Package ‘dsge’

May 8, 2026

Title Dynamic Stochastic General Equilibrium Models

Version 1.0.0

Description Specify, solve, and estimate dynamic stochastic general equilibrium (DSGE) models by maximum likelihood and Bayesian methods. Supports both linear models via an equation-based formula interface and nonlinear models via string-based equations with first-order perturbation (linearization around deterministic steady state). Solution uses the method of undetermined coefficients (Klein, 2000 <doi:10.1016/S0165-1889(99)00045-7>). Likelihood evaluated via the Kalman filter. Bayesian estimation uses adaptive Random-Walk Metropolis-Hastings with prior specification. Additional tools include Kalman smoothing, historical shock decomposition, local identification diagnostics, parameter sensitivity analysis, second-order perturbation, occasionally binding constraints, impulse-response functions, forecasting, and robust standard errors.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports grDevices, graphics, stats, numDeriv

Suggests coda, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Mustapha Wasseja Mohammed [aut, cre]

Maintainer Mustapha Wasseja Mohammed <muswaseja@gmail.com>

Repository CRAN

Date/Publication 2026-04-02 07:50:10 UTC

Contents

`bayes_dsge` *Estimate a DSGE Model by Bayesian Methods*

Description

Estimates the parameters of a DSGE model using Random-Walk Metropolis-Hastings (RWMH) with adaptive proposal covariance. Supports both linear models (dsge_model) and nonlinear models (dsgenl_model). For nonlinear models, the steady state is re-solved and the model re-linearized at each candidate parameter vector.

Usage

```
bayes_dsge(
  model,
  data,
  priors,
  chains = 2L,
  iter = 5000L,
  warmup = floor(iter/2),
  thin = 1L,
  proposal_scale = 0.1,
  demean = TRUE,
  seed = NULL
)
```

Arguments

`model`	A `dsge_model` or `dsgenl_model` object.
`data`	A data frame, matrix, or `ts` object containing the observed variables.
`priors`	Named list of `dsge_prior` objects (one per free parameter). Shock standard deviations get default `inv_gamma(0.01, 0.01)` priors unless overridden (names: `"sd_e.shock_name"`).
`chains`	Integer. Number of MCMC chains. Default is 2.
`iter`	Integer. Total iterations per chain (warmup + sampling). Default is 5000.
`warmup`	Integer. Number of warmup iterations. Default is `floor(iter / 2)`.
`thin`	Integer. Thinning interval. Default is 1.
`proposal_scale`	Numeric. Initial proposal standard deviation scale. Default is 0.1.
`demean`	Logical. If `TRUE` (default), observed variables are demeaned before estimation. For nonlinear models, demeaning is not applied because the Kalman filter operates around the parameter-specific steady state.
`seed`	Integer. Random seed for reproducibility. If `NULL`, no seed is set.

Details

The sampler operates in unconstrained space with appropriate transformations (log for positive parameters, logit for unit-interval parameters) and Jacobian corrections. The proposal covariance is adapted during warmup to target approximately 25\

Chains are initialized by drawing from the prior. If any draw yields a non-finite log-posterior, the starting values are jittered until a valid point is found.

For nonlinear models (`dsgenl_model`), each posterior evaluation involves: (1) solving the deterministic steady state at the candidate parameters, (2) computing a first-order linearization, (3) solving the resulting linear system, and (4) evaluating the Kalman filter likelihood. If any stage fails (e.g., steady-state non-convergence, Blanchard-Kahn violation), the proposal is safely rejected. This is computationally more expensive than linear Bayesian estimation.

Value

An object of class `"dsge_bayes"` containing posterior draws and diagnostics. For nonlinear models, the result also includes `solve_failures`, the number of parameter draws where steady-state, linearization, or solution failed.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)

set.seed(42)
z <- numeric(200); for (i in 2:200) z[i] <- 0.8 * z[i-1] + rnorm(1)
dat <- data.frame(y = z)

fit <- bayes_dsge(m, data = dat,
                  priors = list(rho = prior("beta", shape1 = 2, shape2 = 2)),
                  chains = 2, iter = 2000, seed = 1)
summary(fit)
```

`check_identification` *Check Local Identification of DSGE Parameters*

Description

Assesses local identification by computing the Jacobian of the mapping from structural parameters to model-implied autocovariance moments. Uses an SVD decomposition to detect rank deficiency (non-identification) and near-collinearity (weak identification).

Usage

```
check_identification(x, ...)

## S3 method for class 'dsge_fit'
check_identification(x, n_lags = 4L, tol = 1e-06, ...)

## S3 method for class 'dsge_bayes'
check_identification(x, n_lags = 4L, tol = 1e-06, ...)
```

Arguments

`x`	A `dsge_fit` or `dsge_bayes` object.
`...`	Additional arguments (currently unused).
`n_lags`	Integer. Number of autocovariance lags to include in the moment vector. Default is 4.
`tol`	Numeric. Singular values below `tol` times the largest singular value are considered zero (rank deficiency). Default is 1e-6.

Details

The identification check constructs the moment vector $m(\theta) = \text{vec}(\Gamma(0), \Gamma(1), \ldots, \Gamma(K))$ where $\Gamma(k)$ is the autocovariance of observables at lag k, implied by the state-space solution. The Jacobian $J = \partial m / \partial \theta$ is computed numerically.

A parameter is locally identified if the Jacobian has full column rank. If the rank is deficient, some linear combination of parameters cannot be distinguished from the data.

Per-parameter identification strength is measured by the norm of the corresponding Jacobian column: parameters with small column norms have little influence on the moments and may be weakly identified.

The condition number of J flags near-collinearity: a large condition number indicates that some parameter combinations are hard to distinguish.

Value

An object of class `"dsge_identification"` containing:

jacobian The Jacobian matrix (n_moments x n_params).

svd SVD decomposition of the Jacobian.

rank Numerical rank of the Jacobian.

identified Logical: are all parameters locally identified?

singular_values Vector of singular values.

strength Per-parameter identification strength (norm of corresponding Jacobian column).

condition_number Condition number of the Jacobian.

param_names Character vector of parameter names.

summary Data frame with per-parameter diagnostics.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8*z[i-1]+rnorm(1)
fit <- estimate(m, data = data.frame(y = z))
id <- check_identification(fit)
print(id)
```

---

dsgenl_model *Define a Nonlinear DSGE Model*

Description

Constructs a nonlinear DSGE model from string-based equations. Each equation is a character string of the form `"LHS = RHS"`. State equations are detected automatically when the left-hand side is of the form `VAR(+1)`.

Usage

```
dsgenl_model(
  ...,
  observed = character(0),
  unobserved = character(0),
  exo_state,
  endo_state = character(0),
  fixed = list(),
  start = list(),
  ss_guess = NULL,
  ss_function = NULL
)
```

Arguments

`...`	Character strings, each defining one model equation. Use `VAR(+1)` to denote the one-period-ahead value of variable `VAR`.
`observed`	Character vector of observed control variable names.
`unobserved`	Character vector of unobserved control variable names. Default is `character(0)`.
`exo_state`	Character vector of exogenous state variable names. These have shocks attached. The number of exogenous states must equal the number of observed controls.

