Package ‘diseq’

April 28, 2020

Title Estimation Methods for Markets in Disequilibrium

Version 0.0.11

Date 2020-04-23

Description Provides estimation methods for markets in equilibrium and disequilibrium. Specifically, it supports the estimation of an equilibrium and four disequilibrium models with both correlated and independent shocks. It also provides post-estimation analysis tools, such as aggregation and marginal effects calculations. The estimation methods are based on full information maximum likelihood techniques given in Maddala and Nelson (1974) <doi:10.2307/1914215>. They are implemented using the analytic derivative expressions calculated in Karapanagiotis (2020) <doi:10.2139/ssrn.3525622>. The equilibrium estimation constitutes a special case of a system of simultaneous equations. The disequilibrium models, instead, replace the market clearing condition with a short side rule and allow for different specifications of price dynamics.

Language en-US

URL https://github.com/pi-kappa-devel/diseq/

BugReports https://github.com/pi-kappa-devel/diseq/issues

Depends R (>= 3.5.0)

Imports bbmle (>= 1.0.20), dplyr (>= 0.7.6), magrittr (>= 1.5), MASS (>= 7.3-50), methods, rlang (>= 0.2.1), systemfit (>= 1.1), tibble (>= 1.4.2), tidyr (>= 1.0.2)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Suggests ggplot2 (>= 3.0.0), knitr (>= 1.20), numDeriv (>= 2016.8.1.1), rmarkdown (>= 1.10), testthat (>= 2.0.0)

VignetteBuilder knitr
Collate 'equation_base.R' 'system_base.R' 'equation_basic.R'
    'system_basic.R' 'derivatives_basic.R'
    'equation_deterministic_adjustment.R'
    'system_deterministic_adjustment.R'
    'derivatives_deterministic_adjustment.R'
    'equation_directional.R' 'system_directional.R'
    'derivatives_directional.R' 'system_fiml.R'
    'derivatives_fiml.R' 'equation_stochastic_adjustment.R'
    'system_stochastic_adjustment.R'
    'derivatives_stochastic_adjustment.R' 'diseq.R'
    'model_logger.R' 'model_base.R' 'diseq_base.R' 'diseq_basic.R'
    'diseq_deterministic_adjustment.R' 'diseq_directional.R'
    'diseq_stochastic_adjustment.R' 'eq_base.R' 'eq_2sls.R'
    'eq_fiml.R' 'model_simulation.R'

NeedsCompilation no

Author Pantelis Karapanagiotis [aut, cre]
    (https://orcid.org/0000-0001-9871-1908)

Maintainer Pantelis Karapanagiotis <pikappa.devel@gmail.com>

Repository CRAN

Date/Publication 2020-04-28 10:30:02 UTC

R topics documented:

diseq .......................................................... 3
diseq_base-class ............................................. 4
diseq_basic-class ............................................. 4
diseq_deterministic_adjustment-class ......................... 5
diseq_directional-class ....................................... 5
diseq_stochastic_adjustment-class ............................ 6
equation_base-class .......................................... 7
equation_basic-class .......................................... 7
equation_deterministic_adjustment-class ....................... 7
equation_directional-class .................................... 8
equation_stochastic_adjustment-class ........................ 8
eq_2sls-class .................................................. 8
eq_base-class .................................................. 9
eq_fiml-class .................................................. 9
estimate .......................................................... 10
get_aggregate_demand .......................................... 11
get_aggregate_supply .......................................... 12
get_demanded_quantities ....................................... 13
get_demand_descriptives ...................................... 14
get_marginal_effect_at_mean .................................. 15
get_mean_marginal_effect ..................................... 16
get_model_description ......................................... 17
get_normalized_shortages ...................................... 18
The `diseq` package provides tools to estimate and analyze an equilibrium and four disequilibrium models. The equilibrium model can be estimated with either two-stage least squares or with full information maximum likelihood. The methods are asymptotically equivalent. The disequilibrium models are estimated using full information maximum likelihood. All maximum likelihood models can be estimated both with independent and correlated demand and supply shocks. The disequilibrium estimation is based on Maddala and Nelson (1974). The package is using the expressions of the gradients of the likelihoods derived in Karapanagiotis (2020).

## Usage

The easiest way to get accustomed with the functionality of the package is to check the accompanying vignettes and the README file. These can be found in the following links:

- basic_usage vignette("basic_usage",package = "diseq")
- equilibrium_assessment vignette("equilibrium_assessment",package = "diseq")

## Market model classes:

The model hierarchy is described in the README file. See the documentation of the classes for initialization details.

