Package ‘systemfit’

March 22, 2023

Version 1.1-30

Date 2023-03-22

Title Estimating Systems of Simultaneous Equations

Author Arne Henningsen and Jeff D. Hamann

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

Depends R (>= 3.2.0), Matrix, car (>= 2.0-0), lmtest

Suggests knitr, plm (>= 1.0-1), sem (>= 2.0-0)

Imports stats (>= 2.14.0), sandwich (>= 2.2-9), MASS, methods

Description Econometric estimation of simultaneous systems of linear and nonlinear equations using Ordinary Least Squares (OLS), Weighted Least Squares (WLS), Seemingly Unrelated Regressions (SUR), Two-Stage Least Squares (2SLS), Weighted Two-Stage Least Squares (W2SLS), and Three-Stage Least Squares (3SLS) as suggested, e.g., by Zellner (1962) <doi:10.2307/2281644>, Zellner and Theil (1962) <doi:10.2307/1911287>, and Schmidt (1990) <doi:10.1016/0304-4076(90)90127-F>.

URL https://r-forge.r-project.org/projects/systemfit/

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2023-03-22 17:00:02 UTC

R topics documented:

`bread.systemfit` *Bread for Sandwiches*

Description

Extract the estimator for the bread of sandwhiches (see `bread`).

Usage

```
## S3 method for class 'systemfit'
bread( x, ... )
```

Arguments

`x`	an object of class `systemfit`.
`...`	further arguments (currently ignored).

Value

Quadratic symmetric matrix, which is an estimator for the expectation of the negative derivative of the estimating function (see `estfun.systemfit`).

Warnings

The **sandwich** package must be loaded before this method can be used.

This method might not be suitable for specific formulas for 3SLS estimations in case of unbalanced systems or different instruments for different equations.

Author(s)

Arne Henningsen

See Also

`bread`, `systemfit`.

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )
inst <- ~ income + farmPrice + trend

## OLS estimation
fitols <- systemfit( system, "OLS", data = Kmenta )

## obtain the bread
library( "sandwich" )
bread( fitols )

## this is only true for OLS models
all.equal( bread( fitols ),
   solve( crossprod( model.matrix( fitols ) ) / 40 ) )


## 2SLS estimation
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )

## obtain the bread
bread( fit2sls )

## this is only true for 2SLS models
all.equal( bread( fit2sls ),
   solve( crossprod( model.matrix( fit2sls, which = "xHat" ) ) / 40 ) )


## iterated SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta, maxit = 100 )

## obtain the bread
bread( fitsur )

## this should be true for SUR and WLS models
```

```
all.equal( bread( fitsur ),
   solve( t( model.matrix( fitsur ) ) %*%
      (  ( solve( fitsur$residCovEst ) %x% diag( nrow( Kmenta ) ) ) %*%
      model.matrix( fitsur ) ) / 40 ), check.attributes = FALSE )


## 3SLS estimation
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta )

## obtain the bread
bread( fit3sls )

## this should be true for 3SLS and W2SLS models
all.equal( bread( fit3sls ),
   solve( t( model.matrix( fit3sls, which = "xHat" ) ) %*%
      (  ( solve( fit3sls$residCovEst ) %x% diag( nrow( Kmenta ) ) ) %*%
      model.matrix( fit3sls, which = "xHat" ) ) / 40 ), check.attributes = FALSE )
```

`coef.systemfit` *Coefficients of systemfit object*

Description

These functions extract the coefficients from an object returned by `systemfit`.

Usage

```
   ## S3 method for class 'systemfit'
coef( object, modified.regMat = FALSE, ... )

   ## S3 method for class 'systemfit.equation'
coef( object, ... )

   ## S3 method for class 'summary.systemfit'
coef( object, modified.regMat = FALSE, ... )

   ## S3 method for class 'summary.systemfit.equation'
coef( object, ... )
```

Arguments

`object` an object of class `systemfit`, `systemfit.equation`, `summary.systemfit`, or `summary.systemfit.equation`.

`modified.regMat` logical. If `TRUE`, the coefficients of the modified regressor matrix (original regressor matrix post-multiplied by `restrict.regMat`) rather than the coefficients of the original regressor matrix are returned.

`...` other arguments.

Value

coef.systemfit returns a vector of all estimated coefficients.

coef.systemfit.equation returns a vector of the estimated coefficients of a single equation.

coef.summary.systemfit returns a matrix of all estimated coefficients, their standard errors, t-values, and p-values.

coef.summary.systemfit.equation returns a matrix of the estimated coefficients of a single equation, their standard errors, t-values, and p-values.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit, coef

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## all coefficients
coef( fitols )
coef( summary( fitols ) )

## coefficients of the first equation
coef( fitols$eq[[1]] )
coef( summary( fitols$eq[[1]] ) )

## coefficients of the second equation
coef( fitols$eq[[2]] )
coef( summary( fitols$eq[[2]] ) )

## estimation with restriction by modifying the regressor matrix
modReg <- matrix( 0, 7, 6 )
colnames( modReg ) <- c( "demIntercept", "demPrice", "demIncome",
   "supIntercept", "supPrice2", "supTrend" )
modReg[ 1, "demIntercept" ] <- 1
modReg[ 2, "demPrice" ]     <- 1
modReg[ 3, "demIncome" ]    <- 1
modReg[ 4, "supIntercept" ] <- 1
modReg[ 5, "supPrice2" ]    <- 1
modReg[ 6, "supPrice2" ]    <- 1
modReg[ 7, "supTrend" ]     <- 1
fitols3 <- systemfit( system, data = Kmenta, restrict.regMat = modReg )
```

```
coef( fitols3, modified.regMat  = TRUE )
coef( fitols3 )
```

`confint.systemfit` *Confidence intervals of coefficients*

Description

These functions calculate the confidence intervals of the coefficients from an object returned by `systemfit`.

Usage

```
   ## S3 method for class 'systemfit'
confint( object, parm = NULL, level = 0.95,
      useDfSys = NULL, ... )

   ## S3 method for class 'systemfit.equation'
confint( object, parm, level = 0.95,
      useDfSys = NULL, ... )
```

Arguments

`object`	an object of class `systemfit` or `systemfit.equation`.
`parm`	not used yet.
`level`	confidence level.
`useDfSys`	logical. Use the degrees of freedom of the whole system (in place of the degrees of freedom of the single equation) to calculate the confidence intervals of the coefficients. If it not specified (`NULL`), it is set to `TRUE` if restrictions on the coefficients are imposed and `FALSE` otherwise.
`...`	other arguments.

Value

An object of class `confint.systemfit`, which is a matrix with columns giving lower and upper confidence limits for each estimated coefficient. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

`systemfit`, `print.confint.systemfit`, `confint`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## confidence intervals of all coefficients
confint( fitols )

## confidence intervals of the coefficients of the first equation
confint( fitols$eq[[1]] )

## confidence intervals of the coefficients of the second equation
confint( fitols$eq[[2]] )
```

`correlation.systemfit` *Correlation between Predictions from Equation i and j*

Description

`correlation` returns a vector of the correlations between the predictions of two equations in a set of equations. The correlation between the predictions is defined as,

$$r_{ijk} = \frac{x'_{ik} C_{ij} x_{jk}}{\sqrt{(x'_{ik} C_{ii} x_{ik})(x'_{jk} C_{jj} x_{jk})}}$$

where r_{ijk} is the correlation between the predicted values of equation i and j and C_{ij} is the cross-equation variance-covariance matrix between equations i and j.

Usage

```
correlation.systemfit( results, eqni, eqnj )
```

Arguments

`results`	an object of type `systemfit`.
`eqni`	index for equation i
`eqnj`	index for equation j

Value

`correlation` returns a vector of the correlations between the predicted values in equation i and equation j.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>

References

Greene, W. H. (1993) *Econometric Analysis, Second Edition*, Macmillan.

Hasenauer, H; Monserud, R and T. Gregoire. (1998) Using Simultansous Regression Techniques with Individual-Tree Growth Models. *Forest Science*. 44(1):87-95

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

See Also

systemfit

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
inst <- ~ income + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform 2SLS on each of the equations in the system
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )
print( fit2sls )
print( fit2sls$rcov )

## perform the 3SLS
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta )
print( fit3sls )
print( "covariance of residuals used for estimation (from 2sls)" )
print( fit3sls$rcovest )
print( "covariance of residuals" )
print( fit3sls$rcov )

## examine the correlation between the predicted values
## of suppy and demand by plotting the correlation over
## the value of q
r12 <- correlation.systemfit( fit3sls, 1, 2 )
plot( Kmenta$consump, r12, main="correlation between predictions from supply and demand" )
```

---

createSystemfitModel   *Create a Model for systemfit*

Description

This function creates a model that can be estimated by systemfit. The data, disturbances, and — if not provided by the user — the coefficients as well as the disturbance covariance matrix are generated by random numbers.

Usage

```
createSystemfitModel( nEq, nRegEq, nObs, coef = NULL, sigma = NULL )
```

Arguments

nEq	the number of equations.
nRegEq	the number of regressors in each equation (without the intercept).
nObs	the number of observations.
coef	an optional vector of coefficients.
sigma	an optional covariance matrix of the disturbance terms.

