hetid Package Manual

June 24, 2025

Title Identification Through Heteroskedasticity a La Lewbel (2012)

Version 0.1.0

Author Fernando Duarte [aut, cre]
(<https://orcid.org/0000-0002-8495-6936>)

Maintainer Fernando Duarte <fernando_duarte@brown.edu>

Description Implements the identification through heteroskedasticity
method of Lewbel (2012) for time-series models with endogenous
regressors. Provides tools for estimation and inference when
traditional instruments are not available.

License MIT + file LICENSE

URL https://fernando-duarte.github.io/heteroskedasticity_identification/,
https://github.com/fernando-duarte/heteroskedasticity_identification

BugReports
https://github.com/fernando-duarte/heteroskedasticity_identification/issues

Depends R (>= 4.1)

Imports boot,
dplyr,
furrr,
future,
ggplot2,
gmm,
parallel,
purrr,
rlang,
stats,
tidyr,
tsmethods,
xts

Suggests AER,
covr,
curl (>= 5.0.0),
desc,
goodpractice,
haven,
ivreg,
jsonlite,

knitr,
lintr,
oysteR,
pkgdepends,
REndo (>= 2.4.0),
rmarkdown,
RStata,
sandwich,
testthat (>= 3.0.0),
tsgarch,
tsmarch,
uuid,
withr,
zoo

VignetteBuilder knitr

Config/build/vignette-compression both

Config/Needs/website pkgdown

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Contents

`adjust_se_for_df` *Adjust standard errors for degrees of freedom*

Description

Adjust standard errors for degrees of freedom

Usage

```
adjust_se_for_df(se, n, k, df_adjust = "asymptotic")
```

Arguments

`se`	Asymptotic standard error
`n`	Sample size
`k`	Number of parameters
`df_adjust`	Character string: "asymptotic" (default) or "finite"

Value

Adjusted standard error

`analyze_bootstrap_results` *Analyze Bootstrap Results*

Description

Analyzes and displays bootstrap standard error results for identified sets.

Usage

```
analyze_bootstrap_results(results_main, bootstrap_demo, config, verbose = TRUE)
```

Arguments

`results_main`	Data.frame. Main simulation results.
`bootstrap_demo`	Data.frame. Bootstrap demonstration results.
`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A data.frame with bootstrap examples.

Note

For enhanced table formatting in verbose output, install the knitr package.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_main <- run_main_simulation(config, seeds)
bootstrap_demo <- run_bootstrap_demonstration(config, seeds)
bootstrap_analysis <- analyze_bootstrap_results(
  results_main, bootstrap_demo, config
)

## End(Not run)
```

---

analyze_main_results *Analyze Main Simulation Results*

Description

Provides analysis of the main Monte Carlo simulation results, including performance metrics for point estimators and set identification.

Usage

```
analyze_main_results(results, config, verbose = TRUE)
```

Arguments

`results`	Data.frame. Results from run_main_simulation().
`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A list containing summary tables and statistics.

Note

For enhanced table formatting in verbose output, install the knitr package.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results <- run_main_simulation(config, seeds)
analysis <- analyze_main_results(results, config)

## End(Not run)
```

analyze_sample_size_results

Analyze Sample Size Results

Description

Analyzes consistency of estimators across different sample sizes.

Usage

```
analyze_sample_size_results(results_by_n, config, verbose = TRUE)
```

Arguments

`results_by_n`	Data.frame. Results from run_sample_size_analysis().
`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A data.frame with sample size analysis.

Note

For enhanced table formatting in verbose output, install the knitr package.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_by_n <- run_sample_size_analysis(config, seeds)
size_analysis <- analyze_sample_size_results(results_by_n, config)

## End(Not run)
```

analyze_sensitivity_results

Analyze Sensitivity Results

Description

Analyzes sensitivity of results to heteroscedasticity strength.

Usage

```
analyze_sensitivity_results(results_by_delta, config, verbose = TRUE)
```

Arguments

`results_by_delta`	Data.frame. Results from run_sensitivity_analysis().
`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A data.frame with sensitivity analysis.

Note

For enhanced table formatting in verbose output, install the knitr package.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_by_delta <- run_sensitivity_analysis(config, seeds)
sensitivity_analysis <- analyze_sensitivity_results(
  results_by_delta, config
)

## End(Not run)
```

`calculate_lewbel_bounds`

Calculate Set Identification Bounds for Lewbel Estimator

Description

Computes set identification bounds for the endogenous parameter under a relaxed covariance restriction. Optionally computes bootstrap standard errors for the bounds.

Usage

```
calculate_lewbel_bounds(
  data,
  tau,
  compute_se = FALSE,
  b_reps = .hetid_const("DEFAULT_BOOTSTRAP_REPS"),
  df_adjust = "asymptotic"
)
```

Arguments

`data`	Data.frame. Dataset containing Y1, Y2, Xk, Z variables.
`tau`	Numeric. Relaxation parameter for covariance restriction (0 <= tau < 1). When tau = 0, gives point identification.
`compute_se`	Logical. Whether to compute bootstrap standard errors (default: FALSE).
`b_reps`	Integer. Number of bootstrap replications if compute_se = TRUE (default: 100).
`df_adjust`	Character. Method for degrees of freedom adjustment: • "asymptotic": No adjustment (default) • "finite": Finite sample adjustment using HC2 formula

Details

Note: This parameter currently only affects the interpretation of bootstrap SEs, not the bounds calculation itself.

Under the relaxed assumption |Corr(Z, $\epsilon_1\epsilon_2$)| <= tau |Corr(Z, ϵ_2^2)|, the parameter gamma_1 is set-identified. The bounds are computed as the real roots of a quadratic equation in gamma_1.

Value

A list containing:

- bounds: Numeric vector of length 2 with lower and upper bounds
- se: Numeric vector of length 2 with bootstrap standard errors (if requested)

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

Examples

```
## Not run:
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0, delta_het = 1.2
)
# TODO: Update generate_lewbel_data call if its return columns change to
# snake_case
data <- generate_lewbel_data(1000, params)

# Point identification (tau = 0)
bounds_point <- calculate_lewbel_bounds(data, tau = 0)

# Set identification with bootstrap SE
bounds_set <- calculate_lewbel_bounds(
  data,
  tau = 0.2, compute_se = TRUE, b_reps = 100
)

## End(Not run)
```

`compare_gmm_2sls` *Compare GMM and 2SLS Estimates for Lewbel Model*

Description

Compares GMM estimates with traditional 2SLS (Lewbel) estimates for the triangular system.