`endo_state` Character vector of endogenous (predetermined) state variable names. These have no shocks. Default is `character(0)`.

`fixed` Named list of parameter values to hold fixed during estimation.

`start` Named list of starting values for free parameters.

`ss_guess` Named numeric vector of initial guesses for steady-state solving. If `NULL`, defaults to 1 for all variables.

`ss_function` Optional function that computes the steady state analytically. Must accept a named parameter vector and return a named numeric vector of steady-state variable values.

Details

Equations are written in standard mathematical notation with `=` as the equality sign and `VAR(+1)` for leads. For example:

```
"1/C = beta / C(+1) * (alpha * K^(alpha-1) + 1 - delta)"
```

State equations must have exactly one lead variable on the left-hand side (e.g., `"K(+1) = K^alpha - C + (1 - delta) * K"`). Control equations are all remaining equations.

Control equations must be provided in the order matching the controls vector (observed first, then unobserved).

Value

An object of class `"dsgenl_model"`.

Examples

```
# Simple RBC model
rbc <- dsgenl_model(
  "1/C = beta / C(+1) * (alpha * exp(Z) * K^(alpha-1) + 1 - delta)",
  "K(+1) = exp(Z) * K^alpha - C + (1 - delta) * K",
  "Z(+1) = rho * Z",
  observed = "C",
  endo_state = "K",
  exo_state = "Z",
  fixed = list(alpha = 0.33, beta = 0.99, delta = 0.025),
  start = list(rho = 0.9)
)
```

`dsge_model` *Define a Linear DSGE Model*

Description

Constructs a linear DSGE model object from a set of equations. Each equation is wrapped in `obs()`, `unobs()`, or `state()` to indicate the role of the left-hand-side variable.

Usage

```
dsge_model(..., fixed = list(), start = list())
```

Arguments

`...`	Equation specifications created by `obs()`, `unobs()`, and `state()`.
`fixed`	Named list of parameter values to hold fixed (constrained) during estimation.
`start`	Named list of starting values for free parameters.

Details

A linear DSGE model is specified as a system of equations in which variables enter linearly but parameters may enter nonlinearly.

Use `lead(x)` or `E(x)` within formulas to denote the one-period-ahead model-consistent expectation of variable `x`.

The number of exogenous state variables (those with shocks) must equal the number of observed control variables.

Value

An object of class `"dsge_model"` containing:

equations List of parsed equation objects.

variables List with elements `observed`, `unobserved`, `exo_state`, `endo_state`.

parameters Character vector of all parameter names.

free_parameters Character vector of free (estimable) parameter names.

fixed Named list of fixed parameter values.

start Named list of starting values.

Examples

```
# Simple New Keynesian model
nk <- dsge_model(
  obs(p    ~ beta * lead(p) + kappa * x),
  unobs(x ~ lead(x) - (r - lead(p) - g)),
  obs(r    ~ psi * p + u),
  state(u ~ rhou * u),
```

```
  state(g ~ rhog * g),
  fixed = list(beta = 0.96),
  start = list(kappa = 0.1, psi = 1.5, rhou = 0.7, rhog = 0.9)
)
```

`E` *Expectation Operator (Alias for lead)*

Description

A user-friendly alias for `lead(x, 1)`. Represents the one-period-ahead model-consistent expectation of variable `x`.

Usage

```
E(x)
```

Arguments

`x`	A variable name (unquoted) within a DSGE equation formula.

Details

`E(x)` is equivalent to `lead(x, 1)`. The parser translates `E(x)` to `lead(x, 1)` internally.

Value

This function is not meant to be called directly; it always throws an error. It is recognized as a syntactic marker by the equation parser inside `dsge_model()`.

See Also

`lead()`, `dsge_model()`

`estimate` *Estimate a Linear DSGE Model by Maximum Likelihood*

Description

Estimates the parameters of a linear DSGE model by maximizing the log-likelihood computed via the Kalman filter.

Usage

```
estimate(
  model,
  data,
  start = NULL,
  fixed = NULL,
  method = "BFGS",
  control = list(),
  shock_start = NULL,
  demean = TRUE,
  hessian = TRUE
)
```

Arguments

`model`	A `dsge_model` object created by `dsge_model()`.
`data`	A data frame, matrix, or `ts` object containing the observed variables. Column names must match the observed control variable names in the model.
`start`	Named list of starting values for free parameters. Overrides any starting values specified in the model.
`fixed`	Named list of fixed parameter values. Overrides any fixed values specified in the model.
`method`	Optimization method passed to `stats::optim()`. Default is `"BFGS"`.
`control`	Control list passed to `stats::optim()`.
`shock_start`	Named numeric vector of starting values for shock standard deviations. If `NULL`, defaults are set based on data variability.
`demean`	Logical. If `TRUE` (default), observed variables are demeaned before estimation.
`hessian`	Logical. If `TRUE` (default), the Hessian is computed at the solution for standard errors.

Details

The estimator optimizes over the structural parameters and the log standard deviations of the shocks. Shock standard deviations are parameterized in log-space to ensure positivity.

If the optimizer encounters parameter values for which the model is not saddle-path stable, the log-likelihood is set to `-Inf`.

Value

An object of class `"dsge_fit"`.

Examples

```
# Define a simple AR(1) model
m <- dsge_model(
  obs(y ~ z),
```

```
  state(z ~ rho * z),
  start = list(rho = 0.5)
)

# Simulate some data
set.seed(42)
e <- rnorm(200)
z <- numeric(200)
for (i in 2:200) z[i] <- 0.8 * z[i-1] + e[i]
dat <- data.frame(y = z)

fit <- estimate(m, data = dat)
summary(fit)
```

`fitted.dsge_fit` *Fitted values from a DSGE model*

Description

Returns filtered fitted values of observed variables (using all information up to and including time t).

Usage

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

Arguments

`object`	A "dsge_fit" object.
`...`	Additional arguments (currently unused).

Value

A matrix of fitted values with columns named by observed variables.

`forecast` *Forecast from a DSGE Model*

Description

Generic function for forecasting from estimated DSGE models.