### Equilibrium model classes:

- eq_2sls Equilibrium two-stage least square model
### Disequilibrium model classes:

- **diseq_basic** Disequilibrium model with a only basic short side rule
- **diseq_directional** Disequilibrium model with directional sample separation
- **diseq_deterministic_adjustment** Disequilibrium model with deterministic price dynamics
- **diseq_stochastic_adjustment** Disequilibrium model with stochastic price dynamics

---

**diseq_base-class**  
*Equilibrium model base class*

**Description**

Equilibrium model base class

---

**diseq_basic-class**  
*Basic disequilibrium model with unknown sample separation.*

**Description**

The basic disequilibrium model consists of three equations. Two of them are the demand and supply equations. In addition, the model replaces the market clearing condition with the short side rule. The model is estimated using full information maximum likelihood.

\[
D_{nt} = X_{d,nt} \beta_d + u_{d,nt},
\]

\[
S_{nt} = X_{s,nt} \beta_s + u_{s,nt},
\]

\[
Q_{nt} = \min\{D_{nt}, S_{nt}\}.
\]

**Examples**

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)
```

# initialize the model
model <- new(
  "diseq_basic", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)
Disequilibrium model with deterministic price dynamics.

Description

The disequilibrium model with deterministic price adjustment consists of four equations. The two market equations, the short side rule and price evolution equation. The first two equations are stochastic. The price equation is deterministic. The sample is separated based on the sign of the price changes as in the diseq_directional model. The model is estimated using full information maximum likelihood.

\[
D_{nt} = X_{d,nt}'\beta_d + P_{nt}\alpha_d + u_{d,nt},
\]

\[
S_{nt} = X_{s,nt}'\beta_s + P_{nt}\alpha_s + u_{s,nt},
\]

\[
Q_{nt} = \min\{D_{nt}, S_{nt}\},
\]

\[
\Delta P_{nt} = \frac{1}{\gamma} (D_{nt} - S_{nt}).
\]

Examples

```r
simulated_data <- simulate_model_data(
  "diseq_deterministic_adjustment", 500, 3, # model type, observed entities and time points
  -0.9, 8.9, c(0.03, -0.02), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(0.05, 0.02), # supply coefficients
  1.4 # price adjustment coefficient
)

# initialize the model
model <- new(
  "diseq_deterministic_adjustment", # model type
  c("id", "date"), "date", "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)
```

Directional disequilibrium model with sample separation.
**Description**

The directional disequilibrium model consists of three equations and a separation rule. The market is described by a linear demand, a linear supply equation and the short side rule. The separation rule splits the sample into regimes of excess supply and excess demand. If a price change is positive at the time point of the observation, then the observation is classified as being in an excess demand regime. Otherwise, it is assumed that it represents an excess supply state. The model is estimated using full information maximum likelihood.

\[
\begin{align*}
D_{nt} &= X'_{d,nt} \beta_d + u_{d,nt}, \\
S_{nt} &= X'_{s,nt} \beta_s + u_{s,nt}, \\
Q_{nt} &= \min\{D_{nt}, S_{nt}\}, \\
\Delta P_{nt} \geq 0 &\Rightarrow D_{nt} \geq S_{nt}.
\end{align*}
\]

**Examples**

```r
 simulated_data <- simulate_model_data(
   "diseq_directional", 500, 3, # model type, observed entities, observed time points
   -0.2, 4.3, c(0.03, 0.02), c(0.03, 0.01), # demand coefficients
   0.0, 4.0, c(0.03), c(0.05, 0.02) # supply coefficients
)

# in the directional model prices cannot be included in both demand and supply
model <- new(
   "diseq_directional", # model type
   c("id", "date"), "date", "Q", "P", # keys, time point, quantity, and price variables
   "P + Xd1 + Xd2 + X1 + X2", "Xs1 + X1 + X2", # equation specifications
   simulated_data, # data
   use_correlated_shocks = TRUE # allow shocks to be correlated
)
```

---

**diseq_stochastic_adjustment-class**

* Disequilibrium model with stochastic price dynamics.

**Description**

The disequilibrium model with stochastic price adjustment is described by a system of four equations. Three of them form a stochastic linear system of market equations coupled with a stochastic price evolution equation. The fourth equation is the short side rule. In contrast to the deterministic counterpart, the model does not impose any separation rule on the sample. It is estimated using full information maximum likelihood.

\[
D_{nt} = X'_{d,nt} \beta_d + P_{nt} \alpha_d + u_{d,nt},
\]

---
\[ S_{nt} = X'_{s,nt}\beta_s + P_{nt}\alpha_s + u_{s,nt}, \]
\[ Q_{nt} = \min\{D_{nt}, S_{nt}\}, \]
\[ \Delta P_{nt} = \frac{1}{\gamma} (D_{nt} - S_{nt}) + X'_p,nt\beta_p + u_{p,nt}. \]

**Examples**