Usage

```
compare_gmm_2sls(
  data,
  y1_var = "Y1",
  y2_var = "Y2",
  x_vars = "Xk",
  add_intercept = TRUE,
  true_gamma1 = NA_real_,
  gmm_args = list(),
  tsls_sim_config = list(),
  verbose = TRUE
)
```

Arguments

`data`	Data frame containing all required variables. Must include the dependent variables and any exogenous regressors specified in the model.
`y1_var`	Character. Name of the first dependent variable (default: "Y1").
`y2_var`	Character. Name of the second dependent variable/endogenous regressor (default: "Y2").
`x_vars`	Character vector. Names of exogenous variables (default: "Xk"). For 2SLS via run_single_lewbel_simulation, it assumes a single "Xk" if default simulation parameters are used. For GMM, can be multiple. This function will try to match behavior. If multiple x_vars are given, the 2SLS part might be less comparable if its underlying data generating process assumes one X.
`add_intercept`	Logical. Whether to add an intercept for GMM (default: TRUE). 2SLS via run_single_lewbel_simulation typically includes an intercept.
`true_gamma1`	Numeric. Optional true value of gamma1 for bias calculation.
`gmm_args`	List. Additional arguments passed to `lewbel_gmm`.
`tsls_sim_config`	List. Parameters to override in the default config for `run_single_lewbel_simulation`. The `sample_size` will be set to `nrow(data)`. `lewbel_x_vars` in this config should match `x_vars` here.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A data frame comparing estimates, standard errors, and test statistics.

See Also

`lewbel_gmm` for GMM estimation. `run_single_lewbel_simulation` for 2SLS estimation.

Examples

```
## Not run:
# Generate data
params_dgp <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0, delta_het = 1.2
)
data_comp <- generate_lewbel_data(1000, params_dgp) # Generates Y1, Y2, Xk, Z

# Compare (assuming Xk is the exogenous variable for both)
comparison <- compare_gmm_2sls(data_comp, true_gamma1 = params_dgp$gamma1)
print(comparison)

# Example with multiple X for GMM, 2SLS might be less direct comparison
params_multi <- list(
  beta1_0 = 0.5, beta1_1 = c(1.5, 0.2), gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = c(-1.0, 0.3),
  alpha1 = -0.5, alpha2 = 1.0, delta_het = 1.2
)
data_multi_x <- generate_lewbel_data(1000, params_multi, n_x = 2) # Y1,Y2,X1,X2,Z1,Z2
comparison_multi <- compare_gmm_2sls(data_multi_x,
  x_vars = c("X1", "X2"),
  true_gamma1 = params_multi$gamma1,
  tsls_sim_config = list(
    lewbel_x_vars = c("X1", "X2") # Hypothetical
    # Note: run_single_lewbel_simulation's internal
    # Data generating process might not easily map to this if it assumes 1 Xk.
    # This part is more illustrative for GMM side.
  )
)
print(comparison_multi)

## End(Not run)
```

compare_rigobon_methods

Compare Rigobon with Other Methods

Description

Compares Rigobon's regime-based identification with OLS and standard Lewbel identification (if applicable).

Usage

```
compare_rigobon_methods(
  data,
  true_gamma1 = NULL,
  methods = c("OLS", "Rigobon", "Lewbel"),
  verbose = TRUE
)
```

Arguments

`data`	Data.frame. Must contain required variables for all methods.
`true_gamma1`	Numeric. Optional. True value of the endogenous parameter.
`methods`	Character vector. Methods to compare (default: c("OLS", "Rigobon", "Lewbel")).
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A data frame comparing the methods with columns for estimates, standard errors, bias (if true value provided), and method-specific diagnostics.

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792. doi:10.1162/003465303772815727

See Also

`run_rigobon_estimation`, `run_single_lewbel_simulation`

Examples

```
## Not run:
# Generate data with known true parameter
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0,
  regime_probs = c(0.4, 0.6),
  sigma2_regimes = c(1.0, 2.5)
)
data <- generate_rigobon_data(1000, params)

# Compare methods
comparison <- compare_rigobon_methods(data, true_gamma1 = params$gamma1)

## End(Not run)
```

`create_default_config` *Create Default Configuration for Lewbel Monte Carlo Simulations*

Description

Creates a comprehensive configuration list with all parameters needed for running Lewbel (2012) Monte Carlo simulations.

Usage

```
create_default_config(
  num_simulations = .hetid_const("DEFAULT_NUM_SIMULATIONS"),
  main_sample_size = .hetid_const("DEFAULT_MAIN_SAMPLE_SIZE"),
  sample_sizes = .hetid_const("DEFAULT_SAMPLE_SIZES"),
  delta_het = 0.8,
  delta_het_values = c(0.4, 0.8, 1.2),
  n_reps_by_n = .hetid_const("DEFAULT_BOOTSTRAP_REPS"),
  n_reps_by_delta = .hetid_const("DEFAULT_BOOTSTRAP_REPS"),
  bootstrap_reps = .hetid_const("DEFAULT_BOOTSTRAP_REPS"),
  bootstrap_subset_size = .hetid_const("DEFAULT_BOOTSTRAP_SUBSET_SIZE"),
  bootstrap_demo_size = .hetid_const("DEFAULT_BOOTSTRAP_DEMO_SIZE"),
  beta1_0 = 0.5,
  beta1_1 = 1.5,
  gamma1 = -0.8,
  beta2_0 = 1,
  beta2_1 = -1,
  alpha1 = -0.5,
  alpha2 = 1,
  tau_set_id = 0.2,
  endog_var_name = "Y2",
  exog_var_names = "Xk",
  df_adjust = "asymptotic"
)
```