Usage

```
forecast(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Additional arguments passed to methods.

Value

A forecast object.

See Also

`forecast.dsge_fit()`

`forecast.dsge_fit` *Forecast from a Fitted DSGE Model*

Description

Produces dynamic multi-step forecasts from a fitted DSGE model. Forecasts are generated by iterating the state-space solution forward from the last filtered state.

Usage

```
## S3 method for class 'dsge_fit'
forecast(object, horizon = 12L, ...)
```

Arguments

`object`	A `dsge_fit` object.
`horizon`	Integer. Number of periods to forecast ahead. Default is 12.
`...`	Additional arguments (currently unused).

Value

An object of class "dsge_forecast" containing:

forecasts Data frame with columns period, variable, value.

horizon The forecast horizon.

states Matrix of forecasted state vectors.

---

geweke_test *Geweke convergence diagnostic*

Description

Computes Geweke's (1992) convergence diagnostic, which compares the means of the first and last portions of each chain using a z-test.

Usage

```
geweke_test(object, ...)
```

Arguments

object	A "dsge_bayes" object.
...	Additional arguments passed to methods (e.g., frac1, frac2 for chain window fractions).

Value

An object of class "dsge_geweke" with z-scores and p-values for each parameter and chain.

---

irf.dsge_bayes *Compute Impulse-Response Functions*

Description

Computes the impulse-response functions (IRFs) from a fitted or solved DSGE model. An IRF traces the dynamic response of control and state variables to a one-standard-deviation shock.

Usage

```
## S3 method for class 'dsge_bayes'
irf(
  x,
  periods = 20L,
  impulse = NULL,
  response = NULL,
  se = TRUE,
  level = 0.95,
  n_draws = 200L,
  ...
)

irf(
  x,
  periods = 20L,
  impulse = NULL,
  response = NULL,
  se = TRUE,
  level = 0.95,
  ...
)
```

Arguments

`x`	A `dsge_fit` or `dsge_solution` object.
`periods`	Integer. Number of periods to compute. Default is 20.
`impulse`	Character vector of shock names. If `NULL` (default), computes IRFs for all shocks.
`response`	Character vector of variable names. If `NULL` (default), computes responses for all variables.
`se`	Logical. If `TRUE` (default) and `x` is a `dsge_fit`, compute delta-method standard errors for IRFs.
`level`	Confidence level for bands. Default is 0.95.
`n_draws`	Integer. Number of posterior draws to use for IRF computation. Default is 200. Set to `NULL` to use all draws.
`...`	Additional arguments passed to methods.

Value

An object of class `"dsge_irf"` containing a data frame with columns: `period`, `impulse`, `response`, `value`, and optionally `se`, `lower`, `upper`.

Methods (by class)

- `irf(dsge_bayes)`: Compute posterior IRFs from a Bayesian DSGE fit. Returns pointwise posterior median and credible bands.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
sol <- solve_dsge(m, params = c(rho = 0.8))
irfs <- irf(sol, periods = 10)
```

---

irf_2nd_order *Generalized IRFs Using Second-Order Approximation*

Description

Computes impulse-response functions using the second-order solution. These differ from first-order IRFs because responses depend on the initial state and shock sign.

Usage

```
irf_2nd_order(sol, shock, size = 1, periods = 40L, initial = NULL)
```

Arguments

`sol`	A `dsge_solution` object with `order` = 2.
`shock`	Character. Name of the shock.
`size`	Numeric. Shock size in standard deviations. Default 1.
`periods`	Integer. Number of IRF periods. Default 40.
`initial`	Named numeric vector of initial state deviations. Default is zero (ergodic mean under second-order).

Value

A data frame with columns: period, variable, response, type.

`lead` *Forward Lead Operator for DSGE Equations*

Description

Marks a variable as a one-period-ahead model-consistent expectation in a DSGE equation formula. This is the core primitive for forward-looking variables.

Usage

```
lead(x, k = 1L)
```

Arguments

`x`	A variable name (unquoted) within a DSGE equation formula.
`k`	Integer lead horizon. Currently only `k = 1` is supported. Default is 1.

Details

`lead(x)` in a DSGE equation represents the expectation of variable `x` one period ahead, conditional on the model. In the literature, this corresponds to $E_t[x_{t+1}]$.

This function is not meant to be called directly. It is recognized by the equation parser inside `dsge_model()`.

Value

This function is not meant to be called directly; it always throws an error. It is recognized as a syntactic marker by the equation parser inside `dsge_model()`.

See Also

`E()` for a user-friendly alias, `dsge_model()`

`linearize` *Linearize a Nonlinear DSGE Model*

Description

Computes a first-order Taylor expansion of the nonlinear model around its deterministic steady state and returns the structural matrices in the canonical linear form.

Usage

```
linearize(model, steady_state, params = NULL)
```

Arguments

model	A dsgenl_model object.
steady_state	A dsgenl_steady_state object or a named numeric vector of steady-state values.
params	Named numeric vector of parameter values. If NULL and steady_state is a dsgenl_steady_state object, uses the parameters stored there.

Value

A list of structural matrices (A0, A1, A2, A3, A4, B0, B1, B2, B3, C, D) plus the steady-state values. A4 captures lead-state coefficients in control equations (often zero).

---

marginal_likelihood *Marginal likelihood estimation*

Description

Estimates the log marginal likelihood using the modified harmonic mean estimator (Geweke, 1999). This provides a practical Bayesian model comparison tool via Bayes factors: BF12 = exp(logML1 - logML2).

Usage

```
marginal_likelihood(object, ...)
```

Arguments

object	A "dsge_bayes" object.
...	Additional arguments passed to methods (e.g., method, tau).

Details

The harmonic mean estimator is known to be numerically unstable in some cases. The modified version (with truncation parameter tau) reduces this instability. Results should be interpreted with caution and compared across models only when both use similar MCMC settings.

Value

An object of class "dsge_marginal_likelihood".

mcmc_diagnostics — *MCMC diagnostic summary*

Description

Comprehensive MCMC diagnostic summary combining ESS, R-hat, Geweke, and acceptance rate information.

Usage

```
mcmc_diagnostics(object, ...)
```

Arguments

`object`	A `"dsge_bayes"` object.
`...`	Additional arguments passed to `geweke_test()`.

Value

An object of class `"dsge_mcmc_summary"`.

model_covariance — *Model-implied covariance and correlation matrices*

Description

Computes the unconditional (model-implied) covariance and correlation matrices of observable variables from a solved or estimated DSGE model. These are the theoretical second moments implied by the model at the given parameter values.

Usage

```
model_covariance(x, variables = NULL, n_lags = 0L, ...)
```

Arguments

`x`	A fitted model (`"dsge_fit"`, `"dsge_bayes"`) or a solved model (`"dsge_solution"`).
`variables`	Character vector of variable names to include. Default `NULL` returns all observable variables.
`n_lags`	Integer. If positive, also compute autocovariances at lags 1, ..., n_lags. Default 0 (contemporaneous only).
`...`	Additional arguments (currently unused).

Value

An object of class "dsge_covariance" containing:

covariance Covariance matrix of selected variables.

correlation Correlation matrix of selected variables.

std_dev Standard deviations (square root of diagonal).

autocovariances List of lagged autocovariance matrices (empty if n_lags = 0).

variables Variable names.

n_lags Number of autocovariance lags computed.

Examples

```
mod <- dsge_model(
  obs(pi ~ beta * lead(pi) + kappa * x),
  unobs(x ~ lead(x) - (r - lead(pi) - g)),
  obs(r ~ psi * pi + u),
  state(u ~ rhou * u),
  state(g ~ rhog * g),
  fixed = list(beta = 0.99),
  start = list(kappa = 0.1, psi = 1.5, rhou = 0.5, rhog = 0.5)
)
p <- list(kappa = 0.1, psi = 1.5, rhou = 0.5, rhog = 0.5)
s <- c(u = 0.5, g = 0.5)
sol <- solve_dsge(mod, params = p, shock_sd = s)
model_covariance(sol)
```

---

obc_constraint *Create an Occasionally Binding Constraint*

Description

Specifies an inequality constraint for use with simulate_occbin.

Usage

```
obc_constraint(variable, type = ">=", bound = 0, shock = NULL)
```

Arguments

variable	Character. Name of the constrained variable (must be a control variable in the model).
type	Character. Either ">=" or "<=".
bound	Numeric. The constraint bound. For models with steady state, this is in levels; for linear models, in deviations.
shock	Character or NULL. Name of the shock channel used to enforce the constraint. If NULL (default), auto-detected as the shock with the largest impact on the constrained variable.

Value

An obc_constraint object.

Examples

```
# Zero lower bound on nominal interest rate
obc_constraint("r", ">=", 0)

# Upper bound on debt ratio
obc_constraint("b", "<=", 0.6)
```

---

obs — *Define an Observed Control Variable Equation*

Description

Wraps a formula to mark it as an equation for an observed control variable in a linear DSGE model.

Usage

```
obs(formula)
```

Arguments

formula	A formula of the form variable ~ expression.

Value

A list with class "dsge_equation" containing the parsed equation and its type.

See Also

unobs(), state(), dsge_model()

parameter_sensitivity *Parameter Sensitivity Analysis for DSGE Models*

Description

Evaluates the sensitivity of key model outputs to one-at-a-time parameter perturbations. For each free parameter, the model is re-solved at theta +/- delta, and changes in the log-likelihood, impulse responses, steady state, and policy matrix are recorded.

Usage

```
parameter_sensitivity(x, ...)

## S3 method for class 'dsge_fit'
parameter_sensitivity(
  x,
  what = c("loglik", "irf"),
  delta = 0.01,
  irf_horizon = 20L,
  ...
)

## S3 method for class 'dsge_bayes'
parameter_sensitivity(
  x,
  what = c("loglik", "irf"),
  delta = 0.01,
  irf_horizon = 20L,
  ...
)
```

Arguments

`x`	A `dsge_fit` or `dsge_bayes` object.
`...`	Additional arguments (currently unused).
`what`	Character vector specifying which outputs to assess. Any subset of `c("loglik", "irf", "steady_state", "policy")`. Default is `c("loglik", "irf")`.
`delta`	Numeric. Perturbation size as a fraction of the parameter value. Default is 0.01 (1 percent).
`irf_horizon`	Integer. Number of IRF periods. Default is 20.

Details

For each parameter θ_j, the model is solved at $\theta_j(1+\delta)$ and $\theta_j(1-\delta)$. The numerical derivative is approximated as a central difference. Elasticities are reported as $(\theta_j/f)\cdot(df/d\theta_j)$, representing the percentage change in the output for a 1 percent change in the parameter.

Value

An object of class "dsge_sensitivity" containing:

loglik Data frame of log-likelihood sensitivities (if requested).

irf Data frame of IRF sensitivities (if requested).

steady_state Data frame of steady-state sensitivities (if requested).

policy Data frame of policy matrix sensitivities (if requested).

param_names Character vector of parameter names.

delta Perturbation fraction used.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8*z[i-1]+rnorm(1)
fit <- estimate(m, data = data.frame(y = z))
sa <- parameter_sensitivity(fit)
print(sa)
```

---

perfect_foresight *Perfect Foresight / Deterministic Transition Paths*

Description

Simulate deterministic transition paths for DSGE models under perfect foresight. Supports temporary shocks, permanent shocks, and initial condition experiments using the linearized solution.

Usage

```
perfect_foresight(
  x,
  shocks = NULL,
  initial = NULL,
  horizon = 40L,
  params = NULL,
  shock_sd = NULL,
  in_sd = FALSE
)
```

Arguments

x
: A solved DSGE model object. Can be a `dsge_solution` (from `solve_dsge`), `dsge_fit` (from `estimate`), or `dsge_bayes` (from `bayes_dsge`).

shocks
: Deterministic shock specification. Can be:

- A named list where each element is a numeric vector giving the shock path for that shock variable (e.g., `list(u = c(1, 0, 0))`). Unnamed periods after the vector ends are treated as zero.
- A matrix of dimension `horizon` x `n_shocks` with column names matching shock names.

Shock values are in units of the shock variable (not standard deviations). If `NULL` (default), no shocks are applied (useful with `initial` to study convergence from displaced initial conditions).

initial
: Named numeric vector of initial state deviations from steady state. Names must match state variable names. Unspecified states default to zero. Default is `NULL` (all states start at steady state).

horizon
: Integer. Number of periods to simulate. Default 40.

params
: Named numeric vector of parameters. Required only when `x` is a model object that has not been solved. For solved objects, parameters are extracted automatically.

shock_sd
: Named numeric vector of shock standard deviations. Used only when `x` is a model object. Default `NULL`.

in_sd
: Logical. If `TRUE`, shock values in `shocks` are interpreted as multiples of the shock standard deviation. Default `FALSE` (shocks are in level units).

Details

The deterministic transition path is computed using the linearized state-space representation:

$$x_{t+1} = Hx_t + M\varepsilon_{t+1}$$

$$y_t = Gx_t$$

where x_t are state deviations from steady state, y_t are control deviations, and ε_t are deterministic shocks.

This uses the first-order linearized solution, so results are approximate for large shocks. For small to moderate shocks, the linearized paths are accurate.

Value

An object of class `"dsge_perfect_foresight"` containing:

states Matrix (horizon x n_states) of state deviations from SS

controls Matrix (horizon x n_controls) of control deviations

state_levels Matrix of state levels (SS + deviation), if SS available

control_levels Matrix of control levels, if SS available

steady_state Named numeric vector of steady-state values

shock_path Matrix (horizon x n_shocks) of applied shocks

initial Named vector of initial state deviations

horizon Integer horizon

state_names Character vector of state names

control_names Character vector of control names

shock_names Character vector of shock names

H State transition matrix used

G Policy matrix used

M Shock impact matrix used

Examples

```
# Simple AR(1) model
mod <- dsge_model(
  obs(p ~ x),
  state(x ~ rho * x),
  start = list(rho = 0.9)
)
sol <- solve_dsge(mod, params = list(rho = 0.9), shock_sd = c(x = 0.01))