```r
simulated_data <- simulate_model_data(
  "diseq_stochastic_adjustment", 500, 3, # model type, observed entities and time points
  -0.1, 9.8, c(0.3, -0.2), c(0.6, 0.1), # demand coefficients
  0.1, 5.1, c(0.9), c(-0.5, 0.2), # supply coefficients
  1.4, 3.1, c(0.8) # price adjustment coefficient
)

# initialize the model
model <- new(
  "diseq_stochastic_adjustment", # model type
  c("id", "date"), "date", "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", "Xp1", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)
```

**equation_base-class**  
*Equation base class*

**Description**  
Equation base class

**equation_basic-class**  
*Basic disequilibrium model equation class*

**Description**  
Basic disequilibrium model equation class

**equation_deterministic_adjustment-class**  
*Deterministic adjustment disequilibrium model equation class*

**Description**  
Deterministic adjustment disequilibrium model equation class
eq_2sls-class

Equilibrium model estimated using 2-stage least squares.

Description

In the first stage, prices are regressed on remaining controls from both the demand and supply
equations. In the second stage, the demand and supply equation is estimated using the fitted prices
instead of the observed. A necessary identification condition is that there is at least one control that
is exclusively part of the demand and one control that is exclusively part of the supply equation.

\[
D_{nt} = X'_{d,nt} \beta_d + P_{nt} \alpha_d + u_{d,nt},
\]

\[
S_{nt} = X'_{s,nt} \beta_s + P_{nt} \alpha_s + u_{s,nt},
\]

\[
Q_{nt} = D_{nt} = S_{nt}.
\]

Slots

first_stage_model  An estimated first stage equation of type \texttt{lm}.

system_model  An estimated system of market equations of type \texttt{systemfit}.
Examples

```r
simulated_data <- simulate_model_data(
  "eq_2sls", 500, 3, # model type, observed entities and time points
  -0.9, 14.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 3.2, c(0.3), c(0.5, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "eq_2sls", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data # data
)
```

---

**eq_base-class**

*Equilibrium model base class*

**Description**

Equilibrium model base class

---

**eq_fiml-class**

*Equilibrium model estimated using full-information maximum likelihood.*

**Description**

The equilibrium model consists of thee equations. The demand, the supply and the market clearing equations. This model is estimated using full information maximum likelihood.

\[
D_{nt} = X_{d,nt}'\beta_d + P_{nt}\alpha_d + u_{d,nt},
\]

\[
S_{nt} = X_{s,nt}'\beta_s + P_{nt}\alpha_s + u_{s,nt},
\]

\[
Q_{nt} = D_{nt} = S_{nt}.
\]

**Examples**

```r
simulated_data <- simulate_model_data(
  "eq_fiml", 500, 3, # model type, observed entities and time points
  -0.9, 14.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 3.2, c(0.3), c(0.5, 0.02) # supply coefficients
)

# initialize the model
model <- new(
```
"eq_fiml", # model type
c("id", "date"). "Q", "P", # keys, quantity, and price variables
"P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
simulated_data, # data
use_correlated_shocks = TRUE # allow shocks to be correlated
)

estimate

Model estimation.

Description

With the exception of eq_2sls all model are estimated by maximum likelihood. The likelihood estimation is using mle2. If no starting values are provided, the function uses linear regression estimates as initializing values. The default optimization method is BFGS. For other alternatives see mle2. The eq_2sls is estimated using two stage least squares. The implementation is based on systemfit.

Usage

estimate(object, ...)

## S4 method for signature 'diseq_base'
estimate(object, use_numerical_hessian = TRUE, ...)

## S4 method for signature 'eq_2sls'
estimate(object, ...)

## S4 method for signature 'eq_fiml'
estimate(object, ...)

Arguments

object

A model object.

...

Named parameter used in the model’s estimation. These are passed further down to the estimation call. For the eq_2sls model, the parameters a passed to systemfit. For the rest of the models, the parameters are passed to mle2.

use_numerical_hessian

If true, the variance-covariance matrix is calculated using the numerically approximated Hessian. Calculated Hessians are available for the basic and directional models.

Value

The object that holds the estimation result.
**get_aggregate_demand**  

**Demand aggregation.**

**Description**

Calculates the sample's aggregate demand at the passed set of parameters.

**Usage**

```
get_aggregate_demand(object, parameters)
```

## S4 method for signature 'model_base'

get_aggregate_demand(object, parameters)

**Arguments**

- **object**  
  A model object.

- **parameters**  
  A vector of model's parameters.

**Functions**

- **estimate, diseq_base-method**: Disequilibrium model estimation.
- **estimate, eq_2sls-method**: Equilibrium model estimation.
- **estimate, eq_fiml-method**: Equilibrium model estimation.

**Examples**

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3,  # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01),  # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02)  # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic",  # model type
  c("id", "date"), "Q", "P",  # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2",  # equation specifications
  simulated_data,  # data
  use_correlated_shocks = TRUE  # allow shocks to be correlated
)

# estimate the model object (by default the maximum optimization is using BFGS)
est <- estimate(model)

# estimate the model by specifying the optimization details that are passed to the optimizer.
est <- estimate(model, control = list(reltol = 1e-4), method = "BFGS")
```
Value

The sum of the demanded quantities evaluated at the given parameters.

See Also

get_demanded_quantities

Examples

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)

# estimate the model object
est <- estimate(model)

# get estimated aggregate demand
get_aggregate_demand(model, est@coef)
```

get_aggregate_supply

Supply aggregation.

Description

Calculates the sample’s aggregate supply at the passed set of parameters.

Usage

```r
get_aggregate_supply(object, parameters)
```

## S4 method for signature 'model_base'

```r
get_aggregate_supply(object, parameters)
```

Arguments

- **object**: A model object.
- **parameters**: A vector of model’s parameters.
get_demanded_quantities

Value

The sum of the supplied quantities evaluated at the given parameters.

See Also

get_supplied_quantities

Examples

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)

# estimate the model object
est <- estimate(model)

# get estimated aggregate supply
get_aggregate_supply(model, est@coef)
```

---

demand_quanaties

Demanded quantities.

Description

Calculates the demanded quantity for each observation.

Usage

```r
get_demanded_quantities(object, parameters)
```

```r
## S4 method for signature 'model_base'
get_demanded_quantities(object, parameters)
```
get_demand_descriptives

Arguments

object
A model object.

parameters
A vector of model's parameters.

Value

A vector with the demanded quantities evaluated at the given parameter vector.

See Also

get_aggregate_demand

Examples

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)

# estimate the model object
est <- estimate(model)

# get estimated demanded quantities
demq <- get_demanded_quantities(model, est@coef)
```

Description

Calculates and returns basic descriptive statistics for the model's demand data. Factor variables are excluded from the calculations.
get_marginal_effect_at_mean

Usage

get_demand_descriptives(object)

## S4 method for signature 'model_base'
get_demand_descriptives(object)

Arguments

object A model object.

Value

A data tibble containing descriptive statistics.

get_marginal_effect_at_mean

Marginal effects at the mean.

Description

Returns the estimated marginal effects evaluated at the mean of a variable.

Usage

get_marginal_effect_at_mean(object, estimation, variable)

## S4 method for signature 'diseq_base'
get_marginal_effect_at_mean(object, estimation, variable)

Arguments

object A disequilibrium model object.
estimation A model estimation object (i.e. a mle2 object).
variable Variable name for which the effect is calculated.

Value

The marginal effect of the passed variable evaluated at the estimated mean.

Examples

simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)
get_mean_marginal_effect

Mean marginal effects

Description

Returns the average estimated marginal effect a variable.

Usage

get_mean_marginal_effect(object, estimation, variable)

## S4 method for signature 'diseq_base'
get_mean_marginal_effect(object, estimation, variable)

Arguments

object A disequilibrium model object.
estimation A model estimation object (i.e. a mle2 object).
variable Variable name for which the effect is calculated.

Value

The mean of the estimated marginal effects of the passed variable.

Examples

simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
  )

# initialize the model
model <- new(
  "diseq_basic", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)

# estimate a model object
est <- estimate(model)

# get the marginal effects at the mean of variable "X1"
get_marginal_effect_at_mean(model, est, "X1")
Model description.

Description

A unique identifying string for the model.

Usage

get_model_description(object)

## S4 method for signature 'model_base'
get_model_description(object)

Arguments

object A model object.

Value

A string representation of the model.
get_normalized_shortages

Normalized shortages.

Description

Returns the shortages normalized by the variance of the difference of the shocks at a given point.

Usage

get_normalized_shortages(object, parameters)

## S4 method for signature 'diseq_base'
get_normalized_shortages(object, parameters)

Arguments

object A disequilibrium model object.
parameters A vector of parameters at which the shortages are evaluated.

Value

A vector with the estimated normalized shortages.

Examples

simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)

# estimate a model object
est <- estimate(model)

# get estimated normalized shortages
nshort <- get_normalized_shortages(model, est@coef)
**get_number_of_observations**

*Number of observations.*

**Description**

Returns the number of observations that are used by an initialized model. The number of used observations may differ from the numbers of observations of the data set that was passed to the model’s initialization.

**Usage**

```r
get_number_of_observations(object)
```

## S4 method for signature 'model_base'

```r
get_number_of_observations(object)
```

**Arguments**

- **object**: A model object.

**Value**

The number of used observations.

---

**get_prefixed_const_variable**

*Constant coefficient variable name.*

**Description**

The constant coefficient name is constructed by concatenating the equation prefix with `CONST`.

**Usage**

```r
get_prefixed_const_variable(object)
```

## S4 method for signature 'equation_base'

```r
get_prefixed_const_variable(object)
```

**Arguments**

- **object**: An equation object.

**Value**

The constant coefficient name.
**get_prefixed_control_variables**

*Control variable names.*

**Description**

The controls of the equation are the independent variables without the price variable. Their names are constructed by concatenating the equation prefix with the name of the price column.

**Usage**

`get_prefixed_control_variables(object)`

**Arguments**

- `object` An equation object.

**Value**

A vector with the control variable names.

---

**get_prefixed_independent_variables**

*Independent variable names.*

**Description**

The names of the independent variables are constructed by concatenating the equation prefix with the column names of the data tibble.

**Usage**

`get_prefixed_independent_variables(object)`

**Arguments**

- `object` An equation object.

**Value**

A vector with the independent variable names.
**get_prefixed_price_variable**

*Price coefficient variable name.*

**Description**

The price coefficient name is constructed by concatenating the equation prefix with the name of the price column.

**Usage**

```r
get_prefixed_price_variable(object)
```

```
## S4 method for signature 'equation_base'
get_prefixed_independent_variables(object)
```

```
## S4 method for signature 'equation_base'
get_prefixed_price_variable(object)
```

```
## S4 method for signature 'equation_base'
get_prefixed_control_variables(object)
```

**Arguments**

- `object` An equation object.

**Value**

The price coefficient variable name.

**get_prefixed_variance_variable**

*Variance variable name.*

**Description**

The variance variables is constructed by concatenating the equation prefix with "VARIANCE".

**Usage**

```r
get_prefixed_variance_variable(object)
```

```
## S4 method for signature 'equation_base'
get_prefixed_variance_variable(object)
```
get_relative_shortages

Arguments

object An equation object.

Value

The variable name for the variance of the shock of the equation.

Description

Returns the shortages normalized by the supplied quantity at a given point.

Usage

get_relative_shortages(object, parameters)

## S4 method for signature 'diseq_base'
get_relative_shortages(object, parameters)

Arguments

object A disequilibrium model object.

parameters A vector of parameters at which the shortages are evaluated.

Value

A vector with the estimated normalized shortages.

Examples

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)
```
get_shortage_probabilities

# estimate a model object
est <- estimate(model)

# get estimated relative shortages
rshort <- get_relative_shortages(model, est@coef)

description

get_shortage_probabilities

Shortage probabilities.

Description

Returns the shortage probabilities, i.e. the probabilities of an observation coming from an excess demand regime, at the given point.

Usage

get_shortage_probabilities(object, parameters)

## S4 method for signature 'diseq_base'
get_shortage_probabilities(object, parameters)

Arguments

- object: A disequilibrium model object.
- parameters: A vector of parameters at which the shortage probabilities are evaluated.

Value

A vector with the estimated shortage probabilities.

Examples

simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
c("id", "date"), "Q", "P", # keys, quantity, and price variables
"P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
simulated_data, # data
use_correlated_shocks = TRUE # allow shocks to be correlated
)
```r
# estimate a model object
est <- estimate(model)

# get the estimated shortage probabilities
probs <- get_shortage_probabilities(model, est@coef)
```

---

### get_supplied_quantities

**Supplied quantities.**

**Description**

Calculates the supplied quantity for each observation.

**Usage**

```r
get_supplied_quantities(object, parameters)
```

**Arguments**

- `object`: A model object.
- `parameters`: A vector of model's parameters.

**Value**

A vector with the supplied quantities evaluated at the given parameter vector.

**See Also**

- `get_aggregate_supply`

**Examples**

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
  ...)
```

get_supply_descriptives

Calculates and returns basic descriptive statistics for the model’s supply data. Factor variables are excluded from the calculations.

Usage
get_supply_descriptives(object)

Arguments
object A model object.

Value
A data tibble containing descriptive statistics.

Description
Checks if an observation is in a shortage stage.

Description
Returns TRUE for the indices at which the shortages of the market are non-negative, i.e. the market is in an excess demand state. Returns FALSE for the remaining indices. The evaluation of the shortages is performed using the passed parameter vector.
Usage

```r
has_shortage(object, parameters)
```

## S4 method for signature 'diseq_base'

```r
has_shortage(object, parameters)
```

Arguments

- `object`: A disequilibrium model object.
- `parameters`: A vector of parameters at which the shortage probabilities are evaluated.

Value

A vector of Boolean values indicating observations with shortages.

Examples

```r
simulated_data <- simulate_model_data(
  "diseq_basic", 500, 3, # model type, observed entities, observed time points
  -0.9, 8.9, c(0.3, -0.2), c(-0.03, -0.01), # demand coefficients
  0.9, 4.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
  c("id", "date"), "Q", "P", # keys, quantity, and price variables
  "P + Xd1 + Xd2 + X1 + X2", "P + Xs1 + X1 + X2", # equation specifications
  simulated_data, # data
  use_correlated_shocks = TRUE # allow shocks to be correlated
)

# estimate a model object
est <- estimate(model)

# get the indices of estimated shortages
has_short <- has_shortage(model, est@coef)
```

Model initialization

Model initialization
Usage

## S4 method for signature 'diseq_basic'
initialize(
  .Object,
  key_columns,
  quantity_column,
  price_column,
  demand_specification,
  supply_specification,
  data,
  use_correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'diseq_deterministic_adjustment'
initialize(
  .Object,
  key_columns,
  time_column,
  quantity_column,
  price_column,
  demand_specification,
  supply_specification,
  data,
  use_correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'diseq_directional'
initialize(
  .Object,
  key_columns,
  time_column,
  quantity_column,
  price_column,
  demand_specification,
  supply_specification,
  data,
  use_correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'diseq_stochastic_adjustment'
initialize(
  .Object,
  key_columns,
  time_column,
  quantity_column,
initialize_model_base

Initialize model base

## S4 method for signature 'init'
initialize(
  .Object,
  key_columns,
  quantity_column,
  price_column,
  demand_specification,
  supply_specification,
  data,
  use_correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'eq_2sls'
initialize(
  .Object,
  key_columns,
  quantity_column,
  price_column,
  demand_specification,
  supply_specification,
  data,
  verbose = 0
)

## S4 method for signature 'eq_fiml'
initialize(
  .Object,
  key_columns,
  quantity_column,
  price_column,
  demand_specification,
  supply_specification,
  data,
  use_correlated_shocks = TRUE,
  verbose = 0
)

### Arguments

- **.Object**
  - The object to be constructed.
- **key_columns**
  - Key columns of the data set.
- **quantity_column**
  - The quantity variable of the data set.
- **price_column**
  - The price variable of the data set.
- **demand_specification**
  - A formula representation of the right hand side of the demand equation.
- **supply_specification**
  - A formula representation of the right hand side of the supply equation.
- **data**
  - The data set.
- **use_correlated_shocks**
  - Should the model be estimated using correlated shocks?
verbose

Verbosity level.

time_column

The time column of the data set.

price_specification

A formula representation of the price equation.

Details

## Common initialization
### Variable construction

The constructor prepares the model’s variables using the passed specifications. The specification strings are expected to follow the syntax of `formula`. The construction of the model’s data uses the variables that are extracted by these specification. The demand variables are extracted by a formula that uses the `quantity_column` on the left hand side and the `demand_specification` on the right hand side of the formula. The supply variables are constructed by the `quantity_column` and the `supply_specification`. In the case of the `diseq_stochastic_adjustment` model, the price dynamics’ variables are extracted using the `quantity_column` and the `price_specification`

### Data preparation

1. If the passed data set contains rows with NA values, they are dropped. If the verbosity level allows warnings, a warning is emitted reporting how many rows were dropped.
2. After dropping the rows, factor levels may be invalidated. If needed the constructor readjusts the factor variables by removing the unobserved levels. Factor indicators and interaction terms are automatically created.
3. The primary column is constructed by pasting the values of the `key_columns`.
4. In the case of the `diseq_directional`, `diseq_deterministic_adjustment`, and the `diseq_stochastic_adjustment` models, a column with lagged prices is constructed. Since lagged prices are unavailable for the observation of the first time points, these observations are dropped. If the verbosity level allows the emission of information messages, the constructor prints the number of dropped observations.
5. In the case of the `diseq_directional`, and the `diseq_stochastic_adjustment` models, a column with price differences is created.

Value

The initialized model.

Functions

• `initialize,diseq_basic-method`: Basic disequilibrium model base constructor
• `initialize,diseq_deterministic_adjustment-method`: Disequilibrium model with deterministic price adjustment constructor
• `initialize,diseq_directional-method`: Directional disequilibrium model base constructor
• `initialize,diseq_stochastic_adjustment-method`: Disequilibrium model with stochastic price adjustment constructor
• `initialize,eq_2sls-method`: Two stage least squares equilibrium model constructor
• `initialize,eq_fiml-method`: Full information maximum likelihood model constructor
minus_log_likelihood  Minus log-likelihood.

Description

Returns the opposite of the log-likelihood. The likelihood functions are based on Maddala and Nelson (1974). The likelihood expressions that the function uses are derived in Karapanagiotis (2020). The function calculates the model’s log likelihood by evaluating the log likelihood of each observation in the sample and summing the evaluation results.

Usage

minus_log_likelihood(object, parameters)

## S4 method for signature 'diseq_basic'
minus_log_likelihood(object, parameters)

## S4 method for signature 'diseq_deterministic_adjustment'
minus_log_likelihood(object, parameters)

## S4 method for signature 'diseq_directional'
minus_log_likelihood(object, parameters)

## S4 method for signature 'diseq_stochastic_adjustment'
minus_log_likelihood(object, parameters)

## S4 method for signature 'eq_fiml'
minus_log_likelihood(object, parameters)

Arguments

object  A model object.
parameters  A vector of parameters at which the function is to be evaluated.

Value

The opposite of the sum of the likelihoods evaluated for each observation.
Slots

logger  Logger object.

key_columns  Vector of column names that uniquely identify data records. For panel data this vector should contain an entity and a time point identifier.

time_column  Column name for the time point data.

explanatory_columns  Vector of explanatory column names for all model’s equations.
data_columns  Vector of model’s data column names. This is the union of the quantity, price and explanatory columns.
columns  Vector of primary key and data column names for all model’s equations.

model_tibble  Model data tibble.

model_type_string  Model type string description.

system  Model’s system of equations.

---

simulate_model_data  Simulate model data.

Description

Returns a data tibble with simulated data from a generating process that matches the passed model string. By default, the simulated observations of the controls are drawn from a normal distribution.
Usage

simulate_model_data(
  model_string,
  nobs,
  tobs,
  alpha_d,
  beta_d0,
  beta_d,
  eta_d,
  alpha_s,
  beta_s0,
  beta_s,
  eta_s,
  gamma,
  beta_p0,
  beta_p,
  sigma_d = 1,
  sigma_s = 1,
  sigma_p = 1,
  rho_ds = 0,
  rho_dp = 0,
  rho_sp = 0,
  seed = NA_integer_,
  price_generator = function(nobs) stats::rnorm(n = nobs, mean = 2.5, sd = 0.5),
  control_generator = function(nobs) stats::rnorm(n = nobs, mean = 2.5, sd = 0.5)
)

## S4 method for signature 'ANY'
simulate_model_data(
  model_string,
  nobs,
  tobs,
  alpha_d,
  beta_d0,
  beta_d,
  eta_d,
  alpha_s,
  beta_s0,
  beta_s,
  eta_s,
  gamma,
  beta_p0,
  beta_p,
  sigma_d = 1,
  sigma_s = 1,
  sigma_p = 1,
  rho_ds = 0,
  rho_dp = 0,
simulate_model_data

rho_sp = 0,
seed = NA_integer_,
price_generator = function(nobs) stats::rnorm(n = nobs, mean = 2.5, sd = 0.5),
control_generator = function(nobs) stats::rnorm(n = nobs, mean = 2.5, sd = 0.5)
)

Arguments

model_string Model type. It should be among eq_2sls, eq_fiml, diseq_basic, diseq_directional, diseq_deterministic_adjustment, and diseq_stochastic_adjustment.
nobs Number of simulated entities.
tobs Number of simulated dates.
alpha_d Price coefficient of demand.
beta_d0 Constant coefficient of demand.
beta_d Coefficients of exclusive demand controls.
eta_d Demand coefficients of common controls.
alpha_s Price coefficient of supply.
beta_s0 Constant coefficient of supply.
beta_s Coefficients of exclusive supply controls.
eta_s Supply coefficients of common controls.
gamma Price equation’s stability factor.
beta_p0 Price equation’s constant coefficient.
beta_p Price equation’s control coefficients.
sigma_d Demand shock’s standard deviation.
sigma_s Supply shock’s standard deviation.
sigma_p Price equation shock’s standard deviation.
rho_ds Demand and supply shocks’ correlation coefficient.
rho_dp Demand and price shocks’ correlation coefficient.
rho_sp Supply and price shocks’ correlation coefficient.
seed Pseudo random number generator seed.
price_generator Pseudo random number generator callback for prices.
control_generator Pseudo random number generator callback for non-price controls.

Value

The simulated data.
Index

diseq, 3
diseq_base-class, 4
diseq_basic, 4
diseq_basic-class, 4
diseq_deterministic_adjustment, 4, 29
diseq_deterministic_adjustment-class, 5
diseq_directional, 4, 5, 29
diseq_directional-class, 5
diseq_stochastic_adjustment, 4, 29
diseq_stochastic_adjustment-class, 6
eq_2sls, 3, 10
eq_2sls-class, 8
eq_base-class, 9
eq_fiml, 4
eq_fiml-class, 9
equation_base-class, 7
equation_basic-class, 7
equation_deterministic_adjustment-class, 7
equation_directional-class, 8
equation_stochastic_adjustment-class, 8
estimate, 10
estimate, diseq_base-method (estimate), 10
estimate, eq_2sls-method (estimate), 10
estimate, eq_fiml-method (estimate), 10

formula, 29

get_aggregate_demand, 11
get_aggregate_demand, model_base-method
   (get_aggregate_demand), 11
get_aggregate_supply, 12
get_aggregate_supply, model_base-method
   (get_aggregate_supply), 12
get_demand_descriptives, 14
get_demand_descriptives, model_base-method
   (get_demand_descriptives), 14
get_demanded_quantities, 13
get_demanded_quantities, model_base-method
   (get_demanded_quantities), 13
get_marginal_effect_at_mean, 15
get_marginal_effect_at_mean, diseq_base-method
   (get_marginal_effect_at_mean), 15
get_mean_marginal_effect, 16
get_mean_marginal_effect, diseq_base-method
   (get_mean_marginal_effect), 16
get_model_description, 17
get_model_description, model_base-method
   (get_model_description), 17
get_normalized_shortages, 18
get_normalized_shortages, diseq_base-method
   (get_normalized_shortages), 18
get_number_of_observations, 19
get_number_of_observations, model_base-method
   (get_number_of_observations), 19
get_prefixed_const_variable, 19
get_prefixed_const_variable, equation_base-method
   (get_prefixed_const_variable), 19
get_prefixed_control_variables, 20
get_prefixed_control_variables, equation_base-method
   (get_prefixed_price_variable), 21
get_prefixed_independent_variables, 20
get_prefixed_independent_variables, equation_base-method
   (get_prefixed_price_variable), 21
get_prefixed_price_variable, 21
get_prefixed_price_variable, equation_base-method
   (get_prefixed_price_variable), 21
get_prefixed_variance_variable, 21
get_prefixed_variance_variable, equation_base-method (get_prefixed_variance_variable), 21
get_relative_shortages, 22
get_relative_shortages, diseq_base-method (get_relative_shortages), 22
get_shortage_probabilities, 23
get_shortage_probabilities, diseq_base-method (get_shortage_probabilities), 23
get_supplied_quantities, 24
get_supplied_quantities, model_base-method (get_supplied_quantities), 24
get_supply_descriptives, 25
get_supply_descriptives, model_base-method (get_supply_descriptives), 25
has_shortage, 25
has_shortage, diseq_base-method (has_shortage), 25
initialize, diseq_basic-method (initialize_model_base), 26
initialize, diseq_deterministic_adjustment-method (initialize_model_base), 26
initialize, diseq_directional-method (initialize_model_base), 26
initialize, diseq_stochastic_adjustment-method (initialize_model_base), 26
initialize, eq_2sls-method (initialize_model_base), 26
initialize, eq_fiml-method (initialize_model_base), 26
initialize, model_base, 26
lm, 8
minus_log_likelihood, 30
minus_log_likelihood, diseq_basic-method (minus_log_likelihood), 30
minus_log_likelihood, diseq_deterministic_adjustment-method (minus_log_likelihood), 30
minus_log_likelihood, diseq_directional-method (minus_log_likelihood), 30
minus_log_likelihood, diseq_stochastic_adjustment-method (minus_log_likelihood), 30
minus_log_likelihood, eq_fiml-method (minus_log_likelihood), 30
mle2, 10, 15, 16
simulate_model_data, 31
simulate_model_data, ANY-method (simulate_model_data), 31
model_logger, 31
systemfit, 8, 10
model_logger-class, 31
simulate_model_data, ANY-method (simulate_model_data), 31
systemfit, 8, 10
model_logger-class, 31
simulate_model_data, ANY-method (simulate_model_data), 31
systemfit, 8, 10