Value

`createSystemfitModel` returns a list with following elements:

formula	a list of the model equations (objects of class `formula`).
data	a `data.frame` that contains the data.
coef	a vector of (true) coefficients.
sigma	the covariance matrix of the disturbance terms.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit

Examples

```
## create a model by random numbers
systemfitModel <- createSystemfitModel( 3, 4, 100 )

## estimate this model by "SUR"
fitsur <- systemfit( systemfitModel$formula, "SUR", data = systemfitModel$data )

## compare the "true" and the estimated coefficients
cbind( systemfitModel$coef, coef( fitsur ) )
```

estfun.systemfit *Extract Gradients of the Objective Function at each Observation*

Description

Extract the gradients of the objective function with respect to the coefficients evaluated at each observation ('Empirical Estimating Function', see `estfun`).

Usage

```
## S3 method for class 'systemfit'
estfun( x, residFit = TRUE, ... )
```

Arguments

`x`	an object of class `systemfit`.
`residFit`	logical. If `FALSE`, the residuals are calculated based on observed regressors. If `TRUE`, the residuals are calculated based on fitted regressors. This argument is ignored if no instrumental variable are used.
`...`	further arguments (currently ignored).

Value

Matrix of gradients of the objective function with respect to the coefficients evaluated at each observation.

Warnings

The **sandwich** package must be loaded before this method can be used.

In specific estimations with the 3SLS method, not all columns of the matrix returned by the `estfun` method sum up to zero, which indicates that an inappropriate estimating function is returned. This can be either with argument `residFit` set to `TRUE` or with this argument set to `FALSE` or even in both cases. This problem depends on the formula used for the 3SLS estimation and seems to be related to unbalanced systems and systems where different instruments are used in different equations.

Author(s)

Arne Henningsen

See Also

`estfun`, `systemfit`.

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )
inst <- ~ income + farmPrice + trend

## OLS estimation
fitols <- systemfit( system, "OLS", data = Kmenta )

## obtain the estimation function
library( "sandwich" )
estfun( fitols )

## this is only true for OLS models
all.equal( estfun( fitols ),
   unlist( residuals( fitols ) ) * model.matrix( fitols ) )

# each column should sum up to (approximately) zero
colSums( estfun( fitols ) )


## 2SLS estimation
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )

## obtain the estimation function
estfun( fit2sls )

## this is only true for 2SLS models
all.equal( estfun( fit2sls ),
   drop( rep( Kmenta$consump, 2 ) -  model.matrix( fit2sls, which = "xHat" ) %*%
   coef( fit2sls ) ) * model.matrix( fit2sls, which = "xHat" ) )
all.equal( estfun( fit2sls, residFit = FALSE ),
   unlist( residuals( fit2sls ) ) * model.matrix( fit2sls, which = "xHat" ) )

# each column should sum up to (approximately) zero
colSums( estfun( fit2sls ) )
colSums( estfun( fit2sls, residFit = FALSE ) )


## iterated SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta, maxit = 100 )

## obtain the estimation function
estfun( fitsur )

## this should be true for SUR and WLS models
all.equal( estfun( fitsur ),
   unlist( residuals( fitsur ) ) *
   ( ( solve( fitsur$residCovEst ) %x% diag( nrow( Kmenta ) ) ) %*%
      model.matrix( fitsur ) ), check.attributes = FALSE )
```

```
# each column should sum up to (approximately) zero
colSums( estfun( fitsur ) )


## 3SLS estimation
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta )

## obtain the estimation function
estfun( fit3sls )
estfun( fit3sls, residFit = FALSE )

## this should be true for 3SLS and W2SLS models
all.equal( estfun( fit3sls ),
   drop( rep( Kmenta$consump, 2 ) -
   model.matrix( fit2sls, which = "xHat" ) %*% coef( fit3sls ) ) *
   ( ( solve( fit3sls$residCovEst ) %x% diag( nrow( Kmenta ) ) ) %*%
      model.matrix( fit3sls, which = "xHat" ) ), check.attributes = FALSE )
all.equal( estfun( fit3sls, residFit = FALSE ),
   unlist( residuals( fit3sls ) ) *
   ( ( solve( fit3sls$residCovEst ) %x% diag( nrow( Kmenta ) ) ) %*%
      model.matrix( fit3sls, which = "xHat" ) ), check.attributes = FALSE )

# each column should sum up to (approximately) zero
colSums( estfun( fit3sls ) )
colSums( estfun( fit3sls, residFit = FALSE ) )
```

`fitted.systemfit` *Fitted values*

Description

These functions extract the fitted values from an object returned by `systemfit`.

Usage

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

   ## S3 method for class 'systemfit.equation'
fitted( object, na.rm = FALSE, ... )
```

Arguments

`object`	an object of class `systemfit` or `systemfit.equation`.
`na.rm`	a logical value indicating whether `NA` values (corresponding to observations that were not included in the estimation) should be removed from the vector of fitted values before it is returned.
`...`	other arguments.

Value

`fitted.systemfit` returns a data.frame of all fitted values, where each column contains the fitted values of one equation.

`fitted.systemfit.equation` returns a vector of the fitted values of a single equation.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

`systemfit`, `fitted`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## all fitted values
fitted( fitols )

## fitted values of the first equation
fitted( fitols$eq[[1]] )

## fitted values of the second equation
fitted( fitols$eq[[2]] )
```

`formula.systemfit` *Model Formulae of systemfit Objects*

Description

This method extracts the model formulae from fitted objects returned by `systemfit`.

Usage

```
   ## S3 method for class 'systemfit'
formula( x, ... )
   ## S3 method for class 'systemfit.equation'
formula( x, ... )
```

Arguments

x	an object of class systemfit.
...	currently not used.

Value

formula.systemfit.equation returns the formula of a single equation of a systemfit object. formula.systemfit.equation returns a list of formulae: one formula object for each equation of the systemfit object.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit, formula

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform a SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta )

## formula of the second equation
formula( fitsur$eq[[2]] )

## all formulae of the system
formula( fitsur )
```

GrunfeldGreene *Grunfeld Data as published by Greene (2003)*

Description

Panel data on 5 US firms for the years 1935-1954.

Usage

```
data("GrunfeldGreene")
```

Format

A data frame containing 20 annual observations on 3 variables for 5 firms.

invest gross investment.

value market value of the firm (at the end of the previous year).

capital capital stock of the firm (at the end of the previous year).

firm name of the firm ("General Motors", "Chrysler", "General Electric", "Westinghouse" or "US Steel").

year year.

Details

There exist several different versions of this data set, and this version is considered incorrect (see https://web.archive.org/web/20170426034143/http://web.stanford.edu/~clint/bench/grunfeld.htm for details). However, we provide this incorrect version to replicate the results published in Theil (1971) and Greene (2003). A correct version of this data set with 5 additional firms is available in the `Ecdat` package (data set `Grunfeld`).

Source

Greene (2003), Appendix F, Data Sets Used in Applications, Table F13.1. https://pages.stern.nyu.edu/~wgreene/Text/econometricanalysis.htm (a subset of this data set is available in Theil (1971), p. 296).

References

Greene, W.H. (2003). *Econometric Analysis*, 5th edition. Prentice Hall, Upper Saddle River (NJ).

Grunfeld, Y. (1958). *The Determinants of Corporate Investment*, Unpublished Ph.D. Dissertation, University of Chicago.

Theil, Henri (1971). *Principles of Econometrics*, John Wiley & Sons, New York.

Examples

```
## Repeating the OLS and SUR estimations in Greene (2003, pp. 351)
data( "GrunfeldGreene" )
if( requireNamespace( 'plm', quietly = TRUE ) ) {
library( "plm" )
GGPanel <- pdata.frame( GrunfeldGreene, c( "firm", "year" ) )
formulaGrunfeld <- invest ~ value + capital
# OLS
greeneOls <- systemfit( formulaGrunfeld, "OLS",
   data = GGPanel )
summary( greeneOls )
sapply( greeneOls$eq, function(x){return(summary(x)$ssr/20)} ) # sigma^2
# OLS Pooled
greeneOlsPooled <- systemfit( formulaGrunfeld, "OLS",
   data = GGPanel, pooled = TRUE )
summary( greeneOlsPooled )
sum( sapply( greeneOlsPooled$eq, function(x){return(summary(x)$ssr)}) )/97 # sigma^2
```

```
# SUR
greeneSur <- systemfit( formulaGrunfeld, "SUR",
   data = GGPanel, methodResidCov = "noDfCor" )
summary( greeneSur )
# SUR Pooled
greeneSurPooled <- systemfit( formulaGrunfeld, "SUR",
   data = GGPanel, pooled = TRUE, methodResidCov = "noDfCor",
   residCovWeighted = TRUE )
summary( greeneSurPooled )

## Repeating the OLS and SUR estimations in Theil (1971, pp. 295, 300)
GrunfeldTheil <- subset( GrunfeldGreene,
   firm %in% c( "General Electric", "Westinghouse" ) )
GTPanel <- pdata.frame( GrunfeldTheil, c( "firm", "year" ) )
formulaGrunfeld <- invest ~ value + capital
# OLS
theilOls <- systemfit( formulaGrunfeld, "OLS",
   data = GTPanel )
summary( theilOls )
# SUR
theilSur <- systemfit( formulaGrunfeld, "SUR",
   data = GTPanel, methodResidCov = "noDfCor" )
summary( theilSur )
}
```

---

hausman.systemfit *Hausman Test*

Description

hausman.systemfit returns the Hausman statistic for a specification test.