# One-time shock at period 1
pf <- perfect_foresight(sol, shocks = list(x = 0.01), horizon = 40)
plot(pf)

# Displaced initial condition
pf2 <- perfect_foresight(sol, initial = c(x = 0.05), horizon = 40)
plot(pf2)
```

`plot.dsge_bayes` *Plot Bayesian DSGE Results*

Description

Produces diagnostic plots for posterior draws from a Bayesian DSGE fit.

Usage

```
## S3 method for class 'dsge_bayes'
plot(
  x,
  type = c("trace", "density", "prior_posterior", "running_mean", "acf", "pairs", "all",
    "irf"),
  pars = NULL,
  ...
)
```

Arguments

x	A dsge_bayes object from bayes_dsge().
type	Character. Plot type: "trace" Trace plots showing MCMC chains for each parameter. All chains are overlaid with distinct colors. Useful for assessing convergence and mixing. "density" Posterior density plots for each parameter with the prior distribution overlaid as a red dashed line. Useful for seeing how much the data updated the prior. "prior_posterior" Dedicated prior-vs-posterior comparison. Same layout as "density" but with the title "Prior vs Posterior" to emphasize the comparison. "running_mean" Cumulative posterior mean by iteration for each parameter. All chains are shown. Useful for assessing whether the chain has settled. "acf" Autocorrelation function plots for each parameter, pooled across chains. Useful for diagnosing slow mixing. "pairs" Pairwise scatter plots of posterior draws (lower triangle) with correlation coefficients (upper triangle) and marginal histograms (diagonal). Useful for detecting parameter correlations. Draws are thinned for readability. "all" A combined diagnostic panel showing trace plot (left) and posterior density with prior (right) side by side for each parameter. "irf" Posterior impulse-response functions with credible bands. Calls irf() internally and plots the result. Additional arguments periods, impulse, response, n_draws, and level are passed through.
pars	Character vector of parameter names to include. If NULL (default), all parameters are shown. Applies to trace, density, prior_posterior, running_mean, acf, pairs, and all. Ignored for irf (use impulse/response instead).
...	Additional arguments. For type = "irf", arguments periods, impulse, response, n_draws, and level are passed to irf.dsge_bayes().

Details

All plot types except pairs and irf handle any number of parameters by paginating across multiple plot pages (up to 4 parameters per page). In interactive sessions, devAskNewPage() is used to prompt between pages.

For the "pairs" plot, at most 1000 draws are used to keep the plot readable. The correlation matrix is printed to the console.

Forecast plotting is not currently supported for Bayesian fits. Use irf() for posterior impulse-response analysis.

Value

Invisibly returns the dsge_bayes object x. Called for the side effect of producing diagnostic plots on the active graphics device.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(42)
z <- numeric(200); for (i in 2:200) z[i] <- 0.8 * z[i-1] + rnorm(1)
fit <- bayes_dsge(m, data = data.frame(y = z),
                  priors = list(rho = prior("beta", shape1 = 2, shape2 = 2)),
                  chains = 2, iter = 2000, seed = 1)
plot(fit, type = "trace")
plot(fit, type = "density")
plot(fit, type = "prior_posterior")
plot(fit, type = "running_mean")
plot(fit, type = "acf")
plot(fit, type = "all")
# Parameter selection
plot(fit, type = "trace", pars = "rho")
```

`plot.dsge_decomposition` *Plot Historical Shock Decomposition*

Description

Creates a stacked bar chart showing the contribution of each structural shock to the observed variables over time.

Usage

```
## S3 method for class 'dsge_decomposition'
plot(x, which = NULL, ...)
```

Arguments

`x`	A `dsge_decomposition` object.
`which`	Which observable(s) to plot. Integer or character. Default is all.
`...`	Additional arguments (currently unused).

Value

No return value, called for the side effect of producing stacked bar charts of the historical shock decomposition on the active graphics device.

plot.dsge_forecast *Plot DSGE Forecasts*

Description

Plots forecast paths for observed variables.

Usage

```
## S3 method for class 'dsge_forecast'
plot(x, ...)
```

Arguments

x	A dsge_forecast object from forecast.dsge_fit().
...	Additional arguments passed to base plotting functions.

Value

No return value, called for the side effect of producing forecast path plots on the active graphics device.

plot.dsge_irf *Plot Impulse-Response Functions*

Description

Creates a multi-panel plot of impulse-response functions with optional confidence bands.

Usage

```
## S3 method for class 'dsge_irf'
plot(x, impulse = NULL, response = NULL, ci = TRUE, ...)
```

Arguments

x	A dsge_irf object from irf().
impulse	Character vector of impulse variables to plot. If NULL, plots all.
response	Character vector of response variables to plot. If NULL, plots all.
ci	Logical. If TRUE (default), plot confidence bands if available.
...	Additional arguments passed to base plotting functions.

Value

No return value, called for the side effect of producing a multi-panel impulse-response plot on the active graphics device.

plot.dsge_occbin *Plot OccBin Simulation Results*

Description

Plot OccBin Simulation Results

Usage

```
## S3 method for class 'dsge_occbin'
plot(x, vars = NULL, compare = TRUE, shade = TRUE, max_panels = 9L, ...)
```

Arguments

x	A dsge_occbin object.
vars	Character vector of variable names to plot. Default: constrained variables plus a few others.
compare	Logical. If TRUE (default), overlay the unconstrained path for comparison.
shade	Logical. If TRUE (default), shade periods where constraints bind.
max_panels	Integer. Maximum panels per page. Default 9.
...	Additional arguments (unused).

Value

No return value, called for the side effect of producing OccBin simulation plots on the active graphics device. Constrained and unconstrained paths are shown, with shaded regions where constraints bind.

plot.dsge_perfect_foresight *Plot Perfect Foresight Transition Paths*

Description

Plot the deterministic transition paths from a perfect_foresight result.

Usage

```
## S3 method for class 'dsge_perfect_foresight'
plot(x, vars = NULL, type = "deviation", max_panels = 9L, ...)
```

Arguments

`x`	A `dsge_perfect_foresight` object.
`vars`	Character vector of variable names to plot. If `NULL` (default), plots all variables with non-trivial paths.
`type`	Character. One of `"deviation"` (default) or `"level"`. Controls whether to plot deviations from SS or levels.
`max_panels`	Integer. Maximum number of panels per plot page. Default 9.
`...`	Additional arguments (currently unused).

Value

No return value, called for the side effect of producing transition path plots on the active graphics device.

`plot.dsge_smoothed` *Plot Smoothed States*

Description

Plot Smoothed States

Usage

```
## S3 method for class 'dsge_smoothed'
plot(x, which = NULL, type = c("states", "fit"), ...)
```

Arguments

`x`	A `dsge_smoothed` object.
`which`	Which states to plot. Integer vector, character vector of state names, or `NULL` (all states).
`type`	Either `"states"` or `"fit"`. `"states"` plots the smoothed state variables; `"fit"` plots the smoothed observables against data.
`...`	Additional arguments passed to `plot()`.