Arguments

`num_simulations`	Integer. Number of main simulation runs (default: 1000).
`main_sample_size`	Integer. Primary sample size for main results (default: 1000).
`sample_sizes`	Integer vector. Sample sizes for consistency analysis (default: c(500, 1000, 2000)).
`delta_het`	Numeric. Heteroscedasticity strength parameter (default: 1.2).
`delta_het_values`	Numeric vector. Delta values for sensitivity analysis (default: c(0.4, 0.8, 1.2)).
`n_reps_by_n`	Integer. Replications per sample size (default: 100).
`n_reps_by_delta`	Integer. Replications per delta value (default: 100).
`bootstrap_reps`	Integer. Number of bootstrap replications (default: 100).
`bootstrap_subset_size`	Integer. Size of bootstrap subset (default: 10).
`bootstrap_demo_size`	Integer. Size of bootstrap demo (default: 5).
`beta1_0`	Numeric. Intercept for first equation (default: 0.5).
`beta1_1`	Numeric. Slope for first equation (default: 1.5).
`gamma1`	Numeric. True value of the endogenous parameter (default: -0.8).
`beta2_0`	Numeric. Intercept for second equation (default: 1.0).
`beta2_1`	Numeric. Slope for second equation (default: -1.0).

`alpha1`	Numeric. Factor loading for first error (default: -0.5).
`alpha2`	Numeric. Factor loading for second error (default: 1.0).
`tau_set_id`	Numeric. Tau parameter for set identification (default: 0.2).
`endog_var_name`	Character. Name of endogenous variable (default: "Y2").
`exog_var_names`	Character. Name of exogenous variable (default: "Xk").
`df_adjust`	Character. Degrees of freedom adjustment method (default: "asymptotic"). Options: "asymptotic", "finite".

Value

A list containing all configuration parameters.

Examples

```
## Not run:
config <- create_default_config()
config$gamma1 # -0.8

# Custom configuration
custom_config <- create_default_config(
  num_simulations = 500,
  gamma1 = -0.5
)

## End(Not run)
```

`create_prono_config` *Create default configuration for Prono simulations*

Description

Creates configuration matching Prono (2014) with returns in percent.

Usage

```
create_prono_config(n = 500, k = 1, ...)
```

Arguments

`n`	Sample size
`k`	Number of exogenous variables
`...`	Additional parameters to override defaults

Value

Configuration list with parameters scaled for percent returns

`ensure_stata_packages` *Helper to ensure Stata packages are installed*

Description

Helper to ensure Stata packages are installed

Usage

```
ensure_stata_packages()
```

`extract_se_ivreg` *Extract adjusted standard errors from ivreg model*

Description

Extract adjusted standard errors from ivreg model

Usage

```
extract_se_ivreg(model, df_adjust = "asymptotic")
```

Arguments

`model`	An ivreg model object
`df_adjust`	Character string: "asymptotic" (default) or "finite"

Value

Named vector of adjusted standard errors

`extract_se_lm` *Extract adjusted standard errors from lm model*

Description

Extract adjusted standard errors from lm model

Usage

```
extract_se_lm(model, df_adjust = "asymptotic")
```

Arguments

`model`	An lm model object
`df_adjust`	Character string: "asymptotic" (default) or "finite"

Value

Named vector of adjusted standard errors

`fit_diagonal_garch_prono`
Diagonal GARCH Implementation for Prono (2014) Method

Description

This file implements the bivariate diagonal GARCH model used in Prono (2014) for heteroskedasticity-based identification. Uses the modern tsmarch package which replaces the deprecated rmgarch.

Usage

```
fit_diagonal_garch_prono(
  data,
  garch_order = c(1, 1),
  ar_ma_order = c(1, 1),
  verbose = TRUE
)
```

Arguments

`data`	Data frame with Y1 (portfolio) and Y2 (market) returns
`garch_order`	GARCH(p,q) order, default c(1,1)
`ar_ma_order`	ARMA order for conditional covariances, default c(1,1)
`verbose`	Logical. Print fitting progress

Value

List containing:

`fit`	The fitted multivariate GARCH model
`sigma2_sq`	Conditional variance of Y2 (market)
`sigma12`	Conditional covariance between Y1 and Y2
`residuals`	Matrix of standardized residuals
`spec`	Model specification object

References

Prono, T. (2014). "The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor." Journal of Applied Econometrics. Fit Bivariate Diagonal GARCH Model (Prono Specification)

Fits a bivariate diagonal GARCH model to portfolio and market returns following Prono's exact specification.

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`prono_diagonal_garch` for complete estimation `fit_dcc_garch_fallback` for fallback implementation

generate_all_plots *Generate All Simulation Plots*

Description

Generates all visualization plots for the Lewbel simulation results.

Usage

```
generate_all_plots(
  results_main,
  results_by_n,
  results_by_delta,
  bootstrap_examples,
  config,
  verbose = TRUE
)
```

Arguments

`results_main`	Data.frame. Main simulation results.
`results_by_n`	Data.frame. Sample size analysis results.
`results_by_delta`	Data.frame. Sensitivity analysis results.
`bootstrap_examples`	Data.frame. Bootstrap examples.
`config`	List. Configuration object containing true parameter values.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A list of ggplot2 objects.

Examples

```
## Not run:
# Run full simulation
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_main <- run_main_simulation(config, seeds)
results_by_n <- run_sample_size_analysis(config, seeds)
results_by_delta <- run_sensitivity_analysis(config, seeds)
bootstrap_demo <- run_bootstrap_demonstration(config, seeds)
bootstrap_examples <- analyze_bootstrap_results(
  results_main, bootstrap_demo, config
)

plots <- generate_all_plots(
  results_main, results_by_n, results_by_delta,
  bootstrap_examples, config
)
```

```
## End(Not run)
```

`generate_all_seeds` *Generate All Seeds for Lewbel Simulation*

Description

Pre-generates all seeds needed for different parts of the Lewbel simulation to ensure reproducibility across parallel execution.

Usage

```
generate_all_seeds(config)
```

Arguments

`config`	List. Configuration object created by `create_default_config` or related configuration functions.