Usage

```
hausman.systemfit( results2sls, results3sls )
```

Arguments

results2sls	result of a *2SLS* (limited information) estimation returned by systemfit.
results3sls	result of a *3SLS* (full information) estimation returned by systemfit.

Details

The null hypotheses of the test is that all exogenous variables are uncorrelated with all disturbance terms. Under this hypothesis both the 2SLS and the 3SLS estimator are consistent but only the 3SLS estimator is (asymptotically) efficient. Under the alternative hypothesis the 2SLS estimator is consistent but the 3SLS estimator is inconsistent.

The Hausman test statistic is

$$m = (b_2 - b_3)'(V_2 - V_3)(b_2 - b_3)$$

where \$b_2\$ and \$V_2\$ are the estimated coefficients and their variance covariance matrix of a *2SLS* estimation and \$b_3\$ and \$V_3\$ are the estimated coefficients and their variance covariance matrix of a *3SLS* estimation.

Value

`hausman.systemfit` returns a list of the class `htest` that contains following elements:

`q`	vector of the differences between the estimated coefficients.
`qVar`	variance covariance matrix of `q` (difference between the variance covariance matrices of the estimated coefficients).
`statistic`	the Hausman test statistic.
`parameter`	degrees of freedom.
`p.value`	P-value of the test.
`method`	character string describing this test.
`data.name`	name of the data.frame used for estimation.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>,
Arne Henningsen <arne.henningsen@googlemail.com>

References

Greene, W. H. (1993) *Econometric Analysis, Fifth Edition*, Macmillan.

Hausman, J. A. (1978) Specification Tests in Econometrics. *Econometrica*. 46:1251-1271.

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

See Also

`systemfit`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
inst <- ~ income + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform the estimations
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta )

## perform the Hausman test
h <- hausman.systemfit( fit2sls, fit3sls )
print( h )
```

`KleinI` *Klein Model I*

Description

Data for Klein's (1950) Model I of the US economy.

Usage

```
data("KleinI")
```

Format

A data frame containing annual observations from 1920 to 1941

year Year.

consump Consumption.

corpProf Corporate profits.

corpProfLag Corporate profits of the previous year.

privWage Private wage bill.

invest Investment.

capitalLag Capital stock of the previous year.

gnp Gross national product.

gnpLag Gross national product of the previous year.

govWage Government wage bill.

govExp Government spending.

taxes Taxes.

wages Sum of private and government wage bill.

trend time trend measured as years from 1931.

Source

Greene (2003), Appendix F, Data Sets Used in Applications, Table F15.1.

https://pages.stern.nyu.edu/~wgreene/Text/econometricanalysis.htm

References

Greene, W.H. (2003). *Econometric Analysis*, 5th edition. Prentice Hall, Upper Saddle River (NJ).

Klein, L. (1950). *Economic Fluctuations in the United States, 1921–1941*. John Wiley, New York.

Examples

```
## Repeating the estimations of Klein's (1950) Model I
## in Greene (2003, pp. 381 and 412)
data( "KleinI" )
eqConsump  <- consump ~ corpProf + corpProfLag + wages
eqInvest   <- invest ~ corpProf + corpProfLag + capitalLag
eqPrivWage <- privWage ~ gnp + gnpLag + trend
inst <- ~ govExp + taxes + govWage + trend + capitalLag + corpProfLag + gnpLag
system <- list( Consumption = eqConsump, Investment = eqInvest,
   PrivateWages = eqPrivWage )
# OLS
kleinOls <- systemfit( system, data = KleinI )
summary( kleinOls )
# 2SLS
klein2sls <- systemfit( system, "2SLS", inst = inst, data = KleinI,
   methodResidCov = "noDfCor" )
summary( klein2sls )
# 3SLS
klein3sls <- systemfit( system, "3SLS", inst = inst, data = KleinI,
   methodResidCov = "noDfCor" )
summary( klein3sls )
# I3SLS
kleinI3sls <- systemfit( system, "3SLS", inst = inst, data = KleinI,
   methodResidCov = "noDfCor", maxit = 500 )
summary( kleinI3sls )
```

Kmenta *Partly Artificial Data on the U. S. Economy*

Description

These are partly contrived data from Kmenta (1986), constructed to illustrate estimation of a simultaneous-equation model.

Usage

```
data("Kmenta")
```

Format

This data frame contains 20 annual observations of 5 variables:

consump food consumption per capita.

price ratio of food prices to general consumer prices.

income disposable income in constant dollars.

farmPrice ratio of preceding year's prices received by farmers to general consumer prices.

trend time trend in years.

Details

The exogenous variables `income`, `farmPrice`, and `trend` are based on real data; the endogenous variables `price` and `consump` were generated by simulation.

Source

Kmenta (1986), Table 13-1, p. 687.

References

Kmenta, J. (1986). *Elements of Econometrics*, Second Edition, Macmillan, New York.

Examples

```
## Replicating the estimations in Kmenta (1986), p. 712, Tab 13-2
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
inst <- ~ income + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## OLS estimation
fitOls <- systemfit( system, data = Kmenta )
summary( fitOls )

## 2SLS estimation
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )
summary( fit2sls )

## 3SLS estimation
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta )
summary( fit3sls )

## I3LS estimation
fitI3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta,
   maxit = 250 )
summary( fitI3sls )
```

`linearHypothesis.systemfit` *Test Linear Hypothesis*

Description

Testing linear hypothesis on the coefficients of a system of equations by an F-test or Wald-test.

Usage

```
## S3 method for class 'systemfit'
linearHypothesis( model,
      hypothesis.matrix, rhs = NULL, test = c( "FT", "F", "Chisq" ),
      vcov. = NULL, ... )
```

Arguments

`model`	a fitted object of type `systemfit`.
`hypothesis.matrix`	matrix (or vector) giving linear combinations of coefficients by rows, or a character vector giving the hypothesis in symbolic form (see documentation of `linearHypothesis` in package "car" for details).
`rhs`	optional right-hand-side vector for hypothesis, with as many entries as rows in the hypothesis matrix; if omitted, it defaults to a vector of zeroes.
`test`	character string, `"FT"`, `"F"`, or `"Chisq"`, specifying whether to compute Theil's finite-sample F test (with approximate F distribution), the finite-sample Wald test (with approximate F distribution), or the large-sample Wald test (with asymptotic Chi-squared distribution).
`vcov.`	a function for estimating the covariance matrix of the regression coefficients or an estimated covariance matrix (function `vcov` is used by default).
`...`	further arguments passed to `linearHypothesis.default` (package "car").

Details

Theil's F statistic for sytems of equations is

$$F = \frac{(R\hat{b} - q)'(R(X'(\Sigma \otimes I)^{-1}X)^{-1}R')^{-1}(R\hat{b} - q)/j}{\hat{e}'(\Sigma \otimes I)^{-1}\hat{e}/(M \cdot T - K)}$$

where j is the number of restrictions, M is the number of equations, T is the number of observations per equation, K is the total number of estimated coefficients, and Σ is the estimated residual covariance matrix. Under the null hypothesis, F has an approximate F distribution with j and $M \cdot T - K$ degrees of freedom (Theil, 1971, p. 314).

The F statistic for a Wald test is

$$F = \frac{(R\hat{b} - q)'(R\widehat{Cov}[\hat{b}]R')^{-1}(R\hat{b} - q)}{j}$$

Under the null hypothesis, F has an approximate F distribution with j and $M \cdot T - K$ degrees of freedom (Greene, 2003, p. 346).

The χ^2 statistic for a Wald test is

$$W = (R\hat{b} - q)'(R\widehat{Cov}[\hat{b}]R')^{-1}(R\hat{b} - q)$$

Asymptotically, W has a χ^2 distribution with j degrees of freedom under the null hypothesis (Greene, 2003, p. 347).

Value

An object of class anova, which contains the residual degrees of freedom in the model, the difference in degrees of freedom, the test statistic (either F or Wald/Chisq) and the corresponding p value. See documentation of `linearHypothesis` in package "car".

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

References

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

Theil, Henri (1971) *Principles of Econometrics*, John Wiley & Sons, New York.

See Also

`systemfit`, `linearHypothesis` (package "car"), `lrtest.systemfit`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## unconstrained SUR estimation
fitsur <- systemfit( system, method = "SUR", data=Kmenta )

# create hypothesis matrix to test whether beta_2 = \beta_6
R1 <- matrix( 0, nrow = 1, ncol = 7 )
R1[ 1, 2 ] <- 1
R1[ 1, 6 ] <- -1
# the same hypothesis in symbolic form
restrict1 <- "demand_price - supply_farmPrice = 0"

## perform Theil's F test
linearHypothesis( fitsur, R1 )  # rejected
linearHypothesis( fitsur, restrict1 )

## perform Wald test with F statistic
linearHypothesis( fitsur, R1, test = "F" )  # rejected
linearHypothesis( fitsur, restrict1 )

## perform Wald-test with chi^2 statistic
linearHypothesis( fitsur, R1, test = "Chisq" )  # rejected
linearHypothesis( fitsur, restrict1, test = "Chisq" )

# create hypothesis matrix to test whether beta_2 = - \beta_6
R2 <- matrix( 0, nrow = 1, ncol = 7 )
R2[ 1, 2 ] <- 1
R2[ 1, 6 ] <- 1
```

```
# the same hypothesis in symbolic form
restrict2 <- "demand_price + supply_farmPrice = 0"

## perform Theil's F test
linearHypothesis( fitsur, R2 )  # accepted
linearHypothesis( fitsur, restrict2 )

## perform Wald test with F statistic
linearHypothesis( fitsur, R2, test = "F" )  # accepted
linearHypothesis( fitsur, restrict2 )

## perform Wald-test with chi^2 statistic
linearHypothesis( fitsur, R2, test = "Chisq" )  # accepted
linearHypothesis( fitsur, restrict2, test = "Chisq" )
```

`logLik.systemfit` *Log-Likelihood value of systemfit object*

Description

This method calculates the log-likelihood value of a fitted object returned by `systemfit`.

Usage

```
   ## S3 method for class 'systemfit'
logLik( object, residCovDiag = FALSE, ... )
```

Arguments

`object`	an object of class `systemfit`.
`residCovDiag`	logical. If this argument is set to `TRUE`, the residual covaraince matrix that is used for calculating the log-likelihood value is assumed to be diagonal, i.e. all covariances are set to zero. This may be desirable for models estimated by OLS, 2SLS, WLS, and W2SLS.
`...`	currently not used.

Details

The residual covariance matrix that is used for calculating the log-likelihood value is calculated based on the actually obtained (final) residuals (not correcting for degrees of freedom). In case of systems of equations with unequal numbers of observations, the calculation of the residual covariance matrix is only based on the residuals/observations that are available in all equations.

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 <arne.henningsen@googlemail.com>

See Also

systemfit, logLik

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform a SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta )

## residuals of all equations
logLik( fitsur )
```

---

lrtest.systemfit *Likelihood Ratio test for Equation Systems*

Description

Testing linear hypothesis on the coefficients of a system of equations by a Likelihood Ratio test.

Usage

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

Arguments

`object`	a fitted model object of class `systemfit`.
`...`	further fitted model objects of class `systemfit`.

Details

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

The LR-statistic for sytems of equations is

$$LR = T \cdot \left(log \left| \hat{\hat{\Sigma}}_r \right| - log \left| \hat{\hat{\Sigma}}_u \right| \right)$$

where T is the number of observations per equation, and $\hat{\hat{\Sigma}}_r$ and $\hat{\hat{\Sigma}}_u$ are the residual covariance matrices calculated by formula "0" (see systemfit) of the restricted and unrestricted estimation, respectively. Asymptotically, LR has a χ^2 distribution with j degrees of freedom under the null hypothesis (Green, 2003, p. 349).

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 <arne.henningsen@googlemail.com>

References

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

See Also

systemfit, lrtest (package "lmtest"), linearHypothesis.systemfit

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## unconstrained SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta )

# create restriction matrix to impose \eqn{beta_2 = \beta_6}
R1 <- matrix( 0, nrow = 1, ncol = 7 )
R1[ 1, 2 ] <- 1
R1[ 1, 6 ] <- -1

## constrained SUR estimation
fitsur1 <- systemfit( system, "SUR", data = Kmenta, restrict.matrix = R1 )

## perform LR-test
lrTest1 <- lrtest( fitsur1, fitsur )
print( lrTest1 )   # rejected

# create restriction matrix to impose \eqn{beta_2 = - \beta_6}
R2 <- matrix( 0, nrow = 1, ncol = 7 )
R2[ 1, 2 ] <- 1
R2[ 1, 6 ] <- 1

## constrained SUR estimation
fitsur2 <- systemfit( system, "SUR", data = Kmenta, restrict.matrix = R2 )

## perform LR-test
lrTest2 <- lrtest( fitsur2, fitsur )
print( lrTest2 )   # accepted
```

`model.frame.systemfit` *Extracting the Data of a systemfit Object*

Description

These functions return the data used by `systemfit` to estimate a system of equations.

Usage

```
## S3 method for class 'systemfit'
model.frame( formula, ... )

## S3 method for class 'systemfit.equation'
model.frame( formula, ... )
```

Arguments

`formula`	an object of class `systemfit` or `systemfit.equation`.
`...`	currently ignored.

Value

`model.frame.systemfit` returns a simple data frame (without a 'terms' attribute) that contains all variables used to estimate the entire system of equations.

`model.frame.systemfit.equation` returns a model frame (data frame with a 'terms' attribute) that contains all variables used to estimate the respective equation.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

`systemfit`, `model.frame`, and `model.matrix.systemfit`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS of the system
fitols <- systemfit( system, data = Kmenta )

## data used to estimate the entire system
model.frame( fitols )
```

```
## data used to estimate the first equation
model.frame( fitols$eq[[ 1 ]] )
```

---

model.matrix.systemfit
Construct Design Matrices for Systems of Equations

Description

These functions create design matrices from objects returned by `systemfit`.

Usage

```
   ## S3 method for class 'systemfit'
model.matrix( object, which = "x", ... )

   ## S3 method for class 'systemfit.equation'
model.matrix( object, which = "x", ... )
```

Arguments

`object`	an object of class `systemfit` or `systemfit.equation`.
`which`	character string: `"x"` indicates the usual model matrix of the regressors, `"xHat"` indicates the model matrix of the fitted regressors, `"z"` indicates the matrix of instrumental variables.
`...`	currently ignored.

Value

`model.matrix.systemfit` returns a design matrix to estimate the specified system of equations.

`model.matrix.systemfit.equation` returns a design matrix to estimate the specified formula of the respective equation.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

`systemfit`, `model.matrix`, and `model.frame.systemfit`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS of the system
fitols <- systemfit( system, data = Kmenta )

## design matrix of the entire system
model.matrix( fitols )

## design matrix of the first equation
model.matrix( fitols$eq[[ 1 ]] )
```

---

nlsystemfit *Nonlinear Equation System Estimation*

Description

Fits a set of structural nonlinear equations using Ordinary Least Squares (OLS), Seemingly Unrelated Regression (SUR), Two-Stage Least Squares (2SLS), Three-Stage Least Squares (3SLS).

Usage

```
nlsystemfit( method="OLS", eqns, startvals,
                      eqnlabels=c(as.character(1:length(eqns))), inst=NULL,
                      data=list(), solvtol=.Machine$double.eps,
                      maxiter=1000, ... )
```

Arguments

method	the estimation method, one of "OLS", "SUR", "2SLS", "3SLS".
eqns	a list of structural equations to be estimated.
startvals	a list of starting values for the coefficients.
eqnlabels	an optional list of character vectors of names for the equation labels.
inst	one-sided model formula specifying instrumental variables or a list of one-sided model formulas if different instruments should be used for the different equations (only needed for 2SLS, 3SLS and GMM estimations).
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment from which nlsystemfit is called.
solvtol	tolerance for detecting linear dependencies in the columns of X in the `qr` function calls.
maxiter	the maximum number of iterations for the `nlm` function.
...	arguments passed to `nlm`.

Details

The nlsystemfit function relies on nlm to perform the minimization of the objective functions and the qr set of functions.

A system of nonlinear equations can be written as:

$$\epsilon_t = q(y_t, x_t, \theta)$$
$$z_t = Z(x_t)$$

where ϵ_t are the residuals from the y observations and the function evaluated at the coefficient estimates.

The objective functions for the methods are:

Method	Instruments	Objective Function	Covariance of θ
OLS	no	$r'r$	$(X(diag(S)^{-1} \bigotimes I)X)^{-1}$
SUR	no	$r'(diag(S)_{OLS}^{-1} \bigotimes I)r$	$(X(S^{-1} \bigotimes I)X)^{-1}$
2SLS	yes	$r'(I \bigotimes W)r$	$(X(diag(S)^{-1} \bigotimes I)X)^{-1}$
3SLS	yes	$r'(S_{2SLS}^{-1} \bigotimes W)r$	$(X(diag(S)^{-1} \bigotimes W)X)^{-1}$

where, r is a column vector for the residuals for each equation, S is variance-covariance matrix between the equations ($\hat{\sigma}_{ij} = (\hat{e}_i'\hat{e}_j)/\sqrt{(T-k_i)*(T-k_j)}$), X is matrix of the partial derivates with respect to the coefficients, W is a matrix of the instrument variables $Z(Z'Z)^{-1}Z$, Z is a matrix of the instrument variables, and I is an nxn identity matrix.

The SUR and 3SLS methods requires two solutions. The first solution for the SUR is an OLS solution to obtain the variance-covariance matrix. The 3SLS uses the variance-covatiance from a 2SLS solution, then fits all the equations simultaneously.

