numeric vector with named elements that are weighting factors for calculating an index that is used to normalize the variables for imposing homogeneity of degree zero in these variables (see documentation of quadFuncEst).

Value

a vector containing the endogenous variable.

Author(s)

Arne Henningsen

See Also

quadFuncEst and quadFuncDeriv.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- quadFuncEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

quadFuncCalc( c( "qLabor", "land", "qVarInput", "time" ), germanFarms,
   coef( estResult ) )
#equal to estResult$fitted
```

---

quadFuncDeriv *Derivatives of a quadratic function*

Description

Calculate the derivatives of a quadratic function.

Usage

```
quadFuncDeriv( xNames, data, coef, coefCov = NULL,
   homWeights = NULL )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe or a vector with named elements containing the data.
coef	vector containing all coefficients: if there are n exogenous variables in xNames, the n+1 alpha coefficients must have names a_0, ..., a_n and the n*(n+1)/2 beta coefficients must have names b_1_1, ..., b_1_n, ..., b_n_n (only the elements of the upper right triangle of the beta matrix are directly obtained from coef; the elements of the lower left triangle are obtained by assuming symmetry of the beta matrix).
coefCov	optional covariance matrix of the coefficients: the row names and column names must be the same as the names of coef.
homWeights	numeric vector with named elements that are weighting factors for calculating an index that is used to normalize the variables for imposing homogeneity of degree zero in these variables (see documentation of quadFuncEst).

Details

Shifter variables do not need to be specified, because they have no effect on the partial derivatives. Hence, you can use this function to calculate partial derivatives even for quadratic functions that have been estimated with shifter variables.

Value

A data frame containing the derivatives, where each column corresponds to one of the independent variables. If argument `coefCov` is provided, it has the attributes `variance` and `stdDev`, which are two data frames containing the variances and the standard deviations, respectively, of the derivatives.

Author(s)

Arne Henningsen

See Also

`quadFuncEst` and `quadFuncCalc`

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- quadFuncEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# compute the marginal products of the inputs
margProducts <- quadFuncDeriv( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), vcov( estResult ) )
# all marginal products
margProducts
# their t-values
margProducts / attributes( margProducts )$stdDev
```

---

quadFuncEla *Elasticities of a Quadratic Function*

Description

Calculate elasticities of a quadratic function.

Usage

```
quadFuncEla( xNames, data, coef, yName = NULL,
   shifterNames = NULL, homWeights = NULL )

## S3 method for class 'quadFuncEst'
elas( object, data = NULL, yObs = FALSE, ... )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe or a vector with named elements containing the data; if argument data of elas.quadFuncEst is not specified, the data frame that was used for the estimation is used for calculating elasticities.
coef	vector containing all coefficients.
yName	an optional string containing the name of the dependent variable. If it is NULL, the dependent variable is calculated from the independent variables and the coefficients.
shifterNames	an optional vector of strings containing the names of the independent variables that are included as shifters only (not in quadratic or interaction terms).
homWeights	numeric vector with named elements that are weighting factors for calculating an index that is used to normalize the variables for imposing homogeneity of degree zero in these variables (see documentation of quadFuncEst).
object	object of class quadFuncEst (returned by quadFuncEst).
yObs	logical. Use observed values of the endogenous variable. If FALSE (default) predicted values calculated by quadFuncCalc are used.
...	currently ignored.

Value

A data.frame of class quadFuncEla, where each column corresponds to one of the independent variables.

Author(s)

Arne Henningsen

See Also

quadFuncEst, quadFuncDeriv, and quadFuncCalc.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
```

```
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- quadFuncEst( yName = "qOutput",
   xNames = c( "qLabor", "land", "qVarInput", "time" ),
   data = germanFarms )

# compute the partial production elasticities with "fitted" output
elaFit <- quadFuncEla( xNames = c( "qLabor", "land", "qVarInput", "time" ),
   data = germanFarms, coef = coef( estResult ) )
elaFit
# same as
elaFit2 <- elas( estResult )
all.equal( elaFit, elaFit2 )

# compute the partial production elasticities with observed output
elaObs <- quadFuncEla( xNames = c( "qLabor", "land", "qVarInput", "time" ),
   data = germanFarms, coef = coef( estResult ), yName = "qOutput" )
elaObs
# same as
elaObs2 <- elas( estResult, yObs = TRUE )
all.equal( elaObs, elaObs2 )
```

---

quadFuncEst *Estimate a quadratic function*

Description

Estimate a quadratic function.

Usage