Value

A list containing seed vectors/matrices for different simulation parts.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
names(seeds) # "main", "by_n", "by_delta", "bootstrap_demo"

## End(Not run)
```

`generate_hetid_test_data`
Generate consistent test data for all comparisons

Description

Generate consistent test data for all comparisons

Usage

```
generate_hetid_test_data(n = .hetid_const("N_DEFAULT"), seed = 42)
```

Arguments

`n`	Integer. Number of observations to generate (default: 1000).
`seed`	Integer. Random seed for reproducibility (default: 42).

Value

A data.frame with test data for Lewbel identification.

---

generate_lewbel_data *Generate Data for Lewbel (2012) Triangular Model*

Description

Creates a dataset based on the triangular model with single-factor error structure that satisfies Lewbel's identifying assumptions. The data generating process uses a common factor structure for the errors to ensure the covariance restriction Cov(Z, $\epsilon_1\epsilon_2$) = 0 is satisfied.

Usage

```
generate_lewbel_data(n_obs, params, n_x = 1)
```

Arguments

`n_obs`	Integer. Sample size.
`params`	List. Parameters for the data generating process containing: • beta1_0, beta1_1: Parameters for first equation (beta1_1 can be a vector for multiple X) • beta2_0, beta2_1: Parameters for second equation (beta2_1 can be a vector for multiple X) • gamma1: Endogenous parameter (key parameter of interest) • alpha1, alpha2: Factor loadings for common factor U • delta_het: Heteroscedasticity strength parameter
`n_x`	Integer. Number of exogenous X variables to generate (default: 1). If n_x > 1, beta1_1 and beta2_1 should be vectors of length n_x.

Details

The triangular model consists of two equations:

$$Y_1 = X'\beta_1 + \gamma_1 Y_2 + \epsilon_1$$

$$Y_2 = X'\beta_2 + \epsilon_2$$

where Y_1 is the outcome variable, Y_2 is the endogenous regressor, X is a vector of exogenous variables, and (ϵ_1, ϵ_2) are the structural errors.

The error structure follows a single-factor model:

$$\epsilon_1 = \alpha_1 U + V_1$$

$$\epsilon_2 = \alpha_2 U + V_2$$

where U, V_1, and V_2 are mutually independent, and heteroskedasticity is introduced through the variance of V_2.

The data generating process uses a unified approach for both single and multiple X:

- For each j: Z_raw_j ~ Uniform(0, 1) independently
- X_j = Z_raw_j
- Z_j = Z_raw_j - mean(Z_raw_j) (centered for use as instrument)
- For heteroskedasticity:
  - If n_x = 1: Z_het = Z_raw
  - If n_x > 1: Z_het = mean(Z_raw_1, ..., Z_raw_k)
- V_2|Z_het ~ N(0, 0.5 + 2*Z_het) (variance equals 0.5 + 2*Z_het)

Value

A data.frame with columns Y1, Y2, epsilon1, epsilon2, and:

- If n_x = 1: Xk, Z
- If n_x > 1: X1, X2, ..., Z1, Z2, ... (one Z per X)

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`verify_lewbel_assumptions` for testing the assumptions, `run_single_lewbel_simulation` for using this data in simulations

Examples

```
## Not run:
# Single X variable (backward compatible)
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0, delta_het = 1.2
)
data <- generate_lewbel_data(1000, params)

# Multiple X variables
params_multi <- list(
  beta1_0 = 0.5, beta1_1 = c(1.5, 3.0), gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = c(-1.0, 0.7),
  alpha1 = -0.5, alpha2 = 1.0, delta_het = 1.2
)
data_multi <- generate_lewbel_data(1000, params_multi, n_x = 2)

## End(Not run)
```

`generate_prono_data` *Prono (2014) Heteroskedasticity-Based Identification with GARCH*

Description

This file implements the Prono (2014) procedure for using conditional heteroskedasticity (GARCH) to generate instruments for identification in triangular systems with endogenous regressors.

Usage

```
generate_prono_data(
  n = 500,
  beta1 = c(0.05, 0.01),
  beta2 = c(0.097, -0.005),
  gamma1 = 1,
  k = 1,
  garch_params = list(omega = 0.2, alpha = 0.1, beta = 0.85),
  sigma1 = 1.5,
  rho = 0.3,
  seed = NULL
)
```

Arguments

`n`	Sample size
`beta1`	Coefficient vector for X in first equation (portfolio return equation). Default c(0.05, 0.01) gives realistic portfolio returns in percent.
`beta2`	Coefficient vector for X in second equation (market return equation). Default c(0.097, -0.005) gives mean market excess return of 0.097% matching Prono.
`gamma1`	Coefficient on Y2 in first equation (the "beta" in asset pricing)
`k`	Number of exogenous variables (excluding constant)
`garch_params`	List with GARCH parameters: omega, alpha, beta. Default values give realistic volatility clustering for weekly returns.
`sigma1`	Standard deviation of epsilon1 in percent (portfolio idiosyncratic risk)
`rho`	Correlation between epsilon1 and epsilon2 (endogeneity)
`seed`	Random seed

Value

Data frame with generated variables (Y1 and Y2 are in percent)

References

Prono, T. (2014). "The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor." Journal of Applied Econometrics, 29(5), 800-824. Generate time series data for Prono's triangular model

Generates data matching Prono (2014) asset pricing application with returns in percent (like the original paper).

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`run_single_prono_simulation` for running a single simulation `create_prono_config` for default configuration `run_prono_monte_carlo` for Monte Carlo analysis

---

`generate_rigobon_data` *Generate Data for Rigobon (2003) Regime-Based Model*

Description

Creates a dataset based on the triangular model with regime-specific heteroskedasticity following Rigobon (2003). This is a special case of Lewbel's method where heteroskedasticity drivers are discrete regime indicators.

Usage