The user should be aware that the function is **VERY** sensative to the starting values and the nlm function may not converge. The nlm function will be called with the `typsize` argument set the absolute values of the starting values for the OLS and 2SLS methods. For the SUR and 3SLS methods, the `typsize` argument is set to the absolute values of the resulting OLS and 2SLS coefficient estimates from the nlm result structre. In addition, the starting values for the SUR and 3SLS methods are obtained from the OLS and 2SLS coefficient estimates to shorten the number of iterations. The number of iterations reported in the summary are only those used in the last call to nlm, thus the number of iterations in the OLS portion of the SUR fit and the 2SLS portion of the 3SLS fit are not included.

Value

`nlsystemfit` returns a list of the class `nlsystemfit.system` and contains all results that belong to the whole system. This list contains one special object: "eq". It is a list and contains one object for each estimated equation. These objects are of the class `nlsystemfit.equation` and contain the results that belong only to the regarding equation.

The objects of the class `nlsystemfit.system` and `nlsystemfit.equation` have the following components (the elements of the latter are marked with an asterisk ($*$)):

`eq` a list object that contains a list object for each equation.

`method` estimation method.

`resids` an $n \times g$ matrix of the residuals.

`g` number of equations.

`n` total number of observations.

`k` total number of coefficients.

`b` vector of all estimated coefficients.

`se` estimated standard errors of `b`.

`t` t values for `b`.

`p` p values for `b`.

`bcov` estimated covariance matrix of `b`.

`rcov` estimated residual covariance matrix.

`drcov` determinant of `rcov`.

`rcovest` residual covariance matrix used for estimation (only SUR and 3SLS).

`rcor` estimated residual correlation matrix.

`nlmest` results from the nlm function call

`solvetol` tolerance level when inverting a matrix or calculating a determinant.

elements of the class nlsystemfit.eq

`eq` a list that contains the results that belong to the individual equations.

`eqnlabel*` the equation label of the ith equation (from the labels list).

`formula*` model formula of the ith equation.

`n*` number of observations of the ith equation.

`k*` number of coefficients/regressors in the ith equation.

`df*` degrees of freedom of the ith equation.

`b*` estimated coefficients of the ith equation.

`se*` estimated standard errors of `b`.

`t*` t values for `b`.

`p*` p values for `b`.

`covb*` estimated covariance matrix of `b`.

`predicted*` vector of predicted values of the ith equation.

`residuals*` vector of residuals of the ith equation.

`ssr*` sum of squared residuals of the ith equation.

`mse*` estimated variance of the residuals (mean of squared errors) of the ith equation.

`s2*` estimated variance of the residuals ($\hat{\sigma}^2$) of the ith equation.

`rmse*` estimated standard error of the residulas (square root of mse) of the ith equation.

`s*` estimated standard error of the residuals ($\hat{\sigma}$) of the ith equation.

`r2*` R-squared (coefficient of determination).

`adjr2*` adjusted R-squared value.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>

References

Gallant, R. H. (1987) *Nonlinear Equation Estimation*, John Wiley and Sons, 610 pp.

SAS Institute (1999) *SAS/ETS User's Guide, Version 8*, Cary NC: SAS Institute 1546 pp.

See Also

systemfit, nlm, and qr

Examples

```
library( systemfit )
data( ppine )

hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev + h4*cr)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr + d4*ba  )
labels <- list( "height.growth", "diameter.growth" )
inst <- ~ tht + dbh + elev + cr + ba
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001, h4=0.08,
                  d0=-0.5, d1=0.009, d2=0.25, d3=0.005, d4=-0.02 )
model <- list( hg.formula, dg.formula )

model.ols <- nlsystemfit( "OLS", model, start.values, data=ppine, eqnlabels=labels )
print( model.ols )

model.sur <- nlsystemfit( "SUR", model, start.values, data=ppine, eqnlabels=labels )
print( model.sur )

model.2sls <- nlsystemfit( "2SLS", model, start.values, data=ppine,
   eqnlabels=labels, inst=inst )
print( model.2sls )

model.3sls <- nlsystemfit( "3SLS", model, start.values, data=ppine,
                                    eqnlabels=labels, inst=inst )
print( model.3sls )
```

---

ppine *Tree Growth Data for Ponderosa Pine*

Description

A subset of tree growth observations from a Ponderosa pine growth database.

The ppine data frame has 166 rows and 8 columns.

Usage

```
data(ppine)
```

Format

This data frame contains the following columns:

elev Altitude of the plot, in feet above mean sea level.

smi Summer moisture index is the ratio of growing season heating degree days to growing season precipitation.

dbh Diameter of the tree at breast height (4.5 feet).

tht Total stem height for the tree.

cr Crown ratio code. The scale is from 1 to 9 where a crown class of one represents a crown ratio between 0 and 15 percent. A crown ratio code of 2 represents a crown ratio value between 16 and 25%,...,8=76-85%, 9 >=85%.

ba Plot basal area at the beginning of the growth period.

dg Five-year diameter increment.

hg Five-year height increment.

Details

The exogenous variables are `elev`, `smi`, `dbh`, `tht`, `cr`, and `ba`; the endogenous variables `dg` and `hg`. There are no lagged variables in the dataset and the observations are for a single remeasurement.

The data was provided by the USDA Forest Service Intermountain Research Station.

Source

William R. Wykoff `<wwykoff@fs.fed.us>` *Rocky Mountain Research Station, 1221 South Main Street, Moscow, ID 83843*

Examples

```
data(ppine)
```

`predict.systemfit` *Predictions from System Estimation*

Description

Returns the predicted values, their standard errors and the confidence limits of prediction.

Usage

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

## S3 method for class 'systemfit.equation'
predict( object, newdata = NULL,
                             se.fit = FALSE, se.pred = FALSE,
                             interval = "none", level=0.95,
                             useDfSys = NULL, ... )
```

Arguments

`object`	an object of class `systemfit` or `systemfit.equation`.
`newdata`	An optional data frame in which to look for variables with which to predict. If it is `NULL`, the fitted values are returned.
`se.fit`	return the standard error of the fitted values?
`se.pred`	return the standard error of prediction?
`interval`	Type of interval calculation ("none", "confidence" or "prediction")
`level`	Tolerance/confidence level.
`useDfSys`	logical. Use the degrees of freedom of the whole system (in place of the degrees of freedom of the single equation) to calculate the confidence or prediction intervals. If it not specified (`NULL`), it is set to `TRUE` if restrictions on the coefficients are imposed and `FALSE` otherwise.
`...`	additional optional arguments.

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.systemfit` returns a dataframe that contains for each equation the predicted values ("<eqnLable>.pred") and if requested the standard errors of the fitted values ("<eqnLable>.se.fit"), the standard errors of the prediction ("<eqnLable>.se.pred"), and the lower ("<eqnLable>.lwr") and upper ("<eqnLable>.upr") limits of the confidence or prediction interval(s).

`predict.systemfit.equation` returns a dataframe that contains the predicted values ("fit") and if requested the standard errors of the fitted values ("se.fit"), the standard errors of the prediction ("se.pred"), and the lower ("lwr") and upper ("upr") limits of the confidence or prediction interval(s).

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

References

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.

See Also

systemfit, predict

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## OLS estimation
fitols <- systemfit( system, data=Kmenta )

## predicted values and limits
predict( fitols )

## predicted values of the first equation
predict( fitols$eq[[1]] )

## predicted values of the second equation
predict( fitols$eq[[2]] )
```

---

print.confint.systemfit

Print confidence intervals of coefficients

Description

This function prints the confidence intervals of the coefficients of the estimated equation system.

Usage

```
    ## S3 method for class 'confint.systemfit'
print( x, digits=3, ... )
```

Arguments

x	an object of type confint.systemfit.
digits	number of digits to print.
...	other arguments.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit, confint.systemfit and confint.systemfit.equation

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## calculate and print the confidence intervals
## of all coefficients
ci <- confint( fitols )
print( ci, digits=4 )

## calculate and print the confidence intervals
## of the coefficients of the second equation
ci2 <- confint( fitols$eq[[2]] )
print( ci2, digits=4 )
```

---

print.nlsystemfit *Print output of nlsystemfit estimation*

Description

These functions print a summary of the estimated equation system.

Usage

```
   ## S3 method for class 'nlsystemfit.system'
print( x, digits=6, ... )

   ## S3 method for class 'nlsystemfit.equation'
print( x, digits=6, ... )
```

Arguments

x	an object of class nlsystemfit.system or nlsystemfit.equation.
digits	number of digits to print.
...	not used by user.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>

See Also

nlsystemfit, summary.nlsystemfit.system

Examples

```
library( systemfit )
data( ppine )

hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev + h4*cr)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr + d4*ba  )
labels <- list( "height.growth", "diameter.growth" )
inst <- ~ tht + dbh + elev + cr + ba
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001, h4=0.08,
                  d0=-0.5, d1=0.009, d2=0.25, d3=0.005, d4=-0.02 )
model <- list( hg.formula, dg.formula )

model.ols <- nlsystemfit( "OLS", model, start.values, data=ppine, eqnlabels=labels )
print( model.ols )

model.3sls <- nlsystemfit( "3SLS", model, start.values, data=ppine,
                                    eqnlabels=labels, inst=inst )
print( model.3sls )
```

---

print.systemfit *Print results of systemfit estimation*

Description

These functions print a few results of the estimated equation system.

Usage

```
   ## S3 method for class 'systemfit'
print( x,
      digits = max( 3, getOption("digits") - 1 ), ... )

   ## S3 method for class 'systemfit.equation'
print( x,
      digits = max( 3, getOption("digits") - 1 ), ... )
```

Arguments

x	an object of class systemfit or systemfit.equation.
digits	number of digits to print.
...	other arguments.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>,
Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit, summary.systemfit

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## results of the whole system
print( fitols )

## results of the first equation
print( fitols$eq[[1]] )

## results of the second equation
print( fitols$eq[[2]] )
```

---

residuals.systemfit *Residuals of systemfit object*

Description

These functions extract the residuals from an object returned by systemfit.