Value

No return value, called for the side effect of producing smoothed state or fit plots on the active graphics device.

policy_matrix *Extract Policy Matrix*

Description

Returns the policy matrix G from a fitted or solved DSGE model. The policy matrix maps state variables to control variables: $y_t = Gx_t$.

Usage

```
policy_matrix(x, se = TRUE, level = 0.95)
```

Arguments

x	A dsge_fit or dsge_solution object.
se	Logical. If TRUE (default) and x is a dsge_fit, compute delta-method standard errors.
level	Confidence level for intervals. Default is 0.95.

Value

If se = FALSE, returns the G matrix. If se = TRUE, returns a list with matrix, se, lower, upper, and a data frame table.

posterior_predictive *Posterior predictive check*

Description

Simulates data from the posterior predictive distribution and compares summary statistics (variance, autocorrelation) with the observed data. This helps assess whether the estimated model can reproduce key features of the data.

Usage

```
posterior_predictive(object, ...)
```

Arguments

object	A "dsge_bayes" object.
...	Additional arguments passed to methods (e.g., n_draws, statistics, seed).

Value

An object of class "dsge_ppc" with posterior predictive distributions and p-values for each statistic.

predict.dsge_fit — *Predict Method for DSGE Models*

Description

Computes one-step-ahead predictions or filtered state estimates from a fitted DSGE model.

Usage

```
## S3 method for class 'dsge_fit'
predict(
  object,
  type = c("observed", "state"),
  method = c("onestep", "filter"),
  newdata = NULL,
  ...
)
```

Arguments

`object`	A `dsge_fit` object.
`type`	Character. `"observed"` (default) for predicted values of observed control variables, or `"state"` for filtered latent state estimates.
`method`	Character. `"onestep"` (default) for one-step-ahead predictions using only past data, or `"filter"` for filtered estimates using past and contemporaneous data.
`newdata`	Optional new data for prediction. If `NULL`, uses the estimation data.
`...`	Additional arguments (currently unused).

Value

A matrix of predictions or state estimates.

prediction_accuracy — *Prediction accuracy measures for a fitted DSGE model*

Description

Computes root mean squared error (RMSE), mean absolute error (MAE), and mean error (bias) of one-step-ahead predictions.

Usage

```
prediction_accuracy(object, ...)
```

Arguments

object	A "dsge_fit" object.
...	Additional arguments (currently unused).

Value

An object of class "dsge_prediction_accuracy" containing:

rmse Named numeric vector of RMSE by variable.

mae Named numeric vector of MAE by variable.

bias Named numeric vector of mean error by variable.

n Number of observations.

variables Variable names.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8 * z[i-1] + rnorm(1)
fit <- estimate(m, data = data.frame(y = z))
acc <- prediction_accuracy(fit)
print(acc)
```

---

prediction_interval *Prediction intervals for DSGE models*

Description

Computes point predictions and prediction intervals using the one-step-ahead innovation variance from the Kalman filter.

Usage

```
prediction_interval(object, level = 0.95, ...)
```

Arguments

object	A "dsge_fit" object.
level	Confidence level for prediction intervals (default 0.95).
...	Additional arguments passed to predict().

Value

An object of class "dsge_prediction_interval" with components fit, lower, upper, se, level, and variables.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8 * z[i-1] + rnorm(1)
fit <- estimate(m, data = data.frame(y = z))
pi <- prediction_interval(fit, level = 0.95)
print(pi)
```

---

prior — *Specify a Prior Distribution*

Description

Creates a prior distribution object for use in Bayesian DSGE estimation.

Usage

```
prior(distribution, ...)
```

Arguments

distribution	Character string specifying the distribution family. One of "normal", "beta", "gamma", "uniform", "inv_gamma".
...	Distribution parameters (see Details).

Details

Distribution parameterizations:

normal mean, sd

beta shape1, shape2 (alpha, beta parameters)

gamma shape, rate

uniform min, max

inv_gamma shape, scale — density: $p(x) \propto x^{-(shape+1)} \exp(-scale/x)$

Value

An object of class "dsge_prior".

Examples

```
prior("normal", mean = 0, sd = 1)
prior("beta", shape1 = 2, shape2 = 2)
prior("inv_gamma", shape = 0.01, scale = 0.01)
```

---

prior_posterior_update

Prior-Posterior Update Diagnostics

Description

Computes diagnostics measuring how informative the data was for each parameter, by comparing the posterior distribution to the prior.

Usage

```
prior_posterior_update(x, ...)

## S3 method for class 'dsge_bayes'
prior_posterior_update(x, ...)
```

Arguments

`x`	A `dsge_bayes` object.
`...`	Additional arguments (currently unused).

Details

For each estimated parameter, the following diagnostics are computed:

sd_ratio Ratio of posterior SD to prior SD. Values near 1 indicate the data was uninformative (posterior tracks the prior). Values much less than 1 indicate strong data information.

mean_shift Absolute difference between posterior mean and prior mean, measured in units of prior SD. Large shifts indicate the data substantially updated beliefs.

update Classification: "strong" if sd_ratio < 0.5 or mean_shift > 2; "moderate" if sd_ratio < 0.8 or mean_shift > 1; "weak" otherwise (posterior closely resembles the prior).

The SD ratio is the primary indicator of data informativeness. A parameter with sd_ratio close to 1 and small mean_shift is effectively determined by the prior, not the data.

Value

An object of class "dsge_prior_posterior" containing:

summary Data frame with per-parameter diagnostics including prior mean/sd, posterior mean/sd, SD ratio, mean shift, and update classification.

param_names Character vector of parameter names.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8*z[i-1]+rnorm(1)
fit <- bayes_dsge(m, data = data.frame(y = z),
  priors = list(rho = prior("beta", shape1 = 2, shape2 = 2)),
  chains = 2, iter = 2000, seed = 1)
pp <- prior_posterior_update(fit)
print(pp)
```

`residuals.dsge_fit` *Residuals from a fitted DSGE model*

Description

Returns one-step-ahead prediction errors.

Usage

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

Arguments

`object`	A `dsge_fit` object.
`...`	Additional arguments (currently unused).

Value

A matrix of prediction errors.

`robust_vcov` *Robust (sandwich) variance-covariance matrix*

Description

Computes the sandwich (Huber-White) variance-covariance matrix for ML-estimated DSGE model parameters. This provides standard errors that are robust to model misspecification.

Usage

```
robust_vcov(object, ...)
```

Arguments

`object`	A `"dsge_fit"` object estimated via ML.
`...`	Additional arguments passed to methods (e.g., `step` for numerical gradient step size).

Details

The sandwich estimator is: V_robust = inv(H) B inv(H), where H is the Hessian of the negative log-likelihood and B is the outer product of the per-observation score vectors.

Value

An object of class `"dsge_robust_vcov"` containing:

vcov Robust variance-covariance matrix.

se Robust standard errors.

se_conventional Conventional (Hessian-based) standard errors for comparison.

param_names Parameter names.

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8 * z[i-1] + rnorm(1)
fit <- estimate(m, data = data.frame(y = z))
rv <- robust_vcov(fit)
print(rv)
```

shock_decomposition *Historical Shock Decomposition*

Description

Decomposes the observed variables into the contributions of each structural shock. At each time t, the observed deviation from steady state is written as a sum of contributions from current and past shocks plus the initial condition contribution.

Usage

```
shock_decomposition(x, ...)

## S3 method for class 'dsge_fit'
shock_decomposition(x, ...)

## S3 method for class 'dsge_bayes'
shock_decomposition(x, ...)
```

Arguments

`x`	A `dsge_fit` or `dsge_bayes` object.
`...`	Additional arguments (currently unused).

Details

The state-space solution gives:

$$x_t = H^t x_0 + \sum_{j=1}^{t} H^{t-j} M \varepsilon_j$$

The historical decomposition partitions the observed variables $y_t = Zx_t$ into the contribution of each structural shock $\varepsilon_j^{(k)}$ accumulated through the propagation mechanism. The sum of all contributions (including the initial condition term) reproduces the smoothed observables exactly.

Value

An object of class `"dsge_decomposition"` containing:

decomposition A 3D array with dimensions `[T, n_obs, n_shocks + 1]`. The last slice contains the initial condition contribution.

obs_names Character vector of observed variable names.

shock_names Character vector of shock names (plus "initial").

observed T x n_obs matrix of observed data (deviations).

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
e <- rnorm(100)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8*z[i-1]+e[i]
fit <- estimate(m, data = data.frame(y = z))
hd <- shock_decomposition(fit)
plot(hd)
```

`simulate_2nd_order` *Simulate Using Second-Order Approximation (Pruned)*

Description

Simulates sample paths using the pruned second-order approximation following Kim, Kim, Schaumburg, and Sims (2008).

Usage

```
simulate_2nd_order(sol, n = 200L, n_burn = 100L, seed = NULL)
```

Arguments

`sol`	A `dsge_solution` object with `order = 2`.
`n`	Integer. Number of periods to simulate.
`n_burn`	Integer. Burn-in periods to discard. Default 100.
`seed`	Random seed.

Value

A list with `states`, `controls`, `state_levels`, `control_levels` matrices (n x n_vars).

simulate_occbin *Simulate with Occasionally Binding Constraints*

Description

Computes deterministic transition paths under piecewise-linear occasionally binding constraints using an iterative shadow-shock method.

Usage

```
simulate_occbin(
  x,
  constraints,
  shocks = NULL,
  initial = NULL,
  horizon = 40L,
  max_iter = 100L,
  tol = 1e-08,
  in_sd = FALSE
)
```

Arguments

x	A solved DSGE model object (dsge_solution, dsge_fit, or dsge_bayes).
constraints	A list of constraints. Each element can be: • An obc_constraint object (from obc_constraint()) • A character string like "r >= 0" (parsed automatically)
shocks	Deterministic shock specification (as in perfect_foresight). Named list or matrix.
initial	Named numeric vector of initial state deviations from steady state. Default NULL (start at SS).
horizon	Integer. Simulation horizon. Default 40.
max_iter	Integer. Maximum OccBin iterations. Default 100.
tol	Numeric. Convergence tolerance for shadow shocks. Default 1e-8.
in_sd	Logical. If TRUE, shock values are in standard deviations. Default FALSE.

Details

The algorithm iteratively:

1. Simulates the path with current shadow shocks
2. Identifies periods where constraints are violated
3. Computes shadow shocks to enforce constraints at binding periods
4. Removes shadow shocks at non-binding periods

5. Repeats until the binding regime stabilises

This captures the feedback effect of constraint enforcement on future dynamics through the state transition.

Value

An object of class "dsge_occbin" containing:

states Matrix of state deviations (constrained path)

controls Matrix of control deviations (constrained path)

states_unc Matrix of state deviations (unconstrained path)

controls_unc Matrix of control deviations (unconstrained path)

binding Logical matrix (horizon x n_constraints) of binding indicators

shadow_shocks Matrix of shadow shocks applied

n_iter Number of OccBin iterations to convergence

converged Logical: did the algorithm converge?

constraints List of constraint objects

steady_state Steady state values if available

horizon Simulation horizon

state_names State variable names

control_names Control variable names

Examples

```
# Simple NK model with ZLB
nk <- dsge_model(
  obs(pi ~ beta * lead(pi) + kappa * x),
  unobs(x ~ lead(x) - (r - lead(pi) - g)),
  obs(r ~ psi * pi + u),
  state(u ~ rhou * u),
  state(g ~ rhog * g),
  fixed = list(beta = 0.99, kappa = 0.1, psi = 1.5),
  start = list(rhou = 0.5, rhog = 0.5)
)
sol <- solve_dsge(nk, params = list(rhou = 0.5, rhog = 0.5),
                  shock_sd = c(u = 0.5, g = 0.5))
obc <- simulate_occbin(sol,
  constraints = list("r >= 0"),
  shocks = list(g = -0.05),
  horizon = 40)
plot(obc)
```

`smooth_shocks` *Extract Smoothed Structural Shocks*

Description

Recovers the structural shocks from the smoothed states using the state transition equation: $\hat{\varepsilon}_{t+1} = M^+(x_{t+1|T} - Hx_{t|T})$ where M^+ is the Moore-Penrose pseudo-inverse of M.

Usage

```
smooth_shocks(x, ...)

## S3 method for class 'dsge_fit'
smooth_shocks(x, ...)

## S3 method for class 'dsge_bayes'
smooth_shocks(x, ...)
```

Arguments

`x`	A `dsge_fit` or `dsge_bayes` object.
`...`	Additional arguments (currently unused).

Details

From the state transition $x_{t+1} = Hx_t + M\varepsilon_{t+1}$, the smoothed innovation is $x_{t+1|T} - Hx_{t|T}$. The structural shocks are recovered by projecting onto M: $\hat{\varepsilon}_{t+1} = (M'M)^{-1}M'(x_{t+1|T} - Hx_{t|T})$.

Value

An object of class `"dsge_smoothed_shocks"` containing:

shocks (T-1) x n_shocks matrix of smoothed structural shocks.

shock_names Character vector of shock names.

`smooth_states` *Smoothed State Estimates from an Estimated DSGE Model*

Description

Computes the Rauch-Tung-Striebel (RTS) smoother to produce optimal state estimates using all available observations. Compared to the filtered states (which only use past data), smoothed states also incorporate future observations.

Usage

```
smooth_states(x, ...)

## S3 method for class 'dsge_fit'
smooth_states(x, ...)

## S3 method for class 'dsge_bayes'
smooth_states(x, ...)
```

Arguments

x	A dsge_fit or dsge_bayes object.
...	Additional arguments (currently unused).

Details

The smoother uses the state-space representation:

$$x_{t+1} = Hx_t + M\varepsilon_{t+1}$$

$$y_t = Zx_t$$

where $Z = D \cdot G$. The smoothed states are the expectation of the state vector conditional on all observations: $x_{t|T} = E[x_t|y_1, \ldots, y_T]$.

For Bayesian models, the smoother is evaluated at the posterior mean.

Value

An object of class "dsge_smoothed" containing:

smoothed_states T x n_s matrix of smoothed state estimates.

filtered_states T x n_s matrix of filtered state estimates.

smoothed_obs T x n_obs matrix of smoothed observable fits.

residuals T x n_obs matrix of observation residuals.

state_names Character vector of state variable names.

obs_names Character vector of observed variable names.

steady_state Steady-state values (if available).

Examples

```
m <- dsge_model(
  obs(y ~ z),
  state(z ~ rho * z),
  start = list(rho = 0.5)
)
set.seed(1)
e <- rnorm(100)
z <- numeric(100); for (i in 2:100) z[i] <- 0.8 * z[i-1] + e[i]
```

```
fit <- estimate(m, data = data.frame(y = z))
sm <- smooth_states(fit)
```

---

solve_dsge   *Solve a Linear or Linearized DSGE Model*

Description

Computes the state-space solution of a DSGE model using the Klein (2000) method. Accepts both linear models (dsge_model) and nonlinear models (dsgenl_model). For nonlinear models, the steady state is computed and the model is linearized automatically.

Usage

```
solve_dsge(model, params = NULL, shock_sd = NULL, tol = 1e-06, order = 1L)
```

Arguments

model	A dsge_model or dsgenl_model object.
params	Named numeric vector of parameter values. If NULL, uses the model's fixed and start values.
shock_sd	Named numeric vector of shock standard deviations. If NULL, defaults to 1 for all shocks.
tol	Tolerance for classifying eigenvalues as stable (|lambda| < 1 + tol). Default is 1e-6.
order	Integer. Approximation order: 1 (default) for first-order, 2 for second-order perturbation. Second-order is only available for nonlinear models (dsgenl_model).

Details

The method forms a companion system from the structural matrices and solves via undetermined coefficients iteration. Saddle-path stability requires that all eigenvalues of H have modulus less than 1.

For nonlinear models, the solver first computes the deterministic steady state, then linearizes the model via first-order Taylor expansion, and finally solves the resulting linear system.

Value

An object of class "dsge_solution" containing:

G Policy matrix (n_controls x n_states).

H State transition matrix (n_states x n_states).

M Shock coefficient matrix (n_states x n_shocks).

D Observation selection matrix.

eigenvalues Complex vector of eigenvalues.

stable Logical: is the system saddle-path stable?

n_stable Number of stable eigenvalues.

params The parameter values used.

model Reference to the model object.

---

stability *Check Stability of DSGE Model*

Description

Checks saddle-path stability of the model by examining eigenvalues. A model is stable when the number of eigenvalues with modulus less than 1 equals the number of state variables.

Usage

```
stability(x)
```

Arguments

x	A `dsge_fit` or `dsge_solution` object.

Value

A list of class `"dsge_stability"` with:

stable Logical: is the system saddle-path stable?

eigenvalues Complex eigenvalue vector.

moduli Moduli of eigenvalues.

classification Character vector: "stable" or "unstable" for each.

n_stable Number of stable eigenvalues.

n_states Number of state variables (required stable count).

state — *Define a State Variable Equation*

Description

Wraps a formula to mark it as an equation for a state variable in a linear DSGE model. State equations describe the evolution of state variables one period ahead.

Usage

```
state(formula, shock = TRUE)
```

Arguments

`formula`	A formula of the form `variable ~ expression`. The left-hand side variable represents the one-period lead of the state.
`shock`	Logical. If `TRUE` (default), an unobserved shock is attached to this state equation (exogenous state). If `FALSE`, no shock is attached (endogenous state / deterministic state).

Value

A list with class `"dsge_equation"` containing the parsed equation and its type.

See Also

`obs()`, `unobs()`, `dsge_model()`

steady_state — *Solve for the Deterministic Steady State*

Description

Finds the steady-state values of all model variables by solving the system of nonlinear equations with all leads set equal to current values.

Usage

```
steady_state(model, ...)

## S3 method for class 'dsgenl_model'
steady_state(
  model,
  params = NULL,
  guess = NULL,
```

```
  maxiter = 200L,
  tol = 1e-10,
  ...
)
```

Arguments

model	A dsgenl_model object.
...	Additional arguments (currently unused).
params	Named numeric vector of parameter values. If NULL, uses the model's fixed and start values.
guess	Named numeric vector of initial guesses for variable values. Overrides the model's ss_guess.
maxiter	Maximum Newton-Raphson iterations. Default is 200.
tol	Convergence tolerance. Default is 1e-10.

Value

An object of class "dsgenl_steady_state" with:

values Named numeric vector of steady-state values.

residuals Equation residuals at the solution.

params Parameter values used.

converged Logical: did the solver converge?

iterations Number of iterations used.

---

summary.dsge_perfect_foresight

Summary of Perfect Foresight Transition

Description

Summary of Perfect Foresight Transition

Usage

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

Arguments

object	A dsge_perfect_foresight object.
...	Additional arguments (unused).

Value

Invisibly returns the dsge_perfect_foresight object. Called for the side effect of printing impact effects, peak deviations, and convergence diagnostics to the console.

transition_matrix *Extract State Transition Matrix*

Description

Returns the transition matrix H from a fitted or solved DSGE model. The transition matrix describes state evolution: $x_{t+1} = Hx_t + M\varepsilon_{t+1}$.

Usage

```
transition_matrix(x, se = TRUE, level = 0.95)
```

Arguments

`x`	A `dsge_fit` or `dsge_solution` object.
`se`	Logical. If `TRUE` (default) and `x` is a `dsge_fit`, compute delta-method standard errors.
`level`	Confidence level for intervals. Default is 0.95.

Value

Same structure as `policy_matrix()`.

unobs *Define an Unobserved Control Variable Equation*

Description

Wraps a formula to mark it as an equation for an unobserved control variable in a linear DSGE model.

Usage

```
unobs(formula)
```

Arguments

`formula`	A formula of the form `variable ~ expression`.

Value

A list with class `"dsge_equation"` containing the parsed equation and its type.

See Also

`obs()`, `state()`, `dsge_model()`

vcov.dsge_fit *Robust vcov via vcov generic*

Description

When type = "robust" is passed to vcov(), returns the sandwich variance-covariance matrix.

Usage

```
## S3 method for class 'dsge_fit'
vcov(object, type = c("conventional", "robust"), ...)
```

Arguments

object	A "dsge_fit" object.
type	Character. "conventional" (default) or "robust".
...	Passed to robust_vcov() when type is "robust".

Value

Variance-covariance matrix.

Index