```
generate_rigobon_data(n_obs, params, n_x = 1)
```

Arguments

`n_obs`	Integer. Sample size.
`params`	List. Parameters for the data generating process containing: • beta1_0, beta1_1: Parameters for first equation • beta2_0, beta2_1: Parameters for second equation • gamma1: Endogenous parameter (key parameter of interest) • alpha1, alpha2: Factor loadings for common factor U • regime_probs: Vector of regime probabilities (must sum to 1) • sigma2_regimes: Vector of variance multipliers for each regime (length must match regime_probs)
`n_x`	Integer. Number of exogenous X variables to generate (default: 1).

Details

The triangular model is:

$$Y_1 = \beta_{1,0} + \beta_{1,1}X + \gamma_1 Y_2 + \epsilon_1$$
$$Y_2 = \beta_{2,0} + \beta_{2,1}X + \epsilon_2$$

The error structure follows Rigobon's regime heteroskedasticity:

$$\epsilon_1 = \alpha_1 U + V_1$$

$$\epsilon_2 = \alpha_2 U + V_2$$

where V_2 has variance that depends on the regime:

$$Var(V_2|regime = s) = \sigma^2_{2,s}$$

Value

A data.frame with columns Y1, Y2, epsilon1, epsilon2, regime, and:

- If n_x = 1: Xk, plus Z1, Z2, ... (one centered dummy per regime)
- If n_x > 1: X1, X2, ..., plus Z1, Z2, ... (one centered dummy per regime)

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792. doi:10.1162/003465303772815727

See Also

`generate_lewbel_data` for continuous heteroskedasticity drivers

Examples

```
## Not run:
# Two-regime example (e.g., pre/post policy change)
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0,
  regime_probs = c(0.4, 0.6), # 40% in regime 1, 60% in regime 2
  sigma2_regimes = c(1.0, 2.5) # Variance is 2.5x higher in regime 2
)
data <- generate_rigobon_data(1000, params)

# Three-regime example
params_3reg <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0,
  regime_probs = c(0.3, 0.4, 0.3),
  sigma2_regimes = c(0.5, 1.0, 2.0)
)
data_3reg <- generate_rigobon_data(1000, params_3reg)

## End(Not run)
```

`generate_seed_matrix` *Generate Seed Matrix for Reproducible Parallel Simulations*

Description

Pre-generates seeds for reproducible parallel simulations to ensure consistent results across different computing environments.

Usage

```
generate_seed_matrix(base_seed, n_experiments, n_reps_each)
```

Arguments

`base_seed`	Integer. Base seed for random number generation.
`n_experiments`	Integer. Number of experiments (rows in matrix).
`n_reps_each`	Integer. Number of replications per experiment (columns).

Value

A matrix of seeds with dimensions n_experiments x n_reps_each.

Examples

```
## Not run:
seeds <- generate_seed_matrix(123, 3, 100)
dim(seeds) # 3 x 100

## End(Not run)
```

`get_critical_value` *Get critical value for confidence intervals*

Description

Get critical value for confidence intervals

Usage

```
get_critical_value(
  n,
  k,
  alpha = .hetid_const("ALPHA_DEFAULT"),
  df_adjust = "asymptotic"
)
```

Arguments

`n`	Sample size
`k`	Number of parameters
`alpha`	Significance level (default 0.05)
`df_adjust`	Character string: "asymptotic" (default) or "finite"

Value

Critical value

`get_stata_path` *Get the actual Stata executable path*

Description

Get the actual Stata executable path

Usage

```
get_stata_path()
```

`has_curl` *Check if curl is available*

Description

Check if curl is available

Usage

```
has_curl()
```

`has_haven` *Check if haven is available*

Description

Check if haven is available

Usage

```
has_haven()
```

`has_rendo` *Check if REndo is available*

Description

Check if REndo is available

Usage

```
has_rendo()
```

`has_rstata` *Check if RStata is available*

Description

Check if RStata is available

Usage

```
has_rstata()
```

`has_stata` *Check if Stata is available via RStata*

Description

Check if Stata is available via RStata

Usage

```
has_stata()
```

`hetid_opt` *Get hetid package options with fallback to constants*

Description

This function retrieves user-configurable options for the hetid package. If a user hasn't set a specific option, it falls back to the default value stored in the package's internal constants environment.

Usage

```
hetid_opt(option_name)
```

Arguments

`option_name`	Character. Name of the option to retrieve. Supported options include: • `display_digits`: Number of digits for displaying results • `alpha_level`: Significance level for statistical tests • `weak_f_threshold`: F-statistic threshold for weak instruments • `parallel_offset`: Offset for parallel processing workers • `bootstrap_reps`: Number of bootstrap replications

Value

The option value. Returns the user-set value if available, otherwise returns the default from the constants environment.

Examples

```
# Get default display digits
hetid_opt("display_digits")

# Set a custom value
options(hetid.display_digits = 6)
hetid_opt("display_digits")

# Reset to default
options(hetid.display_digits = NULL)
hetid_opt("display_digits")
```

`lewbel_gmm` *Estimate Lewbel Model using GMM*

Description

Main function to estimate Lewbel's heteroskedasticity-based identification model using Generalized Method of Moments (GMM).

Usage

```
lewbel_gmm(
  data,
  system = c("triangular", "simultaneous"),
  y1_var = "Y1",
  y2_var = "Y2",
  x_vars = "Xk",
  z_vars = NULL,
  add_intercept = TRUE,
  gmm_type = c("twoStep", "iterative", "cue"),
  initial_values = NULL,
  vcov = c("HAC", "iid", "cluster"),
  cluster_var = NULL,
  compute_se = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Data frame containing all required variables. Must include the dependent variables and any exogenous regressors specified in the model.
`system`	Character. Type of system: "triangular" or "simultaneous" (default: "triangular"). Note: Simultaneous systems require strong identification conditions - either many regimes (4+) or large variance differences across regimes for numerical stability.
`y1_var`	Character. Name of the first dependent variable (default: "Y1").
`y2_var`	Character. Name of the second dependent variable/endogenous regressor (default: "Y2").

`x_vars`	Character vector. Names of exogenous variables (default: "Xk").
`z_vars`	Character vector. Names of heteroskedasticity drivers (default: NULL).
`add_intercept`	Logical. Whether to add an intercept to the exogenous variables (default: TRUE).
`gmm_type`	Character. GMM type: "twoStep", "iterative", or "cue" (default: "twoStep").
`initial_values`	Numeric vector. Initial parameter values (default: NULL, uses OLS).
`vcov`	Character. Type of variance-covariance matrix: "HAC", "iid", or "cluster" (default: "HAC").
`cluster_var`	Character. Variable name for clustering if vcov = "cluster" (default: NULL).
`compute_se`	Logical. Whether to compute standard errors (default: TRUE). Passed to gmm call.
`verbose`	Logical. Whether to print progress messages (default: TRUE).
`...`	Additional arguments passed to gmm().

