Package ‘micEconSNQP’

February 11, 2020

Version 0.6-8

Date 2020-02-10

Title Symmetric Normalized Quadratic Profit Function

Author Arne Henningsen

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

Depends R (>= 2.4.0)

Suggests micEcon (>= 0.6-1)

Imports miscTools (>= 0.6-1), systemfit (>= 1.0-0), MASS

Description Tools for econometric production analysis
with the Symmetric Normalized Quadratic (SNQ) profit function,
e.g. estimation, imposing convexity in prices,
and calculating elasticities and shadow prices.

URL http://www.micEcon.org

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-11 17:10:02 UTC

R topics documented:

predict.snqProfitEst *Predictions from an SNQ profit function*

Description

Returns the predicted values, their standard errors and the confidence limits of prediction for an Symmetric Normalized Quadratic (SNQ) profit function.

Usage

```
## S3 method for class 'snqProfitEst'
predict( object, newdata = object$data,
   se.fit = FALSE, se.pred = FALSE, interval = "none", level = 0.95,
   useDfSys = TRUE, ... )

## S3 method for class 'snqProfitImposeConvexity'
predict( object, newdata = object$data,
   se.fit = FALSE, se.pred = FALSE, interval = "none", level = 0.95,
   useDfSys = TRUE, ... )
```

Arguments

`object`	an object of type `snqProfitEst` or `snqProfitImposeConvexity`.
`newdata`	data frame in which to predict.
`se.fit`	logical. Return the standard error of the fitted values?
`se.pred`	logical. Return the standard error of prediction?
`interval`	Type of interval calculation ("none", "confidence" or "prediction").
`level`	confidence level.
`useDfSys`	logical. Use the degrees of freedom of the whole system (and not the degrees of freedom of the single equation) to calculate the confidence intervals.
`...`	currently not used.

Details

The variance of the fitted values (used to calculate the standard errors of the fitted values and the "confidence interval") is calculated by $Var[E[y^0] - \hat{y}^0] = x^0 \; Var[b] \; x^{0\prime}$
The variances of the predicted values (used to calculate the standard errors of the predicted values and the "prediction intervals") is calculated by $Var[y^0 - \hat{y}^0] = \hat{\sigma}^2 + x^0 \; Var[b] \; x^{0\prime}$

Value

`predict.snqProfitEst` and `predict.snqProfitImposeConvexity` return a dataframe that contains the predicted profit and for each netput the predicted quantities (e.g. "quant1") and if requested the standard errors of the fitted values (e.g. "quant1.se.fit"), the standard errors of the prediction (e.g. "quant1.se.pred"), and the lower (e.g. "quant1.lwr") and upper (e.g. "quant1.upr") limits of the confidence or prediction interval(s).

Author(s)

Arne Henningsen

References

Diewert, W.E. and T.J. Wales (1987) Flexible functional forms and global curvature conditions. *Econometrica*, 55, p. 43-68.

Diewert, W.E. and T.J. Wales (1992) Quadratic Spline Models for Producer's Supply and Demand Functions. *International Economic Review*, 33, p. 705-722.

Greene, W. H. (2003) *Econometric Analysis, Fifth Edition*, Macmillan.

Gujarati, D. N. (1995) *Basic Econometrics, Third Edition*, McGraw-Hill.

Kmenta, J. (1997) *Elements of Econometrics, Second Edition*, University of Michigan Publishing.

Kohli, U.R. (1993) A symmetric normalized quadratic GNP function and the US demand for imports and supply of exports. *International Economic Review*, 34, p. 243-255.

See Also

snqProfitEst, snqProfitCalc and predict

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time      <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
estResult <- snqProfitEst( priceNames, quantNames, c("land","time"), data=germanFarms )
predict( estResult )
predict( estResult, se.fit = TRUE, se.pred = TRUE, interval = "confidence" )
```

---

print.snqProfitEst *Print output of estimated SNQ profit function*

Description

This function prints a summary estimation results of a symmetric normalized quadraric (SNQ) profit function.

Usage

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

Arguments

x	an object of class snqProfitEst.
...	arguments passed to print.

Author(s)

Arne Henningsen

See Also

snqProfitEst

Examples

```
## Not run: library( systemfit )
   data( germanFarms, package = "micEcon" )
   germanFarms$qOutput   <- germanFarms$vOutput / germanFarms$pOutput
   germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
   germanFarms$qLabor    <- -germanFarms$qLabor
   priceNames <- c( "pOutput", "pVarInput", "pLabor" )
   quantNames <- c( "qOutput", "qVarInput", "qLabor" )

   estResult <- snqProfitEst( priceNames, quantNames, "land", data = germanFarms )
   print( estResult )
```

---

residuals.snqProfitEst *Residuals of an SNQ profit function*

Description

Extract the residuals from the estimation of a Symmetric Normalized Quadratic (SNQ) profit function.

Usage

```
## S3 method for class 'snqProfitEst'
residuals( object, scaled = TRUE, ... )

## S3 method for class 'snqProfitImposeConvexity'
residuals( object, scaled = TRUE, ... )
```

Arguments

object	an object of type snqProfitEst or snqProfitImposeconvexity.
scaled	logical. Return scaled quantities?
...	currently not used.

Value

`residuals.snqProfitEst` and `residuals.snqProfitEst` return a dataframe that contains the residuals for each netput and the profit.

Author(s)

Arne Henningsen

See Also

`snqProfitEst`, `snqProfitImposeConvexity` and `residuals`

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time      <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
estResult <- snqProfitEst( priceNames, quantNames, c("land","time"), data=germanFarms )
residuals( estResult )
residuals( estResult, scaled = FALSE )
```

`snqProfitCalc` *Calculations with the SNQ Profit function*

Description

Calculation of netput quantities and profit with the Symmetric Normalized Quadratic (SNQ) Profit function.

Usage

```
snqProfitCalc( priceNames, fixNames, data, weights,
   scalingFactors = rep( 1, length( weights ) ), coef,
   quantNames = NULL, form = 0 )
```

Arguments

`priceNames`	a vector of strings containing the names of netput prices.
`fixNames`	an optional vector of strings containing the names of the quantities of (quasi-)fix inputs.
`data`	a data frame containing the data.
`weights`	vector of weights of the prices for normalization.
`quantNames`	optional vector of strings containing the names of netput quantities.

scalingFactors	factors to scale prices (and quantities).
coef	a list containing the coefficients alpha, beta, delta and gamma.
form	the functional form to be estimated (see snqProfitEst).

Value

a data frame: the first n columns are the netput quantities, the last column is the profit.

Author(s)

Arne Henningsen

References

Diewert, W.E. and T.J. Wales (1987) Flexible functional forms and global curvature conditions. *Econometrica*, 55, p. 43-68.

Diewert, W.E. and T.J. Wales (1992) Quadratic Spline Models for Producer's Supply and Demand Functions. *International Economic Review*, 33, p. 705-722.

Kohli, U.R. (1993) A symmetric normalized quadratic GNP function and the US demand for imports and supply of exports. *International Economic Review*, 34, p. 243-255.

See Also

snqProfitEst and snqProfitWeights.

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time      <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
fixNames <- c( "land", "time" )

estResult <- snqProfitEst( priceNames, quantNames, fixNames, data = germanFarms )
snqProfitCalc( priceNames, fixNames, estResult$data, estResult$weights,
   estResult$scalingFactors, estResult$coef )
```

---

snqProfitEla   *Price Elasticities of SNQ Profit function*

Description

Calculates the Price Elasticities of a Symmetric Normalized Quadratic (SNQ) profit function.

Usage

```
snqProfitEla(  beta, prices, quant, weights,
  scalingFactors = rep( 1, length( weights ) ),
  coefVcov = NULL, df = NULL )
```

Arguments

`beta`	matrix of estimated β coefficients.
`prices`	vector of netput prices at which the elasticities should be calculated.
`quant`	vector of netput quantities at which the elasticities should be calculated.
`weights`	vector of weights of prices used for normalization.
`scalingFactors`	factors to scale prices (and quantities).
`coefVcov`	variance covariance matrix of the coefficients (optional).
`df`	degrees of freedom to calculate P-values of the elasticities (optional).

Value

a list of class `snqProfitEla` containing following elements:

`ela`	matrix of the price elasticities.
`vcov`	variance covariance matrix of the price elasticities.
`stEr`	standard errors of the price elasticities.
`tval`	t-values of the price elasticities.
`pval`	P-values of the price elasticities.

Note

A price elasticity is defined as

$$E_{ij} = \frac{\frac{\partial q_i}{q_i}}{\frac{\partial p_j}{p_j}} = \frac{\partial q_i}{\partial p_j} \cdot \frac{p_j}{q_i}$$

Thus, e.g. $E_{ij} = 0.5$ means that if the price of netput j (p_j) increases by 1%, the quantity of netput i (q_i) will increase by 0.5%.

Author(s)

Arne Henningsen

See Also

`snqProfitEst`.

Examples

```
# just a stupid simple example
snqProfitEla( matrix(101:109,3,3), c(1,1,1), c(1,-1,-1), c(0.4,0.3,0.3) )

# now with real data
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput   / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time      <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )

estResult <- snqProfitEst( priceNames, quantNames, c("land","time"), data=germanFarms )

estResult$ela  # price elasticities at mean prices and mean quantities

# price elasticities at the last observation (1994/95)
snqProfitEla( estResult$coef$beta, estResult$data[ 20, priceNames ],
   estResult$data[ 20, quantNames ], estResult$weights,
   estResult$scalingFactors )
```

`snqProfitEst` *Estimation of a SNQ Profit function*

Description

Estimation of a Symmetric Normalized Quadratic (SNQ) Profit function.

Usage

```
snqProfitEst( priceNames, quantNames, fixNames = NULL, instNames = NULL,
   data, form = 0, base = 1, scalingFactors = NULL,
  weights = snqProfitWeights( priceNames, quantNames, data, "DW92", base = base ),
   method = ifelse( is.null( instNames ), "SUR", "3SLS" ), ... )
```

Arguments

`priceNames`	a vector of strings containing the names of netput prices.
`quantNames`	a vector of strings containing the names of netput quantities (inputs must be negative).
`fixNames`	an optional vector of strings containing the names of the quantities of (quasi-)fixed inputs.
`instNames`	an optional vector of strings containing the names of instrumental variables (for 3SLS estimation).
`data`	a data frame containing the data.
`form`	the functional form to be estimated (see details).

`base`	the base period(s) for scaling prices (see details). If argument `weights` is not specified, argument `base` is also used to obtain the weights for normalizing prices (see `snqProfitWeights`).
`scalingFactors`	a vector of factors to scale prices (see details).
`weights`	a vector of weights for normalizing prices.
`method`	the estimation method (passed to `systemfit`).
`...`	arguments passed to `systemfit`

Details

The Symmetric Normalized Quadratic (SNQ) profit function is defined as follows (this functional form is used if argument `form` equals 0):

$$\pi(p,z) = \sum_{i=1}^{n} \alpha_i p_i + \frac{1}{2} w^{-1} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} p_i p_j + \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{ij} p_i z_j + \frac{1}{2} w \sum_{i=1}^{m} \sum_{j=1}^{m} \gamma_{ij} z_i z_j$$

with π = profit, p_i = netput prices, z_i = quantities of fixed inputs, $w = \sum_{i=1}^{n} \theta_i p_i$ = price index for normalization, θ_i = weights of prices for normalization, and α_i, β_{ij}, δ_{ij} and γ_{ij} = coefficients to be estimated.
The netput equations (output supply in input demand) can be obtained by Hotelling's Lemma ($q_i = \partial\pi / \partial p_i$):

$$x_i = \alpha_i + w^{-1} \sum_{j=1}^{n} \beta_{ij} p_j - \frac{1}{2} \theta_i w^{-2} \sum_{j=1}^{n} \sum_{k=1}^{n} \beta_{jk} p_j p_k + \sum_{j=1}^{m} \delta_{ij} z_j + \frac{1}{2} \theta_i \sum_{j=1}^{m} \sum_{k=1}^{m} \gamma_{jk} z_j z_k$$

In my experience the fit of the model is sometimes not very good, because the effect of the fixed inputs is forced to be proportional to the weights for price normalization θ_i. In this cases I use following extended SNQ profit function (this functional form is used if argument `form` equals 1):

$$\pi(p,z) = \sum_{i=1}^{n} \alpha_i p_i + \frac{1}{2} w^{-1} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} p_i p_j + \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{ij} p_i z_j + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} \gamma_{ijk} p_i z_j z_k$$

The netput equations are now:

$$x_i = \alpha_i + w^{-1} \sum_{j=1}^{n} \beta_{ij} p_j - \frac{1}{2} \theta_i w^{-2} \sum_{j=1}^{n} \sum_{k=1}^{n} \beta_{jk} p_j p_k + \sum_{j=1}^{m} \delta_{ij} z_j + \frac{1}{2} \sum_{j=1}^{m} \sum_{k=1}^{m} \gamma_{ijk} z_j z_k$$

Argument `scalingFactors` can be used to scale prices, e.g. for improving the numerical stability of the estimation (e.g. if prices are very large or very small numbers) or for assessing the robustness of the results when changing the units of measurement. The prices are multiplied by the scaling factors, while the quantities are divided my the scaling factors so that the monetary values of the inputs and outputs and thus, the profit, remains unchanged. If argument `scalingFactors` is `NULL`, argument `base` is used to automatically obtain scaling factors so that the scaled prices are unity in the base period or - if there is more than one base period - that the means of the scaled prices over the base periods are unity. Argument `base` can be either
(a) a single number: the row number of the base prices,
(b) a vector indicating several observations: The means of these observations are used as base

prices,
(c) a logical vector with length equal to the number of rows of the data set that is specified by argument `data`: The means of the observations indicated as 'TRUE' are used as base prices, or
(d) `NULL`: prices are not scaled. If argument `base` is `NULL`, argument `weights` must be specified, because the weights cannot be calculated if the base period is not specified. An alternative way to use unscaled prices is to set argument `scalingFactors` equal to a vector of ones (see examples below).
If the scaling factors are explicitly specified by argument `scalingFactors`, argument `base` is not used for obtaining scaling factors (but it is used for obtaining weights if argument `weights` is not specified).

Value

a list of class `snqProfitEst` containing following objects:

`coef`	a list containing the vectors/matrix of the estimated coefficients: * alpha = α_i. * beta = β_{ij}. * delta = δ_{ij} (only if quasi-fix inputs are present). * gamma = γ_{ij} (only if quasi-fix inputs are present). * allCoef = vector of all coefficients. * allCoefCov = covariance matrix of all coefficients. * stats = all coefficients with standard errors, t-values and p-values. * liCoef = vector of linear independent coefficients. * liCoefCov = covariance matrix of linear independent coefficients.
`ela`	a list of class `snqProfitEla` that contains (amongst others) the price elasticities at mean prices and mean quantities (see `snqProfitEla`).
`fixEla`	matrix of the fixed factor elasticities at mean prices and mean quantities.
`hessian`	hessian matrix of the profit function with respect to prices evaluated at mean prices.
`convexity`	logical. Convexity of the profit function.
`r2`	R^2-values of all netput equations.
`est`	estimation results returned by `systemfit`.
`weights`	the weights of prices used for normalization.
`normPrice`	vector used for normalization of prices.
`data`	data frame of originally supplied data.
`fitted`	data frame that contains the fitted netput quantities and the fitted profit.
`pMeans`	means of the scaled netput prices.
`qMeans`	means of the scaled netput quantities.
`fMeans`	means of the (quasi-)fixed input quantities.
`priceNames`	a vector of strings containing the names of netput prices.
`quantNames`	a vector of strings containing the names of netput quantities (inputs must be negative).
`fixNames`	an optional vector of strings containing the names of the quantities of (quasi-)fixed inputs.

instNames	an optional vector of strings containing the names of instrumental variables (for 3SLS estimation).
form	the functional form (see details).
base	the base period(s) for scaling prices (see details).
weights	vector of weights of the prices for normalization.
scalingFactors	factors to scale prices (and quantities).
method	the estimation method.

Author(s)

Arne Henningsen

References

Diewert, W.E. and T.J. Wales (1987) Flexible functional forms and global curvature conditions. *Econometrica*, 55, p. 43-68.

Diewert, W.E. and T.J. Wales (1992) Quadratic Spline Models for Producer's Supply and Demand Functions. *International Economic Review*, 33, p. 705-722.

Kohli, U.R. (1993) A symmetric normalized quadratic GNP function and the US demand for imports and supply of exports. *International Economic Review*, 34, p. 243-255.

See Also

snqProfitEla and snqProfitWeights.

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )

estResult <- snqProfitEst( priceNames, quantNames, "land", data = germanFarms )
estResult$ela   # Oh, that looks bad!

# it it reasonable to account for technological progress
germanFarms$time <- c( 0:19 )
estResult2 <- snqProfitEst( priceNames, quantNames, c("land","time"),
   data = germanFarms )
estResult2$ela   # Ah, that looks better!

# estimation with unscaled prices
estResultNoScale <- snqProfitEst( priceNames, quantNames, c("land","time"),
 data = germanFarms, scalingFactors = rep( 1, 3 ) )
print( estResultNoScale )

# alternative way of estimation with unscaled prices
```

```
estResultNoScale2 <- snqProfitEst( priceNames, quantNames, c("land","time"),
   data = germanFarms, base = NULL,
   weights = snqProfitWeights( priceNames, quantNames, germanFarms ) )
all.equal( estResultNoScale[-20], estResultNoScale2[] )

# please note that the SNQ Profit function is not invariant
# to units of measurement so that different scaling factors
# result in different estimates of elasticities:
all.equal( estResult2$ela, estResultNoScale$ela )
```

`snqProfitFixEla` *Fixed Factor Elasticities of SNQ Profit function*

Description

Calculates the Fixed Factor Elasticities of a Symmetric Normalized Quadratic (SNQ) profit function.

Usage

```
snqProfitFixEla( delta, gamma, quant, fix, weights,
  scalingFactors = rep( 1, length( weights ) ) )
```

Arguments

`delta`	matrix of estimated δ coefficients.
`gamma`	matrix of estimated γ coefficients.
`quant`	vector of netput quantities at which the elasticities should be calculated.
`fix`	vector of quantities of fixed factors at which the elasticities should be calculated.
`weights`	vector of weights of prices used for normalization.
`scalingFactors`	factors to scale prices (and quantities).

Note

A fixed factor elasticity is defined as

$$E_{ij} = \frac{\frac{\partial q_i}{q_i}}{\frac{\partial z_j}{z_j}} = \frac{\partial q_i}{\partial z_j} \cdot \frac{z_j}{q_i}$$

Thus, e.g. $E_{ij} = 0.5$ means that if the quantity of fixed factor j (z_j) increases by 1%, the quantity of netput i (q_i) will increase by 0.5%.

Author(s)

Arne Henningsen

See Also

snqProfitEst and snqProfitEla.

Examples

```
# just a stupid simple example
snqProfitFixEla( matrix(1:6/6,3,2 ), matrix(4:1/4,2 ), c(1,1,1), c(1,1),
   c(0.4,0.3,0.3) )

# now with real data
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput   / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time      <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
fixNames <- c( "land", "time" )

estResult <- snqProfitEst( priceNames, quantNames, fixNames, data=germanFarms )

estResult$fixEla  # price elasticities at mean quantities of netputs
                  # and fixed factors

# fixed factor elasticities at the last observation (1994/95)
snqProfitFixEla( estResult$coef$delta, estResult$coef$gamma,
   estResult$data[ 20, quantNames ], estResult$data[ 20, fixNames ],
   estResult$weights, estResult$scalingFactors )
```

---

snqProfitHessian *SNQ Profit function: Hessian matrix*

Description

Returns the Hessian (substitution) matrix of a Symmetric Normalized Quadratic (SNQ) Profit Function.

Usage

```
snqProfitHessian( beta, prices, weights,
   scalingFactors = rep( 1, length( weights ) ) )
```

Arguments

beta	matrix of the $beta$ coefficients.
prices	vector of netput prices at which the Hessian should be calculated.
weights	vector of weights of prices for normalization.
scalingFactors	factors to scale prices (and quantities).

Author(s)

Arne Henningsen

See Also

snqProfitEst, snqProfitEla and snqProfitHessianDeriv.

Examples

```
# just a stupid simple example
snqProfitHessian( matrix(101:109,3,3), c(1,1,1), c(0.4,0.3,0.3) )

# now with real data
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput   / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )

estResult <- snqProfitEst( priceNames, quantNames, c("land","time"), data=germanFarms )

estResult$hessian  # the Hessian at mean prices and mean quantities

# Hessian at the last observation (1994/95)
snqProfitHessian( estResult$coef$beta, estResult$data[ 20, priceNames ],
   estResult$weights, estResult$scalingFactors )
```

---

snqProfitHessianDeriv *SNQ Profit function: Derivatives of the Hessian*

Description

Returns the matrix of derivatives of the vector of linear independent values of the Hessian with respect to the vector of the linear independent coefficients.

Usage

```
snqProfitHessianDeriv( prices, weights, nFix = 0, form = 0 )
```

Arguments

prices	vector of netput prices at which the derivatives should be calculated.
weights	vector of weights for normalizing prices.
nFix	number of (quasi-)fix inputs.
form	the functional form to be estimated (see snqProfitEst).

Author(s)

Arne Henningsen

See Also

snqProfitHessian.

Examples

```
# just a stupid simple example
snqProfitHessianDeriv( c(1,2,3),c(0.4,0.3,0.3) )

# now with real data
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput   / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )

estResult <- snqProfitEst( priceNames, quantNames, c("land","time"), data=germanFarms )

snqProfitHessianDeriv( estResult$pMean, estResult$weights, 2 )
```

---

snqProfitImposeConvexity
Imposing Convexity on a SNQ Profit function

Description

Imposing Convexity on a Symmetric Normalized Quadratic (SNQ) Profit function.

Usage

```
snqProfitImposeConvexity( estResult, rankReduction = 0,
   start = 10, optimMethod = "BFGS", control = list( maxit=5000 ),
   stErMethod = "none", nRep = 1000, verbose = 0 )
```

Arguments

estResult	object returned by snqProfitEst.
rankReduction	an integer specifying the reduction of the rank of the β matrix.
start	starting values of the triangular Cholesky matrix.
optimMethod	method to be used by optim.
control	list of control parameters passed to optim.

stErMethod	method to compute standard errors, either 'none', 'resample', 'jackknife' or 'coefSim' (see details).
nRep	number of replications to compute the standard errors if stErMethod is either 'resample' or 'coefSim'.
verbose	an integer idicating the verbose level.

Details

The procedure proposed by Koebel, Falk and Laisney (2000, 2003) is applied to impose convexity in prices on an estimated symmetric normalized quadratic (SNQ) profit function.
The standard errors of the restricted coefficients can be either calculated by bootstrap resampling ('resampling'), jackknife ('jacknife') or by simulating the distribution of the unrestricted coefficients using its variance covariance matrix ('coefSim').

Value

a list of class snqProfitImposeConvexity containing the same objects as an object of class snqProfitEst and additionally the objects:

mindist	object returned by optim.
sim	results of the simulation to obtain the standard errors of the estimated coefficients.

Author(s)

Arne Henningsen

References

Koebel, B., M. Falk and F. Laisney (2000), Imposing and Testing Curvature Conditions on a Box-Cox Cost Function. Discussion Paper No. 00-70, ZEW, Mannheim, https://madoc.bib.uni-mannheim.de/515/1/dp0070.pdf.

Koebel, B., M. Falk and F. Laisney (2003), Imposing and Testing Curvature Conditions on a Box-Cox Cost Function. *Journal of Business and Economic Statistics*, 21, p. 319-335.

See Also

snqProfitEst.

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
estResult <- snqProfitEst( priceNames, quantNames, "land", data = germanFarms )
estResult  # Note: it is NOT convex in netput prices
estResultConvex <- snqProfitImposeConvexity( estResult )
```

```
estResultConvex  # now it is convex
```

`snqProfitShadowPrices` *Shadow Prices of a SNQ Profit function*

Description

Calculates the shadow prices of a Symmetric Normalized Quadratic (SNQ) profit function.

Usage

```
snqProfitShadowPrices( priceNames, fixNames, estResult = NULL,
data = estResult$data, weights = estResult$weights,
scalingFactors = estResult$scalingFactors,
coef = estResult$coef, form = estResult$form )
```

Arguments

`priceNames`	a vector of strings containing the names of netput prices.
`fixNames`	an optional vector of strings containing the names of the quantities of (quasi-)fix inputs.
`estResult`	object returned by `snqProfitEst`.
`data`	a data frame containing the data.
`weights`	vector of weights of prices used for normalization.
`scalingFactors`	factors to scale prices (see details).
`coef`	a list containing the coefficients (at least delta and gamma).
`form`	the functional form to be estimated (see details).

Author(s)

Arne Henningsen

See Also

`snqProfitEst`, `snqProfitCalc` and `snqProfitEla`.

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput   / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
germanFarms$time      <- c( 0:19 )
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
fixNames <- c( "land", "time" )
```

```
estResult <- snqProfitEst( priceNames, quantNames, fixNames, data = germanFarms )

snqProfitShadowPrices( priceNames, fixNames, estResult )
```

snqProfitWeights *SNQ Profit function: Weights of prices for normalization*

Description

Returns a vector of weights to normalize prices on a Symmetric Normalized Quadratic (SNQ) Profit function.

Usage

```
snqProfitWeights( priceNames, quantNames, data, method = "DW92", base = 1 )
```

Arguments

priceNames	a vector of strings containing the names of netput prices.
quantNames	a vector of strings containing the names of netput quantities.
data	a data frame containing the data.
method	the method to determine the weights (see details).
base	the base period(s) for scaling prices (see details).

Details

If argument method is 'DW92' the method of Diewert and Wales (1992) is applied. They predetermine the weights by

$$\theta_i = \frac{|\overline{x}_i|\, p_i^0}{\sum_{i=1}^{n} |\overline{x}_i|\, p_i^0}$$

Defining the scaled netput quantities as $\widetilde{x}_i^t = x_i^t \cdot p_i^0$ we get following formula:

$$\theta_i = \frac{\left|\overline{\widetilde{x}}_i\right|}{\sum_{i=1}^{n} \left|\overline{\widetilde{x}}_i\right|}$$

The prices are scaled that they are unity in the base period or - if there is more than one base period - that the means of the prices over the base periods are unity. The argument base can be either
(a) a single number: the row number of the base prices,
(b) a vector indicating several observations: The means of these observations are used as base prices,
(c) a logical vector with the same length as the data: The means of the observations indicated as 'TRUE' are used as base prices, or (d) NULL: prices are not scaled.

Author(s)

Arne Henningsen

See Also

snqProfitEst.

Examples

```
data( germanFarms, package = "micEcon" )
germanFarms$qOutput   <- germanFarms$vOutput   / germanFarms$pOutput
germanFarms$qVarInput <- -germanFarms$vVarInput / germanFarms$pVarInput
germanFarms$qLabor    <- -germanFarms$qLabor
priceNames <- c( "pOutput", "pVarInput", "pLabor" )
quantNames <- c( "qOutput", "qVarInput", "qLabor" )
snqProfitWeights( priceNames, quantNames, germanFarms )
```

Index