```
quadFuncEst( yName, xNames, data, shifterNames = NULL,
   linear = FALSE, homWeights = NULL,
   regScale = 1, ... )
```

Arguments

yName	a character string containing the name of the dependent variable.
xNames	a vector of strings containing the names of the independent variables.
data	data frame containing the data (possibly a panel data frame created with pdata.frame).
shifterNames	a vector of strings containing the names of the independent variables that should be included as shifters only (not in quadratic or interaction terms).
linear	logical. Restrict the coefficients of all quadratic and interaction terms to be zero so that the estimated function is linear in the exogenous variables?

homWeights	numeric vector with named elements that are weighting factors for calculating an index that is used to normalize the variables for imposing homogeneity of degree zero in these variables (see details).
regScale	a scalar or vector with length equal to nrow(data). All regressors except for shifter variables that are logical or factors are divided by regScale (NOTE: quadratic and interaction terms are also divided by regScale and NOT divided by the square of regScale).
...	further arguments are passed to lm or plm.

Details

If argument homWeights is used to impose homogeneity of degree zero in some variables, the weighting factors in this vector must have names that are equal to the variable names in argument xNames. The order of the elements in homWeights is arbitrary and may or may not be equal to the order of the elements in xNames. Argument homWeights may contain less elements than xNames; in this case, homogeneity of degree zero is imposed only on variables with names in homWeights. Please note that the weighting factor of a variable (P_i) in homWeights ($w_i = \partial P/\partial P_i$) is not really its weight ($(\partial P/\partial P_i)(P_i/P)$), in particular, if the numerical values of the variables ($P_1, \ldots, P_n$) are rather different.

Value

a list of class quadFuncEst containing following objects:

est	the object returned by lm or plm.
nExog	length of argument xNames.
nShifter	length of argument shifterNames.
residuals	residuals.
fitted	fitted values.
coef	vector of all coefficients.
coefCov	covariance matrix of all coefficients.
r2	R^2 value.
r2bar	adjusted R^2 value.
nObs	number of observations.
model.matrix	the model matrix.
call	the matched call.
yName	argument yName.
xNames	argument xNames.
shifterNames	argument shifterNames.
homWeights	argument homWeights.
regScale	argument regScale.

Author(s)

Arne Henningsen

See Also

quadFuncCalc, quadFuncDeriv, translogEst and snqProfitEst.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- quadFuncEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

coef( estResult )
estResult$r2
```

---

residuals.translogEst *Residuals of a Translog function*

Description

Extract the residuals from the estimation of a Translog function.

Usage

```
## S3 method for class 'translogEst'
residuals( object, ... )
```

Arguments

object	an object of class translogEst.
...	currently not used.

Value

residuals.translogEst returns a vector containing the residuals of an estimated translog function.

Author(s)

Arne Henningsen

See Also

translogEst and residuals

summary.translogEst *Summarizing the Estimation of a Translog Function*

Description

summary.translogEst summarizes the estimation results of a Translog Function.

Usage

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

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

Arguments

object	an object of class translogEst.
x	an object of class summary.translogEst.
...	currently ignored.

Value

summary.translogEst returns a list of class summary.translogEst that is currently the provided object, but an element coefTable has been added and the class has been changed.

Author(s)

Arne Henningsen

See Also

translogEst.

translogCalc *Calculate dependent variable of a translog function*

Description

Calculate the dependent variable of a translog function.

Usage

```
translogCalc( xNames, data, coef, shifterNames = NULL,
   dataLogged = FALSE  )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe containing the data.
coef	vector containing all coefficients: if there are n exogenous variables in xNames and m shifter variables in shifterNames, the n+1 alpha coefficients must have names a_0, ..., a_n, the n*(n+1)/2 beta coefficients must have names b_1_1, ..., b_1_n, ..., b_n_n, and the m delta coefficients must have names d_1, ..., d_m (only the elements of the upper right triangle of the beta matrix are directly obtained from coef; the elements of the lower left triangle are obtained by assuming symmetry of the beta matrix).
shifterNames	a vector of strings containing the names of the independent variables that should be included as shifters only (not in quadratic or interaction terms).
dataLogged	logical. Are the values in data already logged?

Value

A vector containing the endogenous variable. If the inputs are provided as logarithmic values (argument dataLogged is TRUE), the endogenous variable is returned as logarithm; non-logarithmic values are returned otherwise.

Author(s)

Arne Henningsen

See Also

translogEst and translogDeriv.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a Translog production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

translogCalc( c( "qLabor", "land", "qVarInput", "time" ), germanFarms,
   coef( estResult ) )
#equal to estResult$fitted
```

translogCheckCurvature *Curvature of a Translog Function*

Description

Check curvature of a translog function.

Usage

```
translogCheckCurvature( xNames, data, coef, convexity = TRUE,
   quasi = FALSE, dataLogged = FALSE, ... )

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

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe containing the data.
coef	vector containing all coefficients.
convexity	logical. Check whether the function is (quasi)convex (default, TRUE) or (quasi)concave (FALSE).
quasi	logical. Check whether the function is quasiconvex/quasiconcave (TRUE) or convex/concave (default, FALSE).
dataLogged	logical. Are the values in data already logged?
x	an object returned by translogCheckCurvature.
...	arguments passed from translogCheckCurvature to semidefiniteness (if argument quasi is FALSE), quasiconvexity (if arguments convexity and quasi are both TRUE), or quasiconcavity (if argument convexity is FALSE and quasi is TRUE). Further arguments to print.translogCheckCurvature are currently ignored.

Value

translogCheckCurvature returns a list of class translogCheckCurvature containing following objects:

obs	a vector indicating whether the condition for the specified curvature is fulfilled at each observation.
convexity	argument convexity.
quasi	argument quasi.

Author(s)

Arne Henningsen

See Also

translogEst and translogCheckMono

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a translog production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# check whether the production function is quasiconcave
translogCheckCurvature( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), convexity = FALSE, quasi = TRUE )
```

---

translogCheckMono *Monotonicity of a Translog Function*

Description

Check monotonicity of a translog function.

Usage

```
translogCheckMono( xNames, data, coef, increasing = TRUE,
   strict = FALSE, dataLogged = FALSE,
   tol = 10 * .Machine$double.eps )

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

## S3 method for class 'translogCheckMono'
summary( object, ... )

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

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe containing the data.
coef	vector containing all coefficients.

increasing	single logical value or vector of logical values of the same length as argument xNames indicating whether it should be checked if the translog function is monotonically increasing (default, TRUE) or decreasing (FALSE) in the explanatory variables.
strict	logical. Check for strict (TRUE) or non-strict (default, FALSE) monotonicity?
dataLogged	logical. Are the values in data already logged?
tol	tolerance level for checking non-strict monotonicity: values between -tol and tol are considered to be zero (ignored if argument strict is TRUE).
x	an object returned by translogCheckMono or by summary.translogCheckMono.
object	an object returned by translogCheckMono.
...	currently not used.

Details

Function translogCheckMono internally calls function translogDeriv and then checks if the derivatives have the sign specified in argument increasing.

Function translogCheckMono does not have an argument shifterNames, because shifter variables do not affect the monotonicity conditions of the eplanatory variables defined in Argument xNames. Therefore, translogCheckMono automatically removes all coefficients of the shifter variables before it calls translogDeriv.

Value

translogCheckMono returns a list of class translogCheckMono containing following objects:

obs	a vector indicating whether monotonicity is fulfilled at each observation.
exog	data frame indicating whether monotonicity is fulfilled for each exogenous variable at each observation.
increasing	argument increasing.
strict	argument strict.

Author(s)

Arne Henningsen

See Also

translogEst, translogDeriv, and translogCheckCurvature

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
```

```
germanFarms$time <- c(1:20)

# estimate a translog production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# check whether the production function is monotonically increasing
# in all inputs
test <- translogCheckMono( xNames = c( "qLabor", "land", "qVarInput", "time" ),
   data = germanFarms, coef = coef( estResult ) )
test
summary( test )

# check whether the production function is monotonically decreasing
# in time and monotonically increasing in all other inputs
test <- translogCheckMono( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), increasing = c( TRUE, TRUE, TRUE, FALSE ) )
test
summary( test )
```

`translogCostEst` *Estimate a translog Cost Function*

Description

Estimate a translog cost function.

NOTE: this function is still under development and incomplete!

Usage

```
translogCostEst( cName, yName, pNames, data, fNames = NULL,
   shifterNames = NULL, dataLogged = FALSE, homPrice = TRUE, ... )
```

Arguments

`cName`	a string containing the name of the variable for total cost.
`yName`	a string containing the name of the variable for the total output quantity.
`pNames`	a vector of strings containing the names of the input prices.
`data`	data frame containing the data (possibly a panel data frame created with `pdata.frame`).
`fNames`	a vector of strings containing the names of fixed inputs.
`shifterNames`	a vector of strings containing the names of the independent variables that should be included as shifters only (not in quadratic or interaction terms).
`dataLogged`	logical. Are the values in `data` already logged?
`homPrice`	logical. Should homogeneity of degree one in prices be imposed?
`...`	further arguments are passed to `lm` or `plm`.

Value

a list of class `translogCostEst` containing following objects:

`est`	the object returned by `lm` or `plm`.
`nExog`	length of argument `xNames`.
`nShifter`	length of argument `shifterNames`.
`residuals`	residuals.
`fitted`	fitted values.
`coef`	vector of all coefficients.
`coefCov`	covariance matrix of all coefficients.
`r2`	R^2 value.
`r2bar`	adjusted R^2 value.
`nObs`	number of observations.
`model.matrix`	the model matrix.
`call`	the matched call.
`cName`	argument `cName`.
`yName`	argument `yName`.
`pNames`	argument `pNames`.
`fNames`	argument `fNames`.
`shifterNames`	argument `shifterNames`.
`dataLogged`	argument `dataLogged`.
`homPrice`	argument `homPrice`.

Author(s)

Arne Henningsen

See Also

`translogEst` and `quadFuncEst`.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# value of labor input
germanFarms$vLabor <- germanFarms$pLabor + germanFarms$qLabor
# total variable cost
germanFarms$cost <- germanFarms$vLabor + germanFarms$vVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a translog cost function
```

```
estResult <- translogCostEst( cName = "cost", yName = "qOutput",
   pNames = c( "pLabor", "pVarInput" ), fNames = "land",
   shifterNames = "time", data = germanFarms, homPrice = FALSE )

summary( estResult$est )
```

`translogDeriv` *Derivatives of a translog function*

Description

Calculate the derivatives of a translog function.

Usage

```
translogDeriv( xNames, data, coef, coefCov = NULL,
   yName = NULL, dataLogged = FALSE )
```

Arguments

`xNames`	a vector of strings containing the names of the independent variables.
`data`	dataframe containing the data.
`coef`	vector containing all coefficients.
`coefCov`	optional covariance matrix of the coefficients.
`yName`	an optional string containing the name of the dependent variable. If it is `NULL`, the dependent variable is calculated from the independent variables and the coefficients.
`dataLogged`	logical. Are the values in `data` already logged?

Value

a list of class `translogDeriv` containing following objects:

`deriv`	data frame containing the derivatives.
`variance`	data frame containing the variances of the derivatives (not implemented yet).
`stdDev`	data frame containing the standard deviations of the derivatives (not implemented yet).

Author(s)

Arne Henningsen

See Also

`translogEst`, `translogCalc` and `translogHessian`

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a translog production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# compute the marginal products of the inputs (with "fitted" Output)
margProducts <- translogDeriv( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), vcov( estResult ) )
margProducts$deriv

# compute the marginal products of the inputs (with observed Output)
margProductsObs <- translogDeriv( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), vcov( estResult ), "qOutput" )
margProductsObs$deriv
```

`translogEla` *Elasticities of a translog Function*

Description

Calculate the elasticities of a translog function.

Usage

```
translogEla( xNames, data, coef, coefCov = NULL,
   dataLogged = FALSE  )

## S3 method for class 'translogEst'
elas( object, data = NULL, dataLogged = NULL,
   ... )
```

Arguments

`xNames`	a vector of strings containing the names of the independent variables.
`data`	dataframe containing the data; if argument `data` of `elas.translogEst` is not specified, the data frame that was used for the estimation is used for calculating elasticities.
`coef`	vector containing all coefficients: if there are `n` exogenous variables in `xNames`, the `n+1` alpha coefficients must have names `a_0`, ..., `a_n` and the `n*(n+1)/2`

beta coefficients must have names b_1_1, ..., b_1_n, ..., b_n_n (only the elements of the upper right triangle of the beta matrix are directly obtained from coef; the elements of the lower left triangle are obtained by assuming symmetry of the beta matrix).

coefCov — optional covariance matrix of the coefficients: the row names and column names must be the same as the names of coef.

dataLogged — logical. Are the values in data already logged? If argument dataLogged of elas.translogEst is not specified, the same value as used in translogEst for creating object is used.

object — object of class translogEst (returned by translogEst).

... — currently ignored.

Details

Shifter variables do not need to be specified, because they have no effect on the elasticities. Hence, you can use this function to calculate elasticities even for translog functions that have been estimated with shifter variables.

Value

A data frame containing the elasticities, where each column corresponds to one of the independent variables. If argument coefCov is provided, it has the attributes variance and stdDev, which are two data frames containing the variances and the standard deviations, respectively, of the elasticities.

Author(s)

Arne Henningsen

See Also

translogEst and translogCalc

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# calculate production elasticities of all inputs
estEla <- translogEla( c( "qLabor", "land", "qVarInput", "time" ),
   data = germanFarms, coef = coef( estResult ),
   coefCov = vcov( estResult ) )
```

```
# all elasticities
estEla
# t-values of all elasticities
estEla / attributes( estEla )$stdDev
```

---

translogEst *Estimate a translog function*

Description

Estimate a translog function.

Usage

```
translogEst( yName, xNames, data, shifterNames = NULL,
   dataLogged = FALSE, ... )

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

Arguments

yName	a string containing the name of the dependent variable.
xNames	a vector of strings containing the names of the independent variables.
data	data frame containing the data (possibly a panel data frame created with pdata.frame).
shifterNames	a vector of strings containing the names of the independent variables that should be included as shifters only (not in quadratic or interaction terms).
dataLogged	logical. Are the values in data already logged? If FALSE, the logarithms of all variables (yName, xNames, shifterNames) are used except for shifter variables that are factors or logical variables.
x	An object of class translogEst.
...	further arguments of translogEst are passed to lm or plm; further arguments of print.translogEst are currently ignored.

Value

a list of class translogEst containing following objects:

est	the object returned by lm or plm.
nExog	length of argument xNames.
nShifter	length of argument shifterNames.
residuals	residuals.
fitted	fitted values.
coef	vector of all coefficients.

coefCov	covariance matrix of all coefficients.
r2	R^2 value.
r2bar	adjusted R^2 value.
nObs	number of observations.
model.matrix	the model matrix.
call	the matched call.
yName	argument yName.
xNames	argument xNames.
shifterNames	argument shifterNames.
dataLogged	argument dataLogged.

Author(s)

Arne Henningsen

See Also

translogCalc, translogDeriv and quadFuncEst.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

estResult
summary( estResult )
```

---

translogHessian *Hessian matrix of a translog function*

Description

Calculate the Hessian matrices of a translog function.

Usage

```
translogHessian( xNames, data, coef, yName = NULL,
dataLogged = FALSE, bordered = FALSE )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe containing the data.
coef	vector containing all coefficients.
yName	an optional string containing the name of the dependent variable. If it is NULL, the dependent variable is calculated from the independent variables and the coefficients.
dataLogged	logical. Are the values in data already logged?
bordered	logical. Should the *bordered* Hessians be returned?

Value

a list containing following the (bordered) Hessian matrices at each data point.

Author(s)

Arne Henningsen

See Also

translogEst, translogDeriv and translogCalc

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# estimate a quadratic production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# compute the Hessian matrices (with "fitted" output)
hessians <- translogHessian( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ) )
hessians[[ 1 ]]

# compute the Hessian matrices (with observed output)
hessiansObs <- translogHessian( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), "qOutput" )
hessiansObs[[ 1 ]]

# compute the bordered Hessian matrices
borderedHessians <- translogHessian( c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms, coef( estResult ), bordered = TRUE )
borderedHessians[[ 1 ]]
```

translogMonoRestr *Monotonicity Restrictions of a Translog Function*

Description

Create matrix to check or impose the monotonicity restrictions of a translog function.

Usage

```
translogMonoRestr( xNames, data,
   dataLogged = FALSE, box = FALSE )
```

Arguments

xNames	a vector of strings containing the names of the independent variables.
data	dataframe containing the data.
dataLogged	logical. Are the values in data already logged?
box	logical. Should monotonicity be imposed within an n-dimensional box that includes all points in data? If FALSE, monotonicity is imposed (only) within an n-dimensional polygon that includes all points in data. (n is the number of independent varables.)

Value

translogMonoRestr returns a matrix of dimension $(n \cdot N) \times c$, where n is the number of independent varables, N is the number of data points at which monotonicity should be imposed (if argument box is FALSE, N is the number of rows in data; if argument box is TRUE, $N = 2^n$), and $c = 1 + n(n+3)/2$ is the number of (linearly independent) coefficients. Multiplying a row of this matrix (e.g.\ the kth row of M) by the vector of coefficients (β) results in the derivative of the dependent variable (y) with respect to one independent variable (e.g.\ x_i) at one data point (e.g.\ j):

$$M[k,] \cdot \beta = \frac{\partial \ln y}{\partial \ln x_i}$$

, evaluated at $x_{1j}, \ldots, x_{nj}$, where $k = (i-1)N + j$. Hence, the observations run faster than the independent variables.

Author(s)

Arne Henningsen

See Also

translogEst, translogDeriv, and translogCheckMono

Examples

```
data( germanFarms )
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput

# matrix to check or impose monotonicity at all observations
monoRestrObs <- translogMonoRestr( c( "qLabor", "land", "qVarInput" ),
   germanFarms )

# matrix to check or impose monotonicity within a box that includes all
# observations
monoRestrBox <- translogMonoRestr( c( "qLabor", "land", "qVarInput" ),
   germanFarms, box = TRUE )
```

---

translogProdFuncMargCost

Marginal Costs of Translog Production Function

Description

Calculate the marginal costs of the output from a translog production function.

Usage

```
translogProdFuncMargCost( yName, xNames, wNames, data, coef,
   dataLogged = FALSE )
```

Arguments

yName	a single character string containing the name of the output quantity.
xNames	a vector of strings containing the names of the input quantities.
wNames	a vector of strings containing the names of the input prices.
data	dataframe containing the data.
coef	vector containing all coefficients: if there are n inputs in xNames, the n+1 alpha coefficients must have names a_0, ..., a_n and the n*(n+1)/2 beta coefficients must have names b_1_1, ..., b_1_n, ..., b_n_n (only the elements of the upper right triangle of the beta matrix are directly obtained from coef; the elements of the lower left triangle are obtained by assuming symmetry of the beta matrix).
dataLogged	logical. Are the values in data already logged?

Value

A vector containing the marginal costs of producing the output.

Author(s)

Arne Henningsen and Geraldine Henningsen

See Also

translogEst, translogCalc, translogDeriv, translogEla and translogCostEst.

Examples

```
data( germanFarms )
# output quantity:
germanFarms$qOutput <- germanFarms$vOutput / germanFarms$pOutput
# quantity of variable inputs
germanFarms$qVarInput <- germanFarms$vVarInput / germanFarms$pVarInput
# a time trend to account for technical progress:
germanFarms$time <- c(1:20)

# generate (artificial) prices
germanFarms$pLand <- 200 + 15 * germanFarms$time
germanFarms$pTime <- 1

# estimate a single-output translog production function
estResult <- translogEst( "qOutput", c( "qLabor", "land", "qVarInput", "time" ),
   germanFarms )

# compute the marginal costs of producing the output
margCost <- translogProdFuncMargCost( yName = "qOutput",
   xNames = c( "qLabor", "land", "qVarInput", "time" ),
   wNames = c( "pLabor", "pLand", "pVarInput", "pTime" ),
   data = germanFarms, coef = coef( estResult ) )
```

---

utility *Fossil Fuel Fired Steam Electric Power Plants*

Description

The data frame utility is a panel data set of 72 investor-owned fossil fuel fired steam electric power plants in the United States over eleven years (1986-1996). This data set has 791 observations and, thus, is almost balanced (only one firm-year observation is missing). This data set is used as an example in Kumbhakar, Wang, and Horncastle (2015).

Usage

```
data(utility)
```

Format

This data frame contains the following columns/variables:

firm firm number (1-72).

year last two digits of the year (86-96).

y net steam electric power generation in MWh.

regu a dummy variable indicating whether the firm is regulated.

k capital input quantity (estimate of capital cost).

labor quantity index of labor and maintenance input (costs of labor and maintenance divided by `wl`).

fuel quantity index of fuel input. (costs of fuel divided by `wf`).

wl cost-share weighted price of labor and maintenance.

wf average price of fuel (coal, oil and gas) in USD per BTU.

wk price of the capital input.

Details

This data set is a revised version of the data set that is used in Kumbhakar and Wang (2006). A detailed description of the data set and the variables can be found in Section 4 of Kumbhakar and Wang (2006).

Source

Companion website to Kumbhakar, Wang, and Horncastle (2015): https://sites.google.com/site/sfbook2014/.

References

Kumbhakar, Subal C. and Hung-Jen Wang (2006): Estimation of Technical and Allocative Inefficiency: A Primal System Approach, Journal of Econometrics 134(2), p. 419-440. (doi:10.1016/j.jeconom.2005.07.001)

Kumbhakar, Subal C., Hung-Jen Wang, and Alan P. Horncastle (2015): A Practitioner's Guide to Stochastic Frontier Analysis Using Stata. Cambridge University Press. (doi:10.1017/CBO9781139342070)

Index