Usage

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

   ## S3 method for class 'systemfit.equation'
residuals( object, na.rm = FALSE, ... )
```

Arguments

object	an object of class systemfit or systemfit.equation.
na.rm	a logical value indicating whether NA values (corresponding to observations that were not included in the estimation) should be removed from the vector of residuals before it is returned.
...	other arguments.

Value

residuals.systemfit returns a data.frame of residuals, where each column contains the residuals of one equation.

residuals.systemfit.equation returns a vector of residuals.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit, residuals

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## residuals of all equations
residuals( fitols )

## residuals of the first equation
residuals( fitols$eq[[1]] )

## residuals of the second equation
residuals( fitols$eq[[2]] )
```

---

se.ratio.systemfit *Ratio of the Standard Errors*

Description

se.ratio.systemfit returns a vector of the ratios of the standard errors of the predictions for two equations.

Usage

```
se.ratio.systemfit( resultsi, resultsj, eqni )
```

Arguments

`resultsi`	an object of type `systemfit`.
`resultsj`	an object of type `systemfit`.
`eqni`	index for equation to obtain the ratio of standard errors.

Value

`se.ratio` returns a vector of the standard errors of the ratios for the predictions between the predicted values in equation i and equation j.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>

References

Hasenauer, H; Monserud, R and T. Gregoire. (1998) Using Simultaneous Regression Techniques with Individual-Tree Growth Models. *Forest Science*. 44(1):87-95

See Also

`systemfit` and `correlation.systemfit`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
inst <- ~ income + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform 2SLS on each of the equations in the system
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta )

## print the results from the fits
print( fit2sls )
print( fit3sls )
print( "covariance of residuals used for estimation (from 2sls)" )
print( fit3sls$residCovEst )
print( "covariance of residuals" )
print( fit3sls$residCov )

## examine the improvement of 3SLS over 2SLS by computing
## the ratio of the standard errors of the estimates
improve.ratio <- se.ratio.systemfit( fit2sls, fit3sls, 2 )
```

```
print( "summary values for the ratio in the std. err. for the predictions" )
print( summary( improve.ratio ) )
```

---

summary.nlsystemfit *Summary of nlsystemfit estimation*

Description

These functions print a summary of the estimated equation system.

Usage

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

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

Arguments

object	an object of class `nlsystemfit.system` or `nlsystemfit.equation`.
...	not used by user.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>

See Also

`nlsystemfit`, `print.nlsystemfit.system`

Examples

```
library( systemfit )
data( ppine )

hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev + h4*cr)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr + d4*ba  )
labels <- list( "height.growth", "diameter.growth" )
inst <- ~ tht + dbh + elev + cr + ba
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001, h4=0.08,
                  d0=-0.5, d1=0.009, d2=0.25, d3=0.005, d4=-0.02 )
model <- list( hg.formula, dg.formula )

model.ols <- nlsystemfit( "OLS", model, start.values, data=ppine, eqnlabels=labels )
print( model.ols )

model.3sls <- nlsystemfit( "3SLS", model, start.values, data=ppine,
                                    eqnlabels=labels, inst=inst )
print( model.3sls )
```

summary.systemfit *Summary of systemfit estimation*

Description

These functions create and print summary results of the estimated equation system.

Usage

```
   ## S3 method for class 'systemfit'
summary( object, useDfSys = NULL,
      residCov = TRUE, equations = TRUE, ... )

   ## S3 method for class 'systemfit.equation'
summary( object, useDfSys = NULL, ... )

   ## S3 method for class 'summary.systemfit'
print( x,
      digits = max( 3, getOption("digits") - 1 ),
      residCov = x$printResidCov, equations = x$printEquations, ... )

   ## S3 method for class 'summary.systemfit.equation'
print( x,
      digits = max( 3, getOption("digits") - 1 ), ... )
```

Arguments

`object`	an object of class `systemfit` or `systemfit.equation`.
`x`	an object of class `summary.systemfit` or `summary.systemfit.equation`.
`useDfSys`	logical. Use the degrees of freedom of the whole system (in place of the degrees of freedom of the single equation) to calculate prob values for the t-test of individual coefficients. If it not specified (`NULL`), it is set to `TRUE` if restrictions on the coefficients are imposed and `FALSE` otherwise.
`digits`	number of digits to print.
`residCov`	logical. If `TRUE`, the residual correlation matrix, the residual covariance matrix, and its determinant are printed.
`equations`	logical. If `TRUE`, summary results of each equation are printed. If `FALSE`, just the coefficients are printed.
`...`	not used by user.

Value

Applying `summary` on an object of class `systemfit` returns a list of class `summary.systemfit`. Applying `summary` on an object of class `systemfit.equation` returns a list of class `summary.systemfit.equation`. An object of class `summary.systemfit` contains all results that belong to the whole system. This list contains one special object: eq. This is a list and contains objects of class summary.systemfit.equation. These objects contain the results that belong to each of the eatimated equations.

The objects of classes summary.systemfit and summary.systemfit.equation have the following components (elements that are marked with a $*$ are available only in objects of class summary.systemfit; elements that are marked with a $+$ are available only in objects of class summary.systemfit.equation):

method	estimation method.
residuals	residuals.
coefficients	a matrix with columns for the estimated coefficients, their standard errors, t-statistic and corresponding (two-sided) p-values.
df	degrees of freedom, a 2-vector, where the first element is the number of coefficients and the second element is the number of observations minus the number of coefficients.
coefCov	estimated covariance matrix of the coefficients.
call*	the matched call of systemfit.
ols.r.squared*	OLS R^2 value of the entire system.
mcelroy.r.squared*	McElroy's R^2 value for the system.
iter*	number of iteration steps (only if the estimation is iterated).
control*	list of control parameters used for the estimation.
residCov*	estimated residual covariance matrix.
residCovEst*	residual covariance matrix used for estimation (only SUR and 3SLS).
residCor*	correlation matrix of the residuals.
detResidCov*	determinant of residCov.
eqnLabel+	equation label.
eqnNo+	equation number.
terms+	the 'terms' object used for the respective equation.
r.squared+	R^2 value of the respective equation.
adj.r.squared+	adjusted R^2 value of the respective equation.
sigma+	estimated standard error of the residuals of the respective equation.
ssr+	sum of squared residuals of the respective equation.
printResidCov*	argument residCov.
printEquations*	argument equations.

Author(s)

Jeff D. Hamann <jeff.hamann@forestinformatics.com>,
Arne Henningsen <arne.henningsen@googlemail.com>

See Also

systemfit, print.systemfit

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
inst <- ~ income + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## results of the system
summary( fitols )

## short results of the system
summary( fitols, residCov = FALSE, equations = FALSE )

## results of the first equation
summary( fitols$eq[[1]] )

## results of the second equation
summary( fitols$eq[[2]] )
```

---

systemfit — *Linear Equation System Estimation*

Description

Fits a set of linear structural equations using Ordinary Least Squares (OLS), Weighted Least Squares (WLS), Seemingly Unrelated Regression (SUR), Two-Stage Least Squares (2SLS), Weighted Two-Stage Least Squares (W2SLS) or Three-Stage Least Squares (3SLS).

Usage

```
systemfit( formula, method = "OLS",
           inst=NULL, data=list(),
           restrict.matrix = NULL, restrict.rhs = NULL, restrict.regMat = NULL,
           pooled = FALSE, control = systemfit.control( ... ), ... )
```

Arguments

`formula` an object of class `formula` (for single-equation models) or (typically) a list of objects of class `formula` (for multiple-equation models); if argument `data` is of class `pdata.frame` (created with `pdata.frame()`), this argument must be a single object of class `formula` that represents the formula to be estimated for all individuals.

`method` the estimation method, one of "OLS", "WLS", "SUR", "2SLS", "W2SLS", or "3SLS" (see details); iterated estimation methods can be specified by setting control parameter `maxiter` larger than 1 (e.g. 500).

inst
: one-sided model formula specifying the instrumental variables (including exogenous explanatory variables) or a list of one-sided model formulas if different instruments should be used for the different equations (only needed for 2SLS, W2SLS, and 3SLS estimations).

data
: an optional data frame containing the variables in the model. By default the variables are taken from the environment from which systemfit is called.

restrict.matrix
: an optional j x k matrix to impose linear restrictions on the coefficients by `restrict.matrix` * b = `restrict.rhs` (j = number of restrictions, k = number of all coefficients, b = vector of all coefficients) or a character vector giving the restrictions in symbolic form (see documentation of `linearHypothesis` in package "car" for details). The number and the names of the coefficients can be obtained by estimating the system without restrictions and applying the `coef` method to the returned object.

restrict.rhs
: an optional vector with j elements to impose linear restrictions (see `restrict.matrix`); default is a vector that contains j zeros.

restrict.regMat
: an optional matrix to impose restrictions on the coefficients by post-multiplying the regressor matrix with this matrix (see details).

control
: list of control parameters. The default is constructed by the function `systemfit.control`. See the documentation of `systemfit.control` for details.

pooled
: logical, restrict coefficients to be equal in all equations (only for panel-like data).