Details

This function implements Lewbel's (2012) heteroskedasticity-based identification using the GMM framework. The method exploits heteroskedasticity in the error terms to generate valid instruments for endogenous regressors.

For simultaneous equation systems, identification becomes more challenging. The system requires sufficient variation in heteroskedasticity patterns to distinguish between the bidirectional effects. In practice, this means you need either many distinct heteroskedasticity regimes or very large differences in variance across existing regimes.

Value

An object of class "gmm" containing estimation results.

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`lewbel_triangular_moments`, `lewbel_simultaneous_moments` for moment condition functions. `rigobon_gmm` for regime-based heteroskedasticity identification. `prono_gmm` for GARCH-based heteroskedasticity identification. `compare_gmm_2sls` for comparing GMM with 2SLS estimates. `run_single_lewbel_simulation` for 2SLS implementation.

Examples

```
## Not run:
# Generate example data
set.seed(123)
n <- 1000
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0, delta_het = 1.2
)
```

```
data <- generate_lewbel_data(n, params)

# Estimate triangular system
gmm_tri <- lewbel_gmm(data, system = "triangular")
summary(gmm_tri)

# Estimate simultaneous system
gmm_sim <- lewbel_gmm(data, system = "simultaneous")
summary(gmm_sim)

# Compare with 2SLS
tsls_result <- run_single_lewbel_simulation(1, c(params, sample_size = n))
cat("2SLS estimate:", tsls_result$tsls_gamma1, "\n")
cat("GMM estimate:", coef(gmm_tri)["gamma1"], "\n")

## End(Not run)
```

lewbel_sim *Simulated Lewbel Test Data*

Description

A simulated dataset for testing Lewbel (2012) identification methods. This dataset contains a triangular system with endogeneity and heteroskedasticity suitable for testing the hetid package functions.

Usage

```
lewbel_sim
```

Format

A data frame with 1000 rows and 5 variables:

id Observation identifier

y Dependent variable (Y1 in the triangular system)

P Endogenous regressor (Y2 in the triangular system)

X1 First exogenous regressor

X2 Second exogenous regressor

Details

The data was generated using the following triangular system:

$$y = 0.5 + 1.5 \cdot X1 + 3.0 \cdot X2 - 0.8 \cdot P + \epsilon_1$$

$$P = 1.0 - 1.0 \cdot X1 + 0.7 \cdot X2 + \epsilon_2$$

The error structure follows a single-factor model with heteroskedasticity:

$$\epsilon_1 = -0.5 \cdot U + V_1$$

$$\epsilon_2 = 1.0 \cdot U + V_2$$

where $V_2 \sim N(0, \exp(1.2 \cdot Z))$ with $Z = X2^2 - E[X2^2]$.

Source

Simulated data using generate_lewbel_data() function

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

`lewbel_simultaneous_moments` *Define GMM Moment Conditions for Lewbel Simultaneous System*

Description

Creates the moment function for GMM estimation of a simultaneous equations system using Lewbel's heteroskedasticity-based identification.

Usage

```
lewbel_simultaneous_moments(
  theta,
  data,
  y1_var,
  y2_var,
  x_vars,
  z_vars,
  add_intercept,
  z_sq
)
```

Arguments

`theta`	Numeric vector. Parameters: c(beta1, gamma1, beta2, gamma2).
`data`	Data frame containing all required variables. Must include the dependent variables and any exogenous regressors specified in the model.
`y1_var`	Character. Name of the first dependent variable (default: "Y1").
`y2_var`	Character. Name of the second dependent variable/endogenous regressor (default: "Y2").
`x_vars`	Character vector. Names of exogenous variables (default: "Xk").
`z_vars`	Character vector. Names of heteroskedasticity drivers (default: NULL).
`add_intercept`	Logical. Whether to add an intercept to the exogenous variables.
`z_sq`	Logical. Whether to include squared terms in the Z matrix for simultaneous equations.

Details

For the simultaneous system:

$$Y_1 = X'\beta_1 + \gamma_1 Y_2 + \epsilon_1$$

$$Y_2 = X'\beta_2 + \gamma_2 Y_1 + \epsilon_2$$

The moment conditions are the same as the triangular system. Note: Requires gamma1 * gamma2 != 1 for identification.

Value

Matrix of moment conditions (n x q).

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`lewbel_gmm` for the main GMM estimation function. `lewbel_triangular_moments` for triangular system moments.

`lewbel_triangular_moments`

Define GMM Moment Conditions for Lewbel Triangular System

Description

Creates the moment function for GMM estimation of a triangular system using Lewbel's heteroskedasticity-based identification.

Usage

```
lewbel_triangular_moments(
  theta,
  data,
  y1_var,
  y2_var,
  x_vars,
  z_vars,
  add_intercept
)
```

Arguments

`theta`	Numeric vector. Parameters to estimate: c(beta1, gamma1, beta2).
`data`	Data frame containing all required variables. Must include the dependent variables and any exogenous regressors specified in the model.
`y1_var`	Character. Name of the first dependent variable (default: "Y1").
`y2_var`	Character. Name of the second dependent variable/endogenous regressor (default: "Y2").
`x_vars`	Character vector. Names of exogenous variables (default: "Xk").
`z_vars`	Character vector. Names of heteroskedasticity drivers (default: NULL, uses centered X).
`add_intercept`	Logical. Whether to add an intercept to the exogenous variables.

Details

For the triangular system:

$$Y_1 = X'\beta_1 + \gamma_1 Y_2 + \epsilon_1$$

$$Y_2 = X'\beta_2 + \epsilon_2$$

The moment conditions are:

- $E[X \times \epsilon_1] = 0$
- $E[X \times \epsilon_2] = 0$
- $E[Z \times \epsilon_1 \times \epsilon_2] = 0$

where Z = g(X) is a mean-zero transformation of X.

Value

Matrix of moment conditions (n x q).

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`lewbel_gmm` for the main GMM estimation function. `lewbel_simultaneous_moments` for simultaneous system moments.

plot_bootstrap_ci *Create Bootstrap Confidence Intervals Plot*

Description

Creates a plot showing set identification bounds with bootstrap confidence intervals.

Usage

```
plot_bootstrap_ci(bootstrap_examples, config)
```

Arguments

bootstrap_examples	Data.frame. Bootstrap examples with standard errors.
config	List. Configuration object containing true parameter values.

Value

A ggplot2 object or NULL if insufficient data.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_main <- run_main_simulation(config, seeds)
bootstrap_demo <- run_bootstrap_demonstration(config, seeds)
bootstrap_examples <- analyze_bootstrap_results(
  results_main, bootstrap_demo, config
)
p5 <- plot_bootstrap_ci(bootstrap_examples, config)
if (!is.null(p5)) print(p5)

## End(Not run)
```

plot_estimator_distributions *Create Distribution Plot of Estimators*

Description

Creates a density plot comparing the distributions of OLS and 2SLS estimators.

Usage

```
plot_estimator_distributions(results_clean, config)
```

Arguments

`results_clean`	Data.frame. Cleaned simulation results.
`config`	List. Configuration object containing true parameter values.

Value

A ggplot2 object.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results <- run_main_simulation(config, seeds)
results_clean <- na.omit(results)
p1 <- plot_estimator_distributions(results_clean, config)
print(p1)

## End(Not run)
```

`plot_first_stage_f_dist`

Create First-Stage F Distribution Plot

Description

Creates a histogram of first-stage F-statistics with weak instrument threshold.

Usage

```
plot_first_stage_f_dist(results_clean, config = NULL, weak_iv_pct = NULL)
```

Arguments

`results_clean`	Data.frame. Cleaned simulation results.
`config`	List. Optional. Configuration object (not currently used).
`weak_iv_pct`	Numeric. Percentage of simulations with weak instruments.

Value

A ggplot2 object.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results <- run_main_simulation(config, seeds)
results_clean <- na.omit(results)
weak_iv_pct <- mean(results_clean$first_stage_F <
  .hetid_const("WEAK_INSTRUMENT_F_THRESHOLD")) * 100
```

```
p4 <- plot_first_stage_f_dist(results_clean, weak_iv_pct)
print(p4)

## End(Not run)
```

---

plot_het_sensitivity *Create Heteroscedasticity Sensitivity Plot*

Description

Creates a boxplot showing 2SLS performance by heteroscedasticity strength.

Usage

```
plot_het_sensitivity(results_by_delta, config)
```

Arguments

results_by_delta	Data.frame. Results from sensitivity analysis.
config	List. Configuration object containing true parameter values.

Value

A ggplot2 object.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_by_delta <- run_sensitivity_analysis(config, seeds)
p3 <- plot_het_sensitivity(results_by_delta, config)
print(p3)

## End(Not run)
```

---

plot_sample_size_consistency *Create Sample Size Consistency Plot*

Description

Creates a boxplot showing 2SLS estimate consistency across sample sizes.

Usage

```
plot_sample_size_consistency(results_by_n, config)
```

Arguments

`results_by_n`	Data.frame. Results from sample size analysis.
`config`	List. Configuration object containing true parameter values.

Value

A ggplot2 object.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
results_by_n <- run_sample_size_analysis(config, seeds)
p2 <- plot_sample_size_consistency(results_by_n, config)
print(p2)

## End(Not run)
```

`print.lewbel_gmm` *Print Method for Lewbel GMM*

Description

Print Method for Lewbel GMM

Usage

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

Arguments

`x`	An object of class "lewbel_gmm".
`...`	Additional arguments passed to print.gmm.

`print_simulation_summary`
Print Simulation Summary

Description

Prints a comprehensive summary of simulation findings.

Usage

```
print_simulation_summary(analysis = NULL, config = NULL, verbose = TRUE)
```

Arguments

`analysis`	List. Optional. Analysis results object containing simulation metrics.
`config`	List. Optional. Configuration object (not currently used).
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Examples

```
## Not run:
print_simulation_summary()

## End(Not run)
```

`prono_diagonal_garch` *Run Prono Estimation with Diagonal GARCH*

Description

Complete Prono estimation using proper diagonal GARCH specification

Usage

```
prono_diagonal_garch(
  data,
  method = c("2sls", "gmm"),
  garch_order = c(1, 1),
  verbose = TRUE
)
```

Arguments

`data`	Data frame with Y1, Y2, and X variables
`method`	Character. Either "2sls" or "gmm"
`garch_order`	GARCH(p,q) order
`verbose`	Print progress

Value

List with estimation results

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`fit_diagonal_garch_prono` for GARCH fitting `run_single_prono_simulation` for 2SLS estimation `prono_gmm` for GMM estimation

Examples

```
## Not run:
# Time-consuming example with diagonal GARCH
data <- generate_prono_data(n = 500)
result <- prono_diagonal_garch(data, method = "2sls")

## End(Not run)
```

---

prono_gmm *GMM Estimation for Prono (2014) GARCH-Based Identification*

Description

Implementation of GMM estimation for Prono's (2014) heteroskedasticity-based identification strategy using GARCH models.

Usage

```
prono_gmm(
  data,
  system = "triangular",
  y1_var = "Y1",
  y2_var = "Y2",
  x_vars = NULL,
  garch_order = c(1, 1),
  fit_garch = TRUE,
  add_intercept = TRUE,
  gmm_type = "twoStep",
  vcov = "HAC",
  initial_values = NULL,
  compute_se = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Data frame containing all variables.
`system`	Character. Either "triangular" (default).
`y1_var`	Character. Name of the first dependent variable (default: "Y1").
`y2_var`	Character. Name of the second dependent variable (default: "Y2").
`x_vars`	Character vector. Names of exogenous variables.
`garch_order`	GARCH(p,q) order (default: c(1,1)).
`fit_garch`	Logical. Whether to fit GARCH model (default: TRUE).
`add_intercept`	Logical. Whether to add an intercept (default: TRUE).
`gmm_type`	Character. GMM type: "onestep", "twoStep" (default), "iterative", or "cue".
`vcov`	Character. Variance-covariance matrix type: "iid", "HAC" (default), or "cluster".

`initial_values`	Numeric vector. Initial parameter values (optional).
`compute_se`	Logical. Whether to compute standard errors (default: TRUE).
`verbose`	Logical. Whether to print progress messages (default: TRUE).
`...`	Additional arguments passed to gmm::gmm.

Value

An object of class "prono_gmm" containing GMM estimation results.

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`prono_triangular_moments` for moment conditions. `run_single_prono_simulation` for 2SLS estimation.

---

prono_triangular_moments

Define GMM Moment Conditions for Prono Triangular System

Description

Creates the moment function for GMM estimation of a triangular system using Prono's GARCH-based identification.

Usage

```
prono_triangular_moments(
  theta,
  data,
  y1_var,
  y2_var,
  x_vars,
  garch_order = c(1, 1),
  add_intercept = TRUE
)
```

Arguments

`theta`	Numeric vector. Parameters to estimate: c(beta1, gamma1, beta2).
`data`	Data frame containing the variables.
`y1_var`	Character. Name of the first dependent variable (default: "Y1").
`y2_var`	Character. Name of the second dependent variable/endogenous regressor (default: "Y2").
`x_vars`	Character vector. Names of exogenous variables.
`garch_order`	GARCH(p,q) order for conditional variance estimation.
`add_intercept`	Logical. Whether to add an intercept to the exogenous variables.

Value

Matrix of moment conditions (n x q).

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`prono_gmm` for the main GMM estimation function. `run_single_prono_simulation` for 2SLS estimation.

`replicate_prono_table2` *Replicate Prono's Table II Results*

Description

Run Monte Carlo simulation matching Prono's exact specification

Usage

```
replicate_prono_table2(
  n_sim = 1000,
  n_obs = 500,
  config = create_prono_config(n = n_obs),
  use_diagonal_garch = TRUE,
  verbose = TRUE
)
```

Arguments

`n_sim`	Number of simulations
`n_obs`	Sample size per simulation
`config`	Configuration from create_prono_config()
`use_diagonal_garch`	Use diagonal GARCH (TRUE) or univariate (FALSE)
`verbose`	Print progress

Value

List with results data frame and summary statistics matching Prono's Table II

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`run_prono_monte_carlo` for standard Monte Carlo `create_prono_config` for configuration

Examples

```
## Not run:
# Replication of Prono Table II (time-consuming)
# Uses 1000 simulations to match paper results
results <- replicate_prono_table2(n_sim = 1000, n_obs = 500)

## End(Not run)
```

`rigobon_gmm` *GMM Estimation for Rigobon (2003) Regime-Based Identification*

Description

Implementation of GMM estimation for Rigobon's (2003) heteroskedasticity-based identification strategy using regime changes.

Usage

```
rigobon_gmm(
  data,
  system = "triangular",
  y1_var = "Y1",
  y2_var = "Y2",
  x_vars = "Xk",
  regime_var = "regime",
  add_intercept = TRUE,
  gmm_type = "twoStep",
  vcov = "HAC",
  initial_values = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Data frame containing all variables.
`system`	Character. Either "triangular" (default) or "simultaneous". Note: Simultaneous systems require many regimes (4+) and large variance differences across regimes for numerical stability and identification.
`y1_var`	Character. Name of the first dependent variable.
`y2_var`	Character. Name of the second dependent variable.
`x_vars`	Character vector. Names of exogenous variables.
`regime_var`	Character. Name of the regime indicator variable.
`add_intercept`	Logical. Whether to add an intercept.

`gmm_type`	Character. GMM type.
`vcov`	Character. Variance-covariance matrix type.
`initial_values`	Numeric vector. Initial parameter values.
`verbose`	Logical. Whether to print progress messages (default: TRUE).
`...`	Additional arguments passed to gmm::gmm.

Value

An object of class "rigobon_gmm" containing GMM estimation results.

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792.

`rigobon_moment_conditions`
Define GMM Moment Conditions for Rigobon Triangular System

Description

Creates the moment function for GMM estimation of a triangular system using Rigobon's regime-based identification.

Usage

```
rigobon_moment_conditions(
  theta,
  data,
  system,
  y1_var,
  y2_var,
  x_vars,
  regime_var,
  add_intercept = TRUE
)
```

Arguments

`theta`	Numeric vector. Parameters to estimate.
`data`	Data frame containing the variables.
`system`	Character. The type of system, either "triangular" or "simultaneous".
`y1_var`	Character. Name of the first dependent variable.
`y2_var`	Character. Name of the second dependent variable.
`x_vars`	Character vector. Names of exogenous variables.
`regime_var`	Character. Name of the regime indicator variable.
`add_intercept`	Logical. Whether to add an intercept.

Value

Matrix of moment conditions.

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792.

`run_bootstrap_demonstration` *Run Bootstrap Demonstration*

Description

Runs a separate demonstration of bootstrap standard errors for set identification bounds.

Usage

```
run_bootstrap_demonstration(config, seeds, verbose = TRUE)
```

Arguments

`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`seeds`	List. Seeds for reproducibility, typically generated by `generate_all_seeds`.
`verbose`	Logical. Whether to show progress information during execution (default: TRUE).

Value

A data.frame containing bootstrap demonstration results.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
bootstrap_results <- run_bootstrap_demonstration(config, seeds)

## End(Not run)
```

`run_lewbel_demo` *Run Quick Lewbel Monte Carlo Demo*

Description

Runs a quick demonstration of the Lewbel Monte Carlo simulation with reduced parameters for faster execution.

Usage

```
run_lewbel_demo(num_simulations = 100, verbose = TRUE)
```

Arguments

`num_simulations`	Integer. Number of simulations to run (default: 100).
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

Results from run_lewbel_monte_carlo() with reduced parameters.

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`run_lewbel_monte_carlo`

Examples

```
## Not run:
# Quick demo with 50 simulations
demo_results <- run_lewbel_demo(50)

# Silent demo
demo_results <- run_lewbel_demo(100, verbose = FALSE)

## End(Not run)
```

`run_lewbel_monte_carlo` *Run Complete Lewbel (2012) Monte Carlo Simulation*

Description

Executes a comprehensive Monte Carlo simulation to evaluate the performance of Lewbel's (2012) heteroscedasticity-based identification strategy. This is the main function that orchestrates all simulