# Package ‘diseq’

October 13, 2022

**Title** Estimation Methods for Markets in Equilibrium and Disequilibrium

**Version** 0.4.6

**Date** 2022-06-01

**Description** Superseded by package markets. Provides estimation methods for markets in equilibrium and disequilibrium. Supports the estimation of an equilibrium and four disequilibrium models with both correlated and independent shocks. Also provides post-estimation analysis tools, such as aggregation, marginal effect, and shortage 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>. Standard errors can be estimated by adjusting for heteroscedasticity or clustering. The equilibrium estimation constitutes a case of a system of linear, simultaneous equations. Instead, the disequilibrium models replace the market-clearing condition with a non-linear, short-side rule and allow for different specifications of price dynamics.

**Language** en-US

**URL** `https://github.com/pi-kappa-devel/diseq/`, `https://diseq.pikappa.eu/`

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

**Depends** R (>= 3.5.0)

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

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.0

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

**VignetteBuilder** knitr

**Collate** 'data.R' 'diseq.R' 'equation_base.R' 'system_base.R' 'model_logger.R' 'market_model.R' 'disequilibrium_model.R' 'diseq_basic.R' 'diseq_deterministic_adjustment.R' 'diseq_directional.R' 'diseq_stochastic_adjustment.R' 'equation_basic.R' 'equation_deterministic_adjustment.R' 'equation_directional.R' 'equation_stochastic_adjustment.R' 'equilibrium_model.R' 'system_basic.R' 'gradient_basic.R' 'system_deterministic_adjustment.R' 'gradient_deterministic_adjustment.R' 'system_directional.R' 'gradient_directional.R' 'system_equilibrium.R' 'gradient_equilibrium.R' 'system_stochastic_adjustment.R' 'gradient_stochastic_adjustment.R' 'hessian_basic.R' 'hessian_directional.R' 'likelihood_basic.R' 'likelihood_deterministic_adjustment.R' 'likelihood_directional.R' 'likelihood_equilibrium.R' 'likelihood_stochastic_adjustment.R' 'market_fit.R' 'model_simulation.R' 'zzz.R'

**LinkingTo** Rcpp, RcppGSL

**SystemRequirements** C++11

**NeedsCompilation** yes

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

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

**Repository** CRAN

**Date/Publication** 2022-06-01 21:30:12 UTC

# R topics documented:

---

`coef,market_fit-method`

*Estimated coefficients of a fitted market model.*

---

## Description

Returns the coefficients of the fitted model.

## Usage

```
## S4 method for signature 'market_fit'
coef(object)
```

## Arguments

| | |
|---|---|
| `object` | A fitted model object. |

## Value

A vector of estimated model coefficients.

## Examples

```
# estimate a model using the houses dataset
fit <- diseq_deterministic_adjustment(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
  RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),  correlated_shocks = FALSE,
  estimation_options = list(control = list(maxit = 1e+6)))
```

```
# access the estimated coefficients
coef(fit)
```

diseq *Estimation of models for markets in equilibrium and disequilibrium*

## Description

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) doi:10.2307/1914215. The package is using the expressions of the gradients of the likelihoods derived in Karapanagiotis (2020) doi:10.2139/ssrn.3525622.

## Details

Overview

This page gives an overview of the market model classes and the available documentation options of the package.

**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("market_clearing_assessment", package = "diseq")

Additionally, one can use the documentation examples. Some of them illustrate the package functionality using the houses dataset.

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

**Equilibrium model classes:**

equilibrium_model Equilibrium model that can be estimated using full information maximum likelihood or two-stage least squares.

**Disequilibrium model classes:**

diseq_basic Disequilibrium model only with a 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.

## equation_classes *Equation classes*

### Description

Equation classes

### Details

Classes with data and functionality describing equations of model systems.

### Functions

- `equation_base-class`: Equation base class
- `equation_basic-class`: Basic disequilibrium model equation class
- `equation_deterministic_adjustment-class`: Deterministic adjustment disequilibrium model equation class
- `equation_directional-class`: Directional disequilibrium model equation class
- `equation_stochastic_adjustment-class`: Stochastic adjustment disequilibrium model equation class

### Slots

`formula` The equation formula using prefixed variables.

`name` The name of the equation.

`variable_prefix` A prefix string for the variables of the equation.

`dependent_vector` The vector of the response.

`independent_matrix` A model data matrix with columns corresponding to the set of independent variables.

`price_vector` The vector of prices.

`control_matrix` A model data matrix with columns corresponding to the set of independent variables without prices.

`alpha_beta` A vector of right hand side coefficients.

`alpha` The price coefficient.

`beta` A vector of right hand side coefficient without the price coefficient.

`var` The variance of the equation's shock.

`sigma` The standard deviation of the equation's shock.

`h`

$$h_x = \frac{x - \mathrm{E}x}{\sqrt{\mathrm{Var}x}}$$

z

$$z_{xy} = \frac{h_x - \rho_{xy} h_y}{\sqrt{1 - \rho_{xy}^2}}$$

psi

$$\psi_x = \phi(h_x)$$

Psi

$$\Psi_x = 1 - \Phi(z_{xy})$$

mu_Q

$$\mu_Q = \mathrm{E}Q$$

var_Q

$$V_Q = \mathrm{Var}Q$$

sigma_Q

$$\sigma_Q = \sqrt{\mathrm{Var}Q}$$

rho_QP

$$\rho_Q = \frac{\mathrm{Cov}(Q,P)}{\sqrt{\mathrm{Var}Q \mathrm{Var}P}}$$

rho_1QP

$$\rho_{1,QP} = \frac{1}{\sqrt{1 - \rho_{QP}}}$$

rho_2QP

$$\rho_{2,QP} = \rho_{QP}\rho_{1,QP}$$

sigma_QP

$$\sigma_{QP} = \mathrm{Cov}(Q,P)$$

h_Q As in slot h

z_PQ As in slot z

z_QP As in slot z

separation_subset A vector of indicators specifying the observations of the sample described by this equation according to the separation rule of the model.

estimate *Model estimation.*

## Description

All models are estimated using full information maximum likelihood. The equilibrium_model can also be estimated using two-stage least squares. The maximum likelihood estimation is based on 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 implementation of the two-stage least square estimation of the equilibrium_model is based on systemfit.

## Usage

```
estimate(object, ...)

## S4 method for signature 'market_model'
estimate(
  object,
  gradient = "calculated",
  hessian = "calculated",
  standard_errors = "homoscedastic",
  ...
)

## S4 method for signature 'equilibrium_model'
estimate(object, method = "BFGS", ...)
```

## Arguments

| | |
|---|---|
| object | A model object. |
| ... | Named parameter used in the model's estimation. These are passed further down to the estimation call. For the equilibrium_model model, the parameters are passed to systemfit, if the method is set to 2SLS, or to mle2 for any other method. For the rest of the models, the parameters are passed to mle2. |
| gradient | One of two potential options: "numerical" and "calculated". By default, all the models are estimated using the analytic expressions of their likelihoods' gradients. |
| hessian | One of three potential options: "skip", "numerical", and "calculated". The default is to use the "calculated" Hessian for the model that expressions are available and the "numerical" Hessian in other cases. Calculated Hessian expressions are available for the basic and directional models. |
| standard_errors | One of three potential options: "homoscedastic", "heteroscedastic", or a vector with variables names for which standard error clusters are to be created. The default value is "homoscedastic". If the option "heteroscedastic" |

is passed, the variance-covariance matrix is calculated using heteroscedasticity adjusted (Huber-White) standard errors. If the vector is supplied, the variance-covariance matrix is calculated by grouping the score matrix based on the passed variables.

`method` A string specifying the estimation method. When the passed value is among `Nelder-Mead`, `BFGS`, `CG`, `L-BFGS-B`, `SANN`, and `Brent`, the model is estimated using full information maximum likelihood based on `mle2` functionality. When `2SLS` is supplied, the model is estimated using two-stage least squares based on `systemfit`. In this case, the function returns a list containing the first and second stage estimates. The default value is `BFGS`.

## Value

The object that holds the estimation result.

## Functions

- `estimate,market_model-method`: Full information maximum likelihood estimation.
- `estimate,equilibrium_model-method`: Equilibrium model estimation.

## Examples

```
# initialize the model using the houses dataset
model <- new(
  "diseq_deterministic_adjustment", # model type
  subject = ID, time = TREND, quantity = HS, price = RM,
  demand = RM + TREND + W + CSHS + L1RM + L2RM + MONTH,
  supply = RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(), # data
  correlated_shocks = FALSE # let shocks be independent
)

# estimate the model object (BFGS is used by default)
fit <- estimate(model)

# estimate the model by specifying the optimization details passed to the optimizer.
fit <- estimate(model, control = list(maxit = 1e+6))

# summarize results
summary(fit)
```

---

`formula,market_model-method`
*Market model formula.*

---

## Description

Market model formula.

## Usage

```
## S4 method for signature 'market_model'
formula(x)
```

## Arguments

x A market model object.

## Details

Market model formulas adhere to the following specification:

`quantity | price | subject | time ~ demand | supply`

where

- quantity The model's traded (observed) quantity variable.
- price The model's price variable.
- quantity The model's subject (e.g. firm) identification variable.
- quantity The model's time identification variable.
- demand The right hand side of the model's demand equation.
- supply The right hand side of the model's supply equation.

The `diseq_stochastic_adjustment` additionally specify price dynamics by appending the right hand side of the equation at the end of the formula, i.e.

`quantity | price | subject | time ~ demand | supply | price_dynamics`

The left hand side part of the model formula specifies the elements that are needed for initializing the model. The market models of the package prepare the data based on these four variables using their respective identification assumptions. See market model classes for more details.

The function provides access to the formula used in model initialization.

## Value

The model's formula

## Examples

```
model <- simulate_model(
  "diseq_stochastic_adjustment", list(
    # observed entities, observed time points
    nobs = 500, tobs = 3,
    # demand coefficients
    alpha_d = -0.1, beta_d0 = 9.8, beta_d = c(0.3, -0.2), eta_d = c(0.6, -0.1),
    # supply coefficients
```

```
    alpha_s = 0.1, beta_s0 = 6.1, beta_s = c(0.9), eta_s = c(-0.5, 0.2),
    # price equation coefficients
    gamma = 1.2, beta_p0 = 3.1, beta_p = c(0.8)
  ),
  seed = 31
)

# access the model's formula
formula(model)
```

---

`gradient` *Gradient*

---

## Description

Returns the gradient of the opposite of the log-likelihood evaluated at the passed parameters.

## Usage

```
gradient(object, parameters)

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

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

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

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

## S4 method for signature 'equilibrium_model'
gradient(object, parameters)
```

## Arguments

| | |
|---|---|
| `object` | A model object. |
| `parameters` | A vector of parameters at which the gradient is to be evaluated. |

## Value

The opposite of the model log likelihood’s gradient.

## hessian — *Hessian*

### Description

Returns the hessian of the opposite of the log-likelihood evaluated at the passed parameters.

### Usage

```
hessian(object, parameters)

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

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

### Arguments

| | |
|---|---|
| `object` | A model object. |
| `parameters` | A vector of parameters at which the hessian is to be evaluated. |

### Value

The opposite of the model log likelihood's hessian.

## houses — *Credit market data for US housing starts*

### Description

Credit market data for US housing starts

### Usage

```
data(houses)

fair_houses()
```

### Format

A data frame with 138 rows and 7 columns

## Details

**The basic houses dataset (`houses`):**

A dataset containing the monthly mortgage rates and other attributes of the US market for new, non-farm houses from July 1958 to December 1969. The variables are as follows:

- `DATE` The date of the record.
- `HS` Private non-farm housing starts in thousands of units (Not seasonally adjusted).
- `RM` FHA Mortgage rate series on new homes in units of 100 ( beginning-of-month Data).
- `DSLA` Savings capital (deposits) of savings and loan associations in millions of dollars.
- `DMSB` Deposits of mutual savings banks in millions of dollars.
- `DHLB` Advances of the federal home loan bank to savings and loan associations in million of dollars.
- `W` Number of working days in month.

**Generate the variables of the Fair & Jaffee (1972) dataset. (`fair_houses`):**

Loads the `houses` dataset and creates the additional variables used by Fair & Jaffee (1972) doi:10.2307/1913181. These are

- `ID` A dummy entity identifier that is always equal to one since the houses data have only a time series component.
- `DSF` Flow of deposits in savings and loan associations and mutual savings banks in million of dollars. Equal to

$$DSLA_t + DMSB_t - (DSLA_{t-1} + DMSB_{t-1}).$$

- `DHF` Flow of advances of the federal home loan bank to savings and loan associations in million of dollars. Equal to

$$DHLB_t - DHLB_{t-1}.$$

- `MONTH` The month of the date of the observation.
- `L1RM` FHA Mortgage rate series on new homes in units of 100, lagged by one date.
- `L2RM` FHA Mortgage rate series on new homes in units of 100, lagged by two dates.
- `L1HS` Private non-farm housing starts in thousands of units (Not seasonally adjusted), lagged by one date.
- `CSHS` The cumulative sum of past housing starts. Used to proxy the stock of houses
- `MA6DSF` Moving average of order 6 of the flow of deposits in savings associations and loan associations and mutual savings banks.
- `MA3DHF` Moving average of order 3 of the flow of advances of the federal home loan bank to savings and loan associations.
- `TREND` A time trend variable.

Returns A modified version of the `houses` dataset.

## Functions

- `fair_houses`: Generate Fair & Jaffee (1972) dataset

### Source

- HS Economic Reports of the President
- RM Fair (1971)
- DSLA Federal Reserve Bulletins
- DMSB Federal Reserve Bulletins
- DHLB Federal Reserve Bulletins
- W Manually calculated

### References


- Fair, R. C. (1971). A short-run forecasting model of the United States economy. Heath Lexington Books.
- Fair, R. C., & Jaffee, D. M. (1972). Methods of Estimation for Markets in Disequilibrium. Econometrica, 40(3), 497. doi:10.2307/1913181
- Maddala, G. S., & Nelson, F. D. (1974). Maximum Likelihood Methods for Models of Markets in Disequilibrium. Econometrica, 42(6), 1013. doi:10.2307/1914215
- Hwang, H. (1980). A test of a disequilibrium model. Journal of Econometrics, 12(3), 319–333. doi:10.1016/03044076(80)900597


### Examples

```
data(houses)
head(houses)
head(fair_houses())
```

---

## initialize_market_model

*Model initialization*

---

### Description

Model initialization

### Usage

```
## S4 method for signature 'diseq_basic'
initialize(
  .Object,
  quantity,
  price,
  demand,
  supply,
  subject,
  time,
  data,
```

```
  correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'diseq_deterministic_adjustment'
initialize(
  .Object,
  quantity,
  price,
  demand,
  supply,
  subject,
  time,
  data,
  correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'diseq_directional'
initialize(
  .Object,
  quantity,
  price,
  demand,
  supply,
  subject,
  time,
  data,
  correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'diseq_stochastic_adjustment'
initialize(
  .Object,
  quantity,
  price,
  demand,
  supply,
  price_dynamics,
  subject,
  time,
  data,
  correlated_shocks = TRUE,
  verbose = 0
)

## S4 method for signature 'equilibrium_model'
```

```
initialize(
  .Object,
  quantity,
  price,
  demand,
  supply,
  subject,
  time,
  data,
  correlated_shocks = TRUE,
  verbose = 0
)
```

## Arguments

| | |
|---|---|
| `.Object` | The object to be Constructed. |
| `quantity` | The quantity variable of the system. |
| `price` | The price variable of the system. |
| `demand` | A formula representation of the right hand side of the demand equation. |
| `supply` | A formula representation of the right hand side of the supply equation. |
| `subject` | The subject identifier of the data set. |
| `time` | The time identifier of the data set. |
| `data` | The data set. |
| `correlated_shocks` | Should the model be estimated using correlated shocks? |
| `verbose` | Verbosity level. |
| `price_dynamics` | A formula representation of the price equation. |

## Details

The following two subsections describe the common initialization steps of all market model classes.

**Variable construction:** The constructor prepares the model's variables using the passed specifications. The specification variables are expected to be of type `language`. The right hand side specifications of the system are expected to follow the syntax of `formula`. The construction of the model's data uses the variables extracted by these specification. The demand variables are extracted by a formula that uses the `quantity` on the left hand side and the `demand` on the right hand side of the formula. The supply variables are constructed by the `quantity` and the `supply` inputs. In the case of the `diseq_stochastic_adjustment` model, the price dynamics' variables are extracted using the `price dynamics` input. The `price dynamics` for the `diseq_stochastic_adjustment` should contain only terms other than that of excess demand. The excess demand term of the price equation is automatically generated by the constructor.

**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 key column is constructed by pasting the values of the columns of the subject and time variables.

4. In the cases 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 observations of the first time point, 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 cases 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,equilibrium_model-method`: Equilibrium model constructor

## Examples

```
simulated_data <- simulate_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, 6.2, c(0.03), c(-0.05, 0.02) # supply coefficients
)

# initialize the model
model <- new(
  "diseq_basic", # model type
  subject = id, time = date, quantity = Q, price = P,
  demand = P + Xd1 + Xd2 + X1 + X2, supply = P + Xs1 + X1 + X2,
  simulated_data, # data
  correlated_shocks = FALSE # use independent shocks
)

show(model)
simulated_data <- simulate_data(
  # model type, observed entities and time points
  "diseq_deterministic_adjustment", 500, 3,
  # demand coefficients
```

```
  -0.9, 8.9, c(0.03, -0.02), c(-0.03, -0.01),
  # supply coefficients
  0.9, 4.2, c(0.03), c(0.05, 0.02),
  # price adjustment coefficient
  1.4
)

# initialize the model
model <- new(
  "diseq_deterministic_adjustment", # model type
  subject = id, time = date, quantity = Q, price = P,
  demand = P + Xd1 + Xd2 + X1 + X2, supply = P + Xs1 + X1 + X2,
  simulated_data, # data
  correlated_shocks = TRUE # allow shocks to be correlated
)

show(model)

simulated_data <- simulate_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
  subject = id, time = date, quantity = Q, price = P,
  demand = P + Xd1 + Xd2 + X1 + X2, supply = Xs1 + X1 + X2,
  simulated_data, # data
  correlated_shocks = TRUE # allow shocks to be correlated
)

show(model)

simulated_data <- simulate_data(
  # model type, observed entities and time points
  "diseq_stochastic_adjustment", 500, 3,
  # demand coefficients
  -0.1, 9.8, c(0.3, -0.2), c(0.6, 0.1),
  # supply coefficients
  0.1, 7.1, c(0.9), c(-0.5, 0.2),
  # price adjustment coefficient
  1.4, 3.1, c(0.8)
)

# initialize the model
model <- new(
  "diseq_stochastic_adjustment", # model type
  subject = id, time = date, quantity = Q, price = P,
  demand = P + Xd1 + Xd2 + X1 + X2, supply = P + Xs1 + X1 + X2,
  price_dynamics = Xp1,
  simulated_data, # data
```

```
  correlated_shocks = TRUE # allow shocks to be correlated
)

show(model)
simulated_data <- simulate_data(
  "equilibrium_model", 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(
  "equilibrium_model", # model type
  subject = id, time = date, quantity = Q, price = P,
  demand = P + Xd1 + Xd2 + X1 + X2, supply = P + Xs1 + X1 + X2,
  simulated_data, # data
  correlated_shocks = TRUE # allow shocks to be correlated
)

show(model)
```

---

## logLik,market_fit-method

*Log likelihood of a fitted market model.*

---

### Description

Specializes the logLik function for the market models of the package estimated with full information minimum likelihood. It returns NULL for the equilibrium model estimated with systemfit.

### Usage

```
## S4 method for signature 'market_fit'
logLik(object)
```

### Arguments

| | |
|---|---|
| object | A fitted model object. |

### Value

A logLik object.

### Examples

```
# estimate a model using the houses dataset
fit <- diseq_deterministic_adjustment(
  HS | RM | ID | TREND ~
```

```
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
  RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),  correlated_shocks = FALSE,
  estimation_options = list(control = list(maxit = 1e+6)))

# get the log likelihood object
logLik(fit)
```

---

`marginal_effects` *Marginal effects*

---

## Description

Returns the estimated effect of a variable.

## Usage

```
shortage_marginal(fit, variable, model, parameters)

shortage_probability_marginal(
  fit,
  variable,
  aggregate = "mean",
  model,
  parameters
)

## S4 method for signature 'missing,ANY,market_model,ANY'
shortage_marginal(variable, model, parameters)

## S4 method for signature 'missing,ANY,ANY,market_model,ANY'
shortage_probability_marginal(variable, aggregate, model, parameters)

## S4 method for signature 'missing,ANY,market_model,ANY'
shortage_marginal(variable, model, parameters)

## S4 method for signature 'market_fit,ANY,missing,missing'
shortage_marginal(fit, variable)

## S4 method for signature 'market_fit,ANY,ANY,missing,missing'
shortage_probability_marginal(fit, variable, aggregate)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted market model. |
| `variable` | Variable name for which the effect is calculated. |

| | |
|---|---|
| `model` | A market model object. |
| `parameters` | A vector of parameters. |
| `aggregate` | Mode of aggregation. Valid options are "mean" (the default) and "at_the_mean". |

## Value

The estimated effect of the passed variable.