...
: arguments passed to `systemfit.control`.

Details

The estimation of systems of equations with unequal numbers of observations has not been thoroughly tested yet. Currently, `systemfit` calculates the residual covariance matrix only from the residuals/observations that are available in all equations.

If argument `data` is of class `pdata.frame` (created with `pdata.frame()` and thus, contains panel data in long format), argument `formula` must be a single equation that is applied to all individuals. In this case, argument `pooled` specifies whether the coefficients are restricted to be equal for all individuals.

If argument `restrict.regMat` is specified, the regressor matrix X is post-multiplied by this matrix: $X^* = X\cdot$ `restrict.regMat`. Then, this modified regressor matrix X^* is used for the estimation of the coefficient vector b^*. This means that the coefficients of the original regressors (X), vector b, can be represented by $b =$ `restrict.regMat` $\cdot b^*$. If `restrict.regMat` is a non-singular quadratic matrix, there are no restrictions on the coefficients imposed, but the coefficients b^* are linear combinations of the original coefficients b. If `restrict.regMat` has less columns than rows, linear restrictions are imposed on the coefficients b. However, imposing linear restrictions by the `restrict.regMat` matrix is less flexible than by providing the matrix `restrict.matrix` and the vector `restrict.rhs`. The advantage of imposing restrictions on the coefficients by the matrix `restrict.regMat` is that the matrix, which has to be inverted during the estimation, gets smaller by this procedure, while it gets larger if the restrictions are imposed by `restrict.matrix` and `restrict.rhs`.

In the context of multi-equation models, the term “weighted” in “weighted least squares” (WLS) and “weighted two-stage least squares” (W2SLS) means that the *equations* might have different weights and *not* that the *observations* have different weights.

It is important to realize the limitations on estimating the residuals covariance matrix imposed by the number of observations T in each equation. With g equations we estimate $g * (g + 1)/2$ elements using $T * g$ observations total. Beck and Katz (1995,1993) discuss the issue and the resulting overconfidence when the ratio of T/g is small (e.g. 3). Even for $T/g = 5$ the estimate is unstable both numerically and statistically and the 95 approximately $[0.5 * \sigma^2, 3 * \sigma^2]$, which is inadequate precision if the covariance matrix will be used for simulation of asset return paths either for investment or risk management decisions. For a starter on models with large cross-sections see Reichlin (2002). [This paragraph has been provided by Stephen C. Bond – Thanks!]

Value

`systemfit` returns a list of the class `systemfit` and contains all results that belong to the whole system. This list contains one special object: "eq". It is a list and contains one object for each estimated equation. These objects are of the class `systemfit.equation` and contain the results that belong only to the regarding equation.

The objects of the class `systemfit` and `systemfit.equation` have the following components (the elements of the latter are marked with an asterisk ($*$)):

- `call` the matched call.
- `method` estimation method.
- `rank` total number of linear independent coefficients = number of coefficients minus number of linear restrictions.
- `df.residual` degrees of freedom of the whole system.
- `iter` number of iteration steps.
- `coefficients` vector of all estimated coefficients.
- `coefCov` estimated covariance matrix of `coefficients`.
- `residCov` estimated residual covariance matrix.
- `residCovEst` residual covariance matrix used for estimation (only WLS, W2SLS, SUR and 3SLS).
- `restrict.matrix` the restriction matrix.
- `restrict.rhs` the restriction vector.
- `restrict.regMat` matrix used to impose restrictions on the coefficients by post-multiplying the regressor matrix with this matrix.
- `control` list of control parameters used for the estimation.
- `panelLike` logical. Was this an analysis with panel-like data?

elements of the class systemfit.eq

- `eq` a list that contains the results that belong to the individual equations.
- `eqnLabel*` the label of this equation.

eqnNo*	the number of this equation.
terms*	the 'terms' object used for the ith equation.
inst*	instruments of the ith equation (only 2SLS, W2SLS, and 3SLS).
termsInst*	the 'terms' object of the instruments used for the ith equation (only 2SLS, W2SLS, and 3SLS).
rank*	number of linear independent coefficients in the ith equation (differs from the number of coefficients only if there are restrictions that are not cross-equation).
nCoef.sys*	total number of coefficients in all equations.
rank.sys*	total number of linear independent coefficients in all equations.
df.residual*	degrees of freedom of the ith equation.
df.residual.sys*	degrees of freedom of the whole system.
coefficients*	estimated coefficients of the ith equation.
covb*	estimated covariance matrix of `coefficients`.
model*	if requested (the default), the model frame of the ith equation.
modelInst*	if requested (the default), the model frame of the instrumental variables of the ith equation (only 2SLS, W2SLS, and 3SLS).
x*	if requested, the model matrix of the ith equation.
y*	if requested, the response of the ith equation.
z*	if requested, the matrix of instrumental variables of the ith equation (only 2SLS, W2SLS, and 3SLS).
fitted.values*	vector of fitted values of the ith equation.
residuals*	vector of residuals of the ith equation.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>,
Jeff D. Hamann <jeff.hamann@forestinformatics.com>

References

Beck, N.; J.N. Katz (1995) What to do (and not to do) with Time-Series Cross-Section Data, *The American Political Science Review*, 89, pp. 634-647.

Beck, N.; J.N. Katz; M.R. Alvarez; G. Garrett; P. Lange (1993) Government Partisanship, Labor Organization, and Macroeconomic Performance: a Corrigendum, *American Political Science Review*, 87, pp. 945-48.

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

Judge, George G.; W. E. Griffiths; R. Carter Hill; Helmut Luetkepohl and Tsoung-Chao Lee (1985) *The Theory and Practice of Econometrics, Second Edition*, Wiley.

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

Reichlin, L. (2002) *Factor models in large cross-sections of time series*, Working Paper, ECARES and CEPR.

Schmidt, P. (1990) *Three-Stage Least Squares with different Instruments for different equations*, Journal of Econometrics 43, p. 389-394.

Theil, H. (1971) *Principles of Econometrics*, Wiley, New York.

See Also

lm and nlsystemfit

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## OLS estimation
fitols <- systemfit( system, data=Kmenta )
print( fitols )

## OLS estimation with 2 restrictions
Rrestr <- matrix(0,2,7)
Rrestr[1,3] <-  1
Rrestr[1,7] <- -1
Rrestr[2,2] <- -1
Rrestr[2,5] <-  1
qrestr <- c( 0, 0.5 )
fitols2 <- systemfit( system, data = Kmenta,
                      restrict.matrix = Rrestr, restrict.rhs = qrestr )
print( fitols2 )

## OLS estimation with the same 2 restrictions in symbolic form
restrict <- c( "demand_income - supply_trend = 0",
   "- demand_price + supply_price = 0.5" )
fitols2b <- systemfit( system, data = Kmenta, restrict.matrix = restrict )
print( fitols2b )

# test whether both restricted estimators are identical
all.equal( coef( fitols2 ), coef( fitols2b ) )

## OLS with restrictions on the coefficients by modifying the regressor matrix
## with argument restrict.regMat
modReg <- matrix( 0, 7, 6 )
colnames( modReg ) <- c( "demIntercept", "demPrice", "demIncome",
   "supIntercept", "supPrice2", "supTrend" )
modReg[ 1, "demIntercept" ] <- 1
modReg[ 2, "demPrice" ]     <- 1
modReg[ 3, "demIncome" ]    <- 1
modReg[ 4, "supIntercept" ] <- 1
modReg[ 5, "supPrice2" ]    <- 1
modReg[ 6, "supPrice2" ]    <- 1
modReg[ 7, "supTrend" ]     <- 1
fitols3 <- systemfit( system, data = Kmenta, restrict.regMat = modReg )
```

```
print( fitols3 )


## iterated SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta, maxit = 100 )
print( fitsur )

## 2SLS estimation
inst <- ~ income + farmPrice + trend
fit2sls <- systemfit( system, "2SLS", inst = inst, data = Kmenta )
print( fit2sls )

## 2SLS estimation with different instruments in each equation
inst1 <- ~ income + farmPrice
inst2 <- ~ income + farmPrice + trend
instlist <- list( inst1, inst2 )
fit2sls2 <- systemfit( system, "2SLS", inst = instlist, data = Kmenta )
print( fit2sls2 )

## 3SLS estimation with GMM-3SLS formula
inst <- ~ income + farmPrice + trend
fit3sls <- systemfit( system, "3SLS", inst = inst, data = Kmenta,
   method3sls = "GMM" )
print( fit3sls )


## Examples how to use systemfit() with panel-like data
## Repeating the SUR estimations in Greene (2003, p. 351)
data( "GrunfeldGreene" )
if( requireNamespace( 'plm', quietly = TRUE ) ) {
library( "plm" )
GGPanel <- pdata.frame( GrunfeldGreene, c( "firm", "year" ) )
formulaGrunfeld <- invest ~ value + capital
# SUR
greeneSur <- systemfit( formulaGrunfeld, "SUR",
   data = GGPanel, methodResidCov = "noDfCor" )
summary( greeneSur )
# SUR Pooled
greeneSurPooled <- systemfit( formulaGrunfeld, "SUR",
   data = GGPanel, pooled = TRUE, methodResidCov = "noDfCor",
   residCovWeighted = TRUE )
summary( greeneSurPooled )
}

## Further examples are in the documentation to the data sets
## 'KleinI' and 'GrunfeldGreene'.
```

---

systemfit.control *Create list of control parameters for systemfit*

Description

Create a list of control pararameters for function systemfit. All control parameters that are not passed to this function are set to default values.

Usage

```
systemfit.control(
      maxiter = 1,
      tol = 1e-5,
      methodResidCov = "geomean",
      centerResiduals = FALSE,
      residCovRestricted = TRUE,
      residCovWeighted = FALSE,
      method3sls = "GLS",
      singleEqSigma = NULL,
      useMatrix = TRUE,
      solvetol = .Machine$double.eps,
      model = TRUE,
      x = FALSE,
      y = FALSE,
      z = FALSE )
```

Arguments

`maxiter`	maximum number of iterations for WLS, SUR, W2SLS and 3SLS estimations.
`tol`	tolerance level indicating when to stop the iteration (only WLS, SUR, W2SLS and 3SLS estimations).
`methodResidCov`	method for calculating the estimated residual covariance matrix, one of "noDfCor", "geomean", "max", or "Theil" (see details).
`centerResiduals`	logical. Subtract the means from the residuals of each equation before calculating the estimated residual covariance matrix.
`residCovRestricted`	logical. If 'FALSE' the residual covariance matrix for a WLS, SUR, W2SLS, or 3SLS estimation is obtained from an unrestricted first-step estimation.
`residCovWeighted`	logical. If 'TRUE' the residual covariance matrix for a SUR or 3SLS estimation is obtained from a WLS or W2SLS estimation.
`method3sls`	method for calculating the 3SLS estimator, one of "GLS", "IV", "GMM", "Schmidt", or "EViews" (see details).
`singleEqSigma`	logical. use different σ^2s for each single equation to calculate the covariance matrix and the standard errors of the coefficients (only OLS and 2SLS)? If `singleEqSigma` is `NULL`, it is automatically determined: It is set to `TRUE`, if restrictions on the coefficients are imposed, and it is set to `FALSE` otherwise.
`useMatrix`	logical. Use package `Matrix` for matrix calculations?

solvetol — tolerance level for detecting linear dependencies when inverting a matrix or calculating a determinant (see solve and det).

model, x, y, z — logical. If 'TRUE' the corresponding components of the fit (the model frame, the model matrix, the response, and the matrix of instruments, respectively) are returned.

Details

If the estimation is iterated (WLS, SUR, W2SLS or 3SLS estimation with maxiter>1), the convergence criterion is

$$\sqrt{\frac{\sum_i (b_{i,g} - b_{i,g-1})^2}{\sum_i b_{i,g-1}^2}} < \texttt{tol}$$

($b_{i,g}$ is the ith coefficient of the gth iteration step).

The method for calculating the estimated covariance matrix of the residuals ($\hat{\Sigma}$) can be one of the following (see Judge et al., 1985, p. 469):
if methodResidCov='noDfCor':

$$\hat{\sigma}_{ij} = \frac{\hat{e}_i' \hat{e}_j}{T}$$

if methodResidCov='geomean':

$$\hat{\sigma}_{ij} = \frac{\hat{e}_i' \hat{e}_j}{\sqrt{(T - k_i) * (T - k_j)}}$$

if methodResidCov='Theil':

$$\hat{\sigma}_{ij} = \frac{\hat{e}_i' \hat{e}_j}{T - k_i - k_j + tr[X_i (X_i' X_i)^{-1} X_i' X_j (X_j' X_j)^{-1} X_j']}$$

if methodResidCov='max':

$$\hat{\sigma}_{ij} = \frac{\hat{e}_i' \hat{e}_j}{T - \max(k_i, k_j)}$$

If $i = j$, the formulas 'geomean', 'Theil', and 'max' are equal. All these three formulas yield unbiased estimators for the diagonal elements of the residual covariance matrix. If $i \neq j$, only formula 'Theil' yields an unbiased estimator for the residual covariance matrix, but it is not neccessarily positive semidefinit. Thus, it is doubtful whether formula 'Theil' is really superior to formula 'noDfCor' (Theil, 1971, p. 322).

The methods for calculating the 3SLS estimator lead to identical results if the same instruments are used in all equations. If different instruments are used in the different equations, only the GMM-3SLS estimator ("GMM") and the 3SLS estimator proposed by Schmidt (1990) ("Schmidt") are consistent, whereas "GMM" is efficient relative to "Schmidt" (see Schmidt, 1990).

If residCovWeighted is TRUE, systemfit does a OLS or 2SLS estimation in a first step. It uses the residuals from the first-step estimation to calculate the residual covariance matrix that is used in a second-step WLS or W2SLS estimation. Then, it uses the residuals from the second-step estimation to calculate the residual covariance matrix that is used in a final SUR or 3SLS estimation. This three-step method is the default method of command "TSCS" in the software LIMDEP that carries out "SUR" estimations in which all coefficient vectors are constrained to be equal (personal information from W.H. Greene, 2006/02/16). If no cross-equation restrictions are imposed, residCovWeighted has no effect on the estimation results.

Value

A list of the above components.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

References

Judge, George G.; W. E. Griffiths; R. Carter Hill; Helmut Luetkepohl and Tsoung-Chao Lee (1985) *The Theory and Practice of Econometrics, Second Edition*, Wiley.

Schmidt, P. (1990) *Three-Stage Least Squares with different Instruments for different equations*, Journal of Econometrics 43, p. 389-394.

Theil, H. (1971) *Principles of Econometrics*, Wiley, New York.

See Also

systemfit

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
eqSystem <- list( demand = eqDemand, supply = eqSupply )

## SUR estimation: calculation of residual covariance
## matrix without correction for degrees of freedom
fitsur <- systemfit( eqSystem, "SUR", data = Kmenta,
    control = systemfit.control( methodResidCov = "noDfCor" ) )
print( fitsur )
```

---

terms.systemfit — *Model Terms of systemfit Objects*

Description

This method extracts the model terms from fitted objects returned by systemfit.

Usage

```
   ## S3 method for class 'systemfit'
terms( x, ... )
   ## S3 method for class 'systemfit.equation'
terms( x, ... )
```

Arguments

`x`	an object of class `systemfit`.
`...`	currently not used.

Value

`terms.systemfit.equation` returns the model terms of a single equation of a `systemfit` object. `terms.systemfit.equation` returns a list of model terms: one model term object for each equation of the `systemfit` object.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

`systemfit`, `terms`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform a SUR estimation
fitsur <- systemfit( system, "SUR", data = Kmenta )

## model terms of the second equation
terms( fitsur$eq[[ 2 ]] )

## all model terms of the system
terms( fitsur )
```

---

vcov.systemfit — *Variance covariance matrix of coefficients*

Description

These functions extract the variance covariance matrix of the coefficients from an object returned by `systemfit`.

Usage

```
   ## S3 method for class 'systemfit'
vcov( object, modified.regMat = FALSE, ... )

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

Arguments

`object` an object of class `systemfit` or `systemfit.equation`.

`modified.regMat` logical. If `TRUE`, the covariance matrix of the coefficients of the modified regressor matrix (original regressor matrix post-multiplied by `restrict.regMat`) rather than the covariance matrix of the coefficients of the original regressor matrix is returned.

`...` other arguments.

Value

`vcov.systemfit` returns the variance covariance matrix of all estimated coefficients.

Author(s)

Arne Henningsen <arne.henningsen@googlemail.com>

See Also

`systemfit`, `vcov`

Examples

```
data( "Kmenta" )
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list( demand = eqDemand, supply = eqSupply )

## perform OLS on each of the equations in the system
fitols <- systemfit( system, data = Kmenta )

## variance covariance matrix of all coefficients
vcov( fitols )

## variance covariance matrix of the coefficients in the first equation
vcov( fitols$eq[[1]] )

## variance covariance matrix of the coefficients in the second equation
vcov( fitols$eq[[2]] )

## estimation with restriction by modifying the regressor matrix
modReg <- matrix( 0, 7, 6 )
colnames( modReg ) <- c( "demIntercept", "demPrice", "demIncome",
   "supIntercept", "supPrice2", "supTrend" )
modReg[ 1, "demIntercept" ] <- 1
modReg[ 2, "demPrice" ]     <- 1
modReg[ 3, "demIncome" ]    <- 1
modReg[ 4, "supIntercept" ] <- 1
modReg[ 5, "supPrice2" ]    <- 1
modReg[ 6, "supPrice2" ]    <- 1
modReg[ 7, "supTrend" ]     <- 1
```

```
fitsur <- systemfit( system, "SUR", data = Kmenta, restrict.regMat = modReg )
vcov( fitsur, modified.regMat  = TRUE )
vcov( fitsur )
```

Index