## Functions

- `shortage_marginal`: Marginal effect on market system
  Returns the estimated marginal effect of a variable on the market system. For a system variable $x$ with demand coefficient $\beta_{d,x}$ and supply coefficient $\beta_{s,x}$, the marginal effect on the market system is given by
  $$M_x = \frac{\beta_{d,x} - \beta_{s,x}}{\sqrt{\sigma_d^2 + \sigma_s^2 - 2\rho_{ds}\sigma_d\sigma_s}}.$$
- `shortage_probability_marginal`: Marginal effect on shortage probabilities
  Returns the estimated marginal effect of a variable on the probability of observing a shortage state. The mean marginal effect on the shortage probability is given by
  $$M_x \mathrm{E}\phi\left(\frac{D - S}{\sqrt{\sigma_d^2 + \sigma_s^2 - 2rho\sigma_d\sigma_s}}\right)$$
  and the marginal effect at the mean by
  $$M_x \phi\left(\mathrm{E}\frac{D - S}{\sqrt{\sigma_d^2 + \sigma_s^2 - 2rho\sigma_d\sigma_s}}\right)$$
  where $M_x$ is the marginal effect on the system, $D$ is the demanded quantity, $S$ the supplied quantity, and $\phi$ is the standard normal density.

## Examples

```
# estimate a model using the houses dataset
fit <- diseq_deterministic_adjustment(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
  RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),  correlated_shocks = FALSE,
  estimation_options = list(control = list(maxit = 1e+5)))

# mean marginal effect of variable "RM" on the shortage probabilities
#' shortage_probability_marginal(fit, "RM")

# marginal effect at the mean of variable "RM" on the shortage probabilities
shortage_probability_marginal(fit, "CSHS", aggregate = "at_the_mean")

# marginal effect of variable "RM" on the system
shortage_marginal(fit, "RM")
```

`market_aggregation` *Market side aggregation.*

## Description

Market side aggregation.

## Usage

```
aggregate_demand(fit, model, parameters)

## S4 method for signature 'missing,market_model,ANY'
aggregate_demand(model, parameters)

aggregate_supply(fit, model, parameters)

## S4 method for signature 'missing,market_model,ANY'
aggregate_supply(model, parameters)

## S4 method for signature 'market_fit,missing,missing'
aggregate_demand(fit)

## S4 method for signature 'market_fit,missing,missing'
aggregate_supply(fit)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted market model object. |
| `model` | A model object. |
| `parameters` | A vector of model's parameters. |

## Details

Calculates the sample's aggregate demand or supply using the estimated coefficients of a fitted model. Alternatively, the function calculates aggregates using a model and a set of parameters passed separately. If the model's data have multiple distinct subjects at each date, aggregation is calculated over subjects per unique date. If the model has time series data, namely a single subject per time point, aggregation is ululated over all time pints.

## Value

The sum of the estimated demanded or supplied quantities evaluated at the given parameters.

## Functions

- `aggregate_demand`: Demand aggregation.
- `aggregate_supply`: Supply aggregation.

### See Also

demanded_quantities, supplied_quantities

### Examples

```
fit <- diseq_basic(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
    RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),
  correlated_shocks = FALSE
)

# get estimated aggregate demand
aggregate_demand(fit)

# simulate the deterministic adjustment model
model <- simulate_model(
  "diseq_deterministic_adjustment", list(
    # observed entities, observed time points
    nobs = 500, tobs = 3,
    # demand coefficients
    alpha_d = -0.6, beta_d0 = 9.8, beta_d = c(0.3, -0.2), eta_d = c(0.6, -0.1),
    # supply coefficients
    alpha_s = 0.2, beta_s0 = 4.1, beta_s = c(0.9), eta_s = c(-0.5, 0.2),
    # price equation coefficients
    gamma = 0.9
  ),
  seed = 1356
)

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

# get estimated aggregate demand
aggregate_demand(fit)

# get estimated aggregate demand
aggregate_supply(fit)
```

---

## market_descriptives *Market side descriptive statistics*

---

### Description

Market side descriptive statistics

## Usage

```
demand_descriptives(object)

supply_descriptives(object)

## S4 method for signature 'market_model'
demand_descriptives(object)

## S4 method for signature 'market_model'
supply_descriptives(object)
```

## Arguments

| | |
|---|---|
| `object` | A model object. |

## Details

Calculates and returns basic descriptive statistics for the model's demand or supply side data. Factor variables are excluded from the calculations. The function calculates and returns:

- `nobs` Number of observations.
- `nmval` Number of missing values.
- `min` Minimum observation.
- `max` Maximum observation.
- `range` Observations' range.
- `sum` Sum of observations.
- `median` Median observation.
- `mean` Mean observation.
- `mean_se` Mean squared error.
- `mean_ce` Confidence interval bound.
- `var` Variance.
- `sd` Standard deviation.
- `coef_var` Coefficient of variation.

## Value

A data `tibble` containing descriptive statistics.

## Functions

- `demand_descriptives`: Demand descriptive statistics.
- `supply_descriptives`: Supply descriptive statistics.

## Examples

```
# initialize the basic model using the houses dataset
model <- new(
  "diseq_basic", # model type
  subject = ID, time = TREND, quantity = HS, price = RM,
  demand = RM + TREND + W + CSHS + L1RM + L2RM + MONTH,
  supply = RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(), # data
  correlated_shocks = FALSE # allow shocks to be correlated
)

# get descriptive statistics of demand side variables
demand_descriptives(model)

# get descriptive statistics of supply side variables
supply_descriptives(model)
```

---

`market_models` *Market model classes*

---

## Description

**diseq_basic:** 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.

$$D_{nt} = X'_{d,nt}\beta_d + u_{d,nt},$$

$$S_{nt} = X'_{s,nt}\beta_s + u_{s,nt},$$

$$Q_{nt} = \min\{D_{nt}, S_{nt}\}.$$

**diseq_deterministic_adjustment:** 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.

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

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

$$Q_{nt} = \min\{D_{nt}, S_{nt}\},$$

$$\Delta P_{nt} = \frac{1}{\gamma}\left(D_{nt} - S_{nt}\right).$$

**diseq_directional:** 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 states 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 state. Otherwise, it is assumed that it represents an excess supply state. The model is estimated using full information maximum likelihood.

$$D_{nt} = X_{d,nt}'\beta_d + u_{d,nt},$$

$$S_{nt} = X_{s,nt}'\beta_s + u_{s,nt},$$

$$Q_{nt} = \min\{D_{nt}, S_{nt}\},$$

$$\Delta P_{nt} \geq 0 \Longrightarrow D_{nt} \geq S_{nt}.$$

**diseq_stochastic_adjustment:** The disequilibrium model with stochastic price adjustment is described by a system of four equations. Three of of them form a stochastic linear system of market equations 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.

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

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

$$Q_{nt} = \min\{D_{nt}, S_{nt}\},$$

$$\Delta P_{nt} = \frac{1}{\gamma}\left(D_{nt} - S_{nt}\right) + X_{p,nt}'\beta_p + u_{p,nt}.$$

**equilibrium_model:** The equilibrium model consists of thee equations. The demand, the supply and the market clearing equations. The model can be estimated using both full information maximum likelihood and two-stage least squares.

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

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

$$Q_{nt} = D_{nt} = S_{nt}.$$

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. In the first stage of the two-stage least square estimation, 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.

### Functions

- market_model-class: Base class for market models
- disequilibrium_model-class: Base class for disequilibrium models
- diseq_basic-class: Basic disequilibrium model with unknown sample separation.
- diseq_deterministic_adjustment-class: Disequilibrium model with deterministic price dynamics.
- diseq_directional-class: Directional disequilibrium model with sample separation.
- diseq_stochastic_adjustment-class: Disequilibrium model with stochastic price dynamics.
- equilibrium_model-class: Equilibrium model

### Slots

logger Logger object.

subject_columns Column name for the subject identifier.

time_column Column name for the time point identifier.

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.

### See Also

initialize_market_model

---

## market_quantities *Estimated market quantities.*

---

### Description

Estimated market quantities.

## Usage

```
demanded_quantities(fit, model, parameters)

## S4 method for signature 'missing,market_model,ANY'
demanded_quantities(model, parameters)

supplied_quantities(fit, model, parameters)

## S4 method for signature 'missing,market_model,ANY'
supplied_quantities(model, parameters)

## S4 method for signature 'market_fit,missing,missing'
demanded_quantities(fit)

## S4 method for signature 'market_fit,missing,missing'
supplied_quantities(fit)
```

## Arguments

| | |
|---|---|
| fit | A fitted model object. |
| model | A model object. |
| parameters | A vector of model's parameters. |

## Details

Calculates and returns the estimated demanded or supplied quantities for each observation at the passed vector of parameters.

## Value

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

## Functions

- `demanded_quantities`: Estimated demanded quantities.
- `supplied_quantities`: Estimated supplied quantities.

## Examples

```
fit <- diseq_basic(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
    RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),
  correlated_shocks = FALSE
)

# get estimated demanded and supplied quantities
```

```
head(cbind(
  demanded_quantities(fit),
  supplied_quantities(fit)
))
```

market_simulation *Market model simulation*

### Description

Market data and model simulation functionality based on the data generating process induced by the market model specifications.

`simulate_data`: 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.

`simulate_model`: Simulates a data `tibble` based on the generating process of the passed model and uses it to initialize a model object. Data are simulated using the `simulate_data` function.

### Usage

```
simulate_data(
  model_type_string,
  nobs = NA_integer_,
  tobs = NA_integer_,
  alpha_d = NA_real_,
  beta_d0 = NA_real_,
  beta_d = NA_real_,
  eta_d = NA_real_,
  alpha_s = NA_real_,
  beta_s0 = NA_real_,
  beta_s = NA_real_,
  eta_s = NA_real_,
  gamma = NA_real_,
  beta_p0 = NA_real_,
  beta_p = NA_real_,
  sigma_d = 1,
  sigma_s = 1,
  sigma_p = 1,
  rho_ds = 0,
  rho_dp = 0,
  rho_sp = 0,
  seed = NA_integer_,
  price_generator = function(n) stats::rnorm(n = n),
  control_generator = function(n) stats::rnorm(n = n),
  verbose = 0
```

```
)

## S4 method for signature 'ANY'
simulate_data(
  model_type_string,
  nobs = NA_integer_,
  tobs = NA_integer_,
  alpha_d = NA_real_,
  beta_d0 = NA_real_,
  beta_d = NA_real_,
  eta_d = NA_real_,
  alpha_s = NA_real_,
  beta_s0 = NA_real_,
  beta_s = NA_real_,
  eta_s = NA_real_,
  gamma = NA_real_,
  beta_p0 = NA_real_,
  beta_p = NA_real_,
  sigma_d = 1,
  sigma_s = 1,
  sigma_p = 1,
  rho_ds = 0,
  rho_dp = 0,
  rho_sp = 0,
  seed = NA_integer_,
  price_generator = function(n) stats::rnorm(n = n),
  control_generator = function(n) stats::rnorm(n = n),
  verbose = 0
)

simulate_model(
  model_type_string,
  simulation_parameters,
  seed = NA,
  verbose = 0,
  ...
)

## S4 method for signature 'ANY'
simulate_model(
  model_type_string,
  simulation_parameters,
  seed = NA,
  verbose = 0,
  ...
)
```

## Arguments

model_type_string
: Model type. It should be among equilibrium_model, 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. The default generator is $N(2.5, 0.25)$.

control_generator
: Pseudo random number generator callback for non-price controls. The default generator is $N(2.5, 0.25)$.

verbose: Verbosity level.

simulation_parameters
: List of parameters used in model simulation. See the simulate_data function for details.

...: Additional parameters to be passed to the model's constructor.

## Value

**simulate_data:** The simulated data.

**simulate_model:** The simulated model.

### Functions

- `simulate_data`: Simulate model data.
- `simulate_model`: Simulate model.

---

`maximize_log_likelihood`

*Maximize the log-likelihood.*

---

### Description

Maximizes the log-likelihood using the `GSL` implementation of the BFGS algorithm. This function is primarily intended for advanced usage. The `estimate` functionality is a fast, analysis-oriented alternative. If the `GSL` is not available, the function returns a trivial result list with status set equal to -1. If the C++17 execution policies are available, the implementation of the optimization is parallelized.

### Usage

```
maximize_log_likelihood(
  object,
  start,
  step,
  objective_tolerance,
  gradient_tolerance,
  max_it
)

## S4 method for signature 'equilibrium_model'
maximize_log_likelihood(
  object,
  start,
  step,
  objective_tolerance,
  gradient_tolerance,
  max_it
)
```

### Arguments

| | |
|---|---|
| `object` | A model object. |
| `start` | Initializing vector. |
| `step` | Optimization step. |
| `objective_tolerance` | Objective optimization tolerance. |
| `gradient_tolerance` | Gradient optimization tolerance. |
| `max_it` | Maximum allowed number of iterations. |

### Value

A list with the optimization output.

### See Also

estimate

### Examples

```
model <- simulate_model(
  "equilibrium_model", list(
    # observed entities, observed time points
    nobs = 500, tobs = 3,
    # demand coefficients
    alpha_d = -0.9, beta_d0 = 14.9, beta_d = c(0.3, -0.2), eta_d = c(-0.03, -0.01),
    # supply coefficients
    alpha_s = 0.9, beta_s0 = 3.2, beta_s = c(0.03), eta_s = c(0.05, 0.02)
  ),
  seed = 99
)

# maximize the model's log-likelihood
mll <- maximize_log_likelihood(
  model,
  start = NULL, step = 1e-5,
  objective_tolerance = 1e-4, gradient_tolerance = 1e-3, max_it = 1e+3
)
mll
```

---

## minus_log_likelihood *Minus log-likelihood.*

---

### Description

Returns the opposite of the log-likelihood. The likelihood functions are based on Maddala and Nelson (1974) doi:10.2307/1914215. The likelihoods expressions that the function uses are derived in Karapanagiotis (2020) doi:10.2139/ssrn.3525622. 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 'equilibrium_model'
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.

---

`model_logger-class` *Logger class*

---

## Description

Logger class

## Slots

`verbosity` Controls the intensity of output messages. Errors are always printed. Other than this, a value of

- **1** prints warnings,
- **2** prints basic information,
- **3** prints verbose information and,
- **4** prints debug information.

## model_name *Model description.*

### Description

A unique identifying string for the model.

### Usage

```
model_name(object)

## S4 method for signature 'market_model'
model_name(object)
```

### Arguments

| | |
|---|---|
| object | A model object. |

### Value

A string representation of the model.

## nobs,market_model-method *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

```
## S4 method for signature 'market_model'
nobs(object)
```

### Arguments

| | |
|---|---|
| object | A model object. |

### Value

The number of used observations.

---

`plot,market_fit,ANY-method`
*Plots the fitted model.*

---

## Description

Displays a graphical illustration of the passed fitted model object. The function creates a scatter plot of quantity-price pairs for the records corresponding to the given subject and time identifiers. Then, it plots the average fitted demand and supply quantities for the same data subset letting prices vary between the minimum and maximum price points observed in the data subset.

## Usage

```
## S4 method for signature 'market_fit,ANY'
plot(x, subject, time, ...)
```

## Arguments

| | |
|---|---|
| `x` | A model object. |
| `subject` | A vector of subject identifiers to be used in the visualization. |
| `time` | A vector of time identifiers to be used in the visualization. |
| `...` | Additional parameter to be used for styling the figure. Specifically `xlab`, `ylab`, and `main` are currently handled by the function. |

## Examples

```
# estimate a model using the houses dataset
fit <- diseq_deterministic_adjustment(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
  RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),  correlated_shocks = FALSE,
  estimation_options = list(control = list(maxit = 1e+6)))

# show model's illustration plot
plot(fit)
```

scores *Likelihood scores.*

## Description

It calculates the gradient of the likelihood at the given parameter point for each observation in the sample. It, therefore, returns an n x k matrix, where n denotes the number of observations in the sample and k the number of estimated parameters. The ordering of the parameters is the same as the one that is used in the summary of the results. The method can be called either using directly a fitted model object, or by separately providing a model object and a parameter vector.

## Usage

```
scores(object, parameters, fit = missing())

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

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

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

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

## S4 method for signature 'equilibrium_model,ANY,ANY'
scores(object, parameters)

## S4 method for signature 'missing,missing,market_fit'
scores(fit)
```

## Arguments

| | |
|---|---|
| object | A model object. |
| parameters | A vector with model parameters. |
| fit | A fitted model object. |

## Value

The score matrix.

## Examples

```
model <- simulate_model(
```

```
  "diseq_basic", list(
    # observed entities, observed time points
    nobs = 500, tobs = 3,
    # demand coefficients
    alpha_d = -0.9, beta_d0 = 8.9, beta_d = c(0.6), eta_d = c(-0.2),
    # supply coefficients
    alpha_s = 0.9, beta_s0 = 7.9, beta_s = c(0.03, 1.2), eta_s = c(0.1)
  ),
  seed = 7523
)

# estimate the model object (BFGS is used by default)
fit <- estimate(model)

# Calculate the score matrix
head(scores(model, coef(fit)))
```

---

`shortage_analysis` *Analysis of shortages*

---

## Description

Analysis of shortages

## Usage

```
shortages(fit, model, parameters)

normalized_shortages(fit, model, parameters)

relative_shortages(fit, model, parameters)

shortage_probabilities(fit, model, parameters)

shortage_indicators(fit, model, parameters)

shortage_standard_deviation(fit, model, parameters)

## S4 method for signature 'missing,market_model,ANY'
shortages(model, parameters)

## S4 method for signature 'missing,market_model,ANY'
normalized_shortages(model, parameters)

## S4 method for signature 'missing,market_model,ANY'
relative_shortages(model, parameters)
```

```
## S4 method for signature 'missing,market_model,ANY'
shortage_probabilities(model, parameters)

## S4 method for signature 'missing,market_model,ANY'
shortage_indicators(model, parameters)

## S4 method for signature 'missing,market_model,ANY'
shortage_standard_deviation(model, parameters)

## S4 method for signature 'missing,diseq_stochastic_adjustment,ANY'
shortage_standard_deviation(model, parameters)

## S4 method for signature 'market_fit,missing,missing'
shortages(fit)

## S4 method for signature 'market_fit,missing,missing'
normalized_shortages(fit)

## S4 method for signature 'market_fit,missing,missing'
relative_shortages(fit)

## S4 method for signature 'market_fit,missing,missing'
shortage_probabilities(fit)

## S4 method for signature 'market_fit,missing,missing'
shortage_indicators(fit)

## S4 method for signature 'market_fit,missing,missing'
shortage_standard_deviation(fit)
```

### Arguments

| | |
|---|---|
| `fit` | A fitted model object. |
| `model` | A market model object. |
| `parameters` | A vector of parameters at which the shortages are evaluated. |

### Details

The following methods offer functionality for analyzing estimated shortages of the market models. The methods can be called either using directly a fitted model object, or by separately providing a model object and a parameter vector.

**shortages:** Returns the predicted shortages at a given point.

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

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

**shortage_probabilities:** Returns the shortage probabilities, i.e. the probabilities of an observation coming from an excess demand state, at the given point.

**shortage_indicators:** Returns a vector of indicators (Boolean values) for each observation. An element of the vector is TRUE for observations at which the estimated shortages are non-negative, i.e. the market at in an excess demand state. The remaining elements are FALSE. The evaluation of the shortages is performed using the passed parameter vector.

**shortage_standard_deviation:** Returns the variance of excess demand.

## Value

A vector with the (estimated) shortages.

## Functions

- `shortages`: Shortages.
- `normalized_shortages`: Normalized shortages.
- `relative_shortages`: Relative shortages.
- `shortage_probabilities`: Shortage probabilities.
- `shortage_indicators`: Shortage indicators.
- `shortage_standard_deviation`: Shortage variance.

## Examples

```
# estimate a model using the houses dataset
fit <- diseq_deterministic_adjustment(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
  RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),  correlated_shocks = FALSE,
  estimation_options = list(control = list(maxit = 1e+5)))

# get estimated normalized shortages
head(normalized_shortages(fit))

# get estimated relative shortages
head(relative_shortages(fit))

# get the estimated shortage probabilities
head(shortage_probabilities(fit))

# get the estimated shortage indicators
head(shortage_indicators(fit))

# get the estimated shortages
head(shortages(fit))

# get the estimated shortage variance
```

```
shortage_standard_deviation(fit)
```

---

show,market_model-method
*Prints a short description of the model.*

---

## Description

Sends basic information about the model to standard output.

## Usage

```
## S4 method for signature 'market_model'
show(object)
```

## Arguments

| | |
|---|---|
| object | A model object. |

## Examples

```
model <- simulate_model(
  "diseq_stochastic_adjustment", list(
    # observed entities, observed time points
    nobs = 500, tobs = 3,
    # demand coefficients
    alpha_d = -0.1, beta_d0 = 9.8, beta_d = c(0.3, -0.2), eta_d = c(0.6, -0.1),
    # supply coefficients
    alpha_s = 0.1, beta_s0 = 7.1, beta_s = c(0.9), eta_s = c(-0.5, 0.2),
    # price equation coefficients
    gamma = 1.2, beta_p0 = 3.1, beta_p = c(0.8)
  ),
  seed = 31
)

# print short model information
show(model)
```

---

`single_call_estimation` *Single call estimation*

---

### Description

Single call estimation

### Usage

```
diseq_basic(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

## S4 method for signature 'formula'
diseq_basic(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

diseq_deterministic_adjustment(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

## S4 method for signature 'formula'
diseq_deterministic_adjustment(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

diseq_directional(
  specification,
  data,
```

```
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

## S4 method for signature 'formula'
diseq_directional(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

diseq_stochastic_adjustment(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

## S4 method for signature 'formula'
diseq_stochastic_adjustment(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

equilibrium_model(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)

## S4 method for signature 'formula'
equilibrium_model(
  specification,
  data,
  correlated_shocks = TRUE,
  verbose = 0,
  estimation_options = list()
)
```

## Arguments

`specification` The model's formula.

`data` The data to be used with the model.

`correlated_shocks` Should the model's system entail correlated shocks? By default the argument is set to `TRUE`.

`verbose` The verbosity with which operations on the model print messages. By default the value is set to `0`, which prints only errors.

`estimation_options` A list with options to be used in the estimation call. See `estimate` for the available options.

## Details

The functions of this section combine model initialization and estimation into a single call. They also provide a less verbose interface to the functionality of the package. The functions expect a formula following the specification described in formula, a dataset, and optionally further initialization (see model initialization) and estimation (see model estimation) options.

Each of these functions parses the passed formula, initializes the model specified by the function's name, fit the model to the passed data using the estimation options and returns fitted model.

## Value

The fitted model.

## Functions

- `diseq_basic`: Basic disequilibrium model.
- `diseq_deterministic_adjustment`: Disequilibrium model with deterministic price adjustments.
- `diseq_directional`: Directional disequilibrium model.
- `diseq_stochastic_adjustment`: Disequilibrium model with stochastic price adjustments.
- `equilibrium_model`: Equilibrium model

---

`summaries` *Model and fit summaries*

---

## Description

Methods that summarize models and their estimates.

`market_model`: Prints basic information about the passed model object. In addition to the output of the `show` method, `summary` prints

- the number of observations,

- the number of observations in each equation for models with sample separation, and
- various categories of variables.

market_fit: Prints basic information about the passed model fit. In addition to the output of the model's summary method, the function prints basic estimation results. For a maximum likelihood estimation, the function prints

- the used optimization method,
- the maximum number of allowed iterations,
- the relative convergence tolerance (see optim),
- the convergence status,
- the initializing parameter values,
- the estimated coefficients, their standard errors, Z values, and P values, and
- $-2 \log L$ evaluated at the maximum.

For a linear estimation of the equilibrium system, the function prints the estimation summary provided by systemfit in addition to the model's summary output.

### Usage

```
## S4 method for signature 'market_model'
summary(object)

## S4 method for signature 'market_fit'
summary(object)
```

### Arguments

| | |
|---|---|
| object | An object to be summarized. |

### Functions

- summary,market_model-method: Summarizes the model.
- summary,market_fit-method: Summarizes the model's fit.

### Examples

```
model <- simulate_model(
  "diseq_stochastic_adjustment", list(
    # observed entities, observed time points
    nobs = 500, tobs = 3,
    # demand coefficients
    alpha_d = -0.1, beta_d0 = 9.8, beta_d = c(0.3, -0.2), eta_d = c(0.6, -0.1),
    # supply coefficients
    alpha_s = 0.1, beta_s0 = 5.1, beta_s = c(0.9), eta_s = c(-0.5, 0.2),
    # price equation coefficients
    gamma = 1.2, beta_p0 = 3.1, beta_p = c(0.8)
  ),
```

```
  seed = 556
)

# print model summary
summary(model)
```

---

system_classes *System classes*

---

## Description

System classes

## Details

Classes with data and functionality describing systems of models.

## Functions

- system_base-class: System base class
- system_basic-class: Basic model's system class
- system_deterministic_adjustment-class: Deterministic adjustment model's system class
- system_directional-class: Directional system class
- system_equilibrium-class: Equilibrium model's system class
- system_stochastic_adjustment-class: Stochastic adjustment model's system class

## Slots

demand Demand equation.

supply Supply equation.

correlated_shocks Boolean indicating whether the shock of the equations of the system are correlated.

sample_separation Boolean indicating whether the sample of the system is separated.

quantity_vector A vector with the system's observed quantities.

price_vector A vector with the system's observed prices.

rho Correlation coefficient of demand and supply shocks.

rho1

$$\rho_1 = \frac{1}{\sqrt{1-\rho}}$$

rho2

$$\rho_2 = \rho\rho_1$$

lh Likelihood values for each observation.

gamma Excess demand coefficient.

delta

$$\delta = \gamma + \alpha_d - \alpha_s$$

mu_P

$$\mu_P = \mathrm{E}P$$

var_P

$$V_P = \mathrm{Var}P$$

sigma_P

$$\sigma_P = \sqrt{V_P}$$

h_P

$$h_P = \frac{P - \mu_P}{\sigma_P}$$

lagged_price_vector A vector with the system's observed prices lagged by one date.

mu_Q

$$\mu_Q = \mathrm{E}Q$$

var_Q

$$V_Q = \mathrm{Var}Q$$

sigma_Q

$$\sigma_Q = \sqrt{V_Q}$$

h_Q

$$h_Q = \frac{Q - \mu_Q}{\sigma_Q}$$

rho_QP

$$\rho_{QP} = \frac{\mathrm{Cov}(Q, P)}{\sqrt{\mathrm{Var}Q\mathrm{Var}P}}$$

rho_1QP

$$\rho_{1,QP} = \frac{1}{\sqrt{1 - \rho_{QP}^2}}$$

rho_2QP

$$\rho_{2,QP} = \rho_{QP}\rho_{1,QP}$$

z_QP

$$z_{QP} = \frac{h_Q - \rho_{QP} h_P}{\sqrt{1 - \rho_{QP}^2}}$$

z_PQ

$$z_{PQ} = \frac{h_P - \rho_{PQ} h_Q}{\sqrt{1 - \rho_{PQ}^2}}$$

price_equation Price equation.

zeta

$$\zeta = \sqrt{1 - \rho_{DS}^2 - \rho_{DP}^2 - \rho_{SP}^2 + 2\rho_D P \rho_D S \rho_S P}$$

zeta_DD

$$\zeta_{DD} = 1 - \rho_{SP}^2$$

zeta_DS

$$\zeta_{DS} = \rho_{DS} - \rho_{DP}\rho_{SP}$$

zeta_DP

$$\zeta_{DP} = \rho_{DP} - \rho_{DS}\rho_{SP}$$

zeta_SS

$$\zeta_{SS} = 1 - \rho_{DP}^2$$

zeta_SP

$$\zeta_{SP} = \rho_{SP} - \rho_{DS}\rho_{DP}$$

zeta_PP

$$\zeta_{PP} = 1 - \rho_{DS}^2$$

mu_D

$$\mu_D = \mathrm{E} D$$

var_D

$$V_D = \mathrm{Var} D$$

sigma_D

$$\sigma_D = \sqrt{V_D}$$

mu_S

$$\mu_S = \mathrm{E} S$$

var_S

$$V_S = \mathrm{Var} S$$

sigma_S

$$\sigma_S = \sqrt{V_S}$$

sigma_DP

$$\sigma_{DP} = \mathrm{Cov}(D, P)$$

sigma_DS

$$\sigma_{DS} = \mathrm{Cov}(D, S)$$

sigma_SP

$$\sigma_{SP} = \mathrm{Cov}(S, P)$$

rho_DS

$$\rho_{DS} = \frac{\mathrm{Cov}(D, S)}{\sqrt{\mathrm{Var} D \mathrm{Var} S}}$$

rho_DP

$$\rho_{DP} = \frac{\mathrm{Cov}(D, P)}{\sqrt{\mathrm{Var} D \mathrm{Var} P}}$$

rho_SP

$$\rho_{SP} = \frac{\mathrm{Cov}(S, P)}{\sqrt{\mathrm{Var} S \mathrm{Var} P}}$$

h_D

$$h_D = \frac{D - \mu_D}{\sigma_D}$$

h_S

$$h_S = \frac{S - \mu_S}{\sigma_S}$$

z_DP

$$z_{DP} = \frac{h_D - \rho_{DP} h_P}{\sqrt{1 - \rho_{DP}^2}}$$

z_PD

$$z_{PD} = \frac{h_P - \rho_{PD} h_D}{\sqrt{1 - \rho_{PD}^2}}$$

z_SP

$$z_{SP} = \frac{h_S - \rho_{SP} h_P}{\sqrt{1-\rho_{SP}^2}}$$

z_PS

$$z_{PS} = \frac{h_P - \rho_{PS} h_S}{\sqrt{1-\rho_{PS}^2}}$$

omega_D

$$\omega_D = \frac{h_D \zeta_{DD} - h_S \zeta_{DS} - h_P \zeta_{DP}}{\zeta_{DD}}$$

omega_S

$$\omega_S = \frac{h_S \zeta_{SS} - h_S \zeta_{SS} - h_P \zeta_{SP}}{\zeta_{SS}}$$

w_D

$$w_D = -\frac{h_D^2 - 2 h_D h_P \rho_{DP} + h_P^2}{2\zeta_{SS}}$$

w_S

$$w_S = -\frac{h_S^2 - 2 h_S h_P \rho_{SP} + h_P^2}{2\zeta_{DD}}$$

psi_D

$$\psi_D = \phi\left(\frac{\omega_D}{\zeta}\right)$$

psi_S

$$\psi_S = \phi\left(\frac{\omega_S}{\zeta}\right)$$

Psi_D

$$\Psi_D = 1 - \Phi\left(\frac{\omega_D}{\zeta}\right)$$

Psi_S

$$\Psi_S = 1 - \Phi\left(\frac{\omega_S}{\zeta}\right)$$

g_D

$$g_D = \frac{\psi_D}{\Psi_D}$$

g_S

$$g_S = \frac{\psi_S}{\Psi_S}$$

rho_ds Shadows rho in the diseq_stochastic_adjustment model

rho_dp Correlation of demand and price equations' shocks.

rho_sp Correlation of supply and price equations' shocks.

L_D Likelihood conditional on excess supply.

L_S Likelihood conditional on excess demand.

---

variable_names *Variable name access*

---

## Description

Methods that provide access to the prefixed variable names that the package uses.

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

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

prefixed_price_variable: The price variable name is constructed by concatenating the equation prefix with the name of the price column.

prefixed_control_variables: 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.

prefixed_control_variables: The variance variable is constructed by concatenating the equation prefix with VARIANCE.

prefixed_quantity_variable: The quantity variable name is constructed by concatenating the equation prefix with the name of the quantity column.

lagged_price_variable: The lagged price variable name is constructed by concatenating LAGGED with the price variable name.

price_differences_variable: The price difference variable name is constructed by concatenating the price variable name with DIFF.

## Usage

```
prefixed_const_variable(object)

prefixed_independent_variables(object)

prefixed_price_variable(object)

prefixed_control_variables(object)
```

```
prefixed_variance_variable(object)

prefixed_quantity_variable(object)

## S4 method for signature 'equation_base'
prefixed_const_variable(object)

## S4 method for signature 'equation_base'
prefixed_independent_variables(object)

## S4 method for signature 'equation_base'
prefixed_price_variable(object)

## S4 method for signature 'equation_base'
prefixed_control_variables(object)

## S4 method for signature 'equation_base'
prefixed_variance_variable(object)

## S4 method for signature 'equation_base'
prefixed_quantity_variable(object)

lagged_price_variable(object)

price_differences_variable(object)

## S4 method for signature 'system_base'
lagged_price_variable(object)

## S4 method for signature 'system_base'
price_differences_variable(object)
```

## Arguments

| | |
|---|---|
| `object` | An equation object. |

## Value

The prefixed variable name(s).

## Functions

- `prefixed_const_variable`: Constant coefficient variable name.
- `prefixed_independent_variables`: Independent variable names.
- `prefixed_price_variable`: Price coefficient variable name.
- `prefixed_control_variables`: Control variable names.
- `prefixed_variance_variable`: Variance variable name.

- `prefixed_quantity_variable`: Quantity variable name.
- `lagged_price_variable`: Lagged price variable name.
- `price_differences_variable`: Price differences variable name.

---

## vcov,market_fit-method

*Variance-covariance matrix for a fitted market model.*

---

### Description

Returns the variance-covariance matrix of the estimated coefficients for the fitted model. Specializes the vcov function for fitted market models.

### Usage

```
## S4 method for signature 'market_fit'
vcov(object)
```

### Arguments

| | |
|---|---|
| object | A fitted model object. |

### Value

A matrix of covariances for the estimated model coefficients.

### Examples

```
# estimate a model using the houses dataset
fit <- diseq_deterministic_adjustment(
  HS | RM | ID | TREND ~
  RM + TREND + W + CSHS + L1RM + L2RM + MONTH |
  RM + TREND + W + L1RM + MA6DSF + MA3DHF + MONTH,
  fair_houses(),  correlated_shocks = FALSE,
  estimation_options = list(control = list(maxit = 1e+6)))

# access the variance-covariance matrix
head(vcov(fit))
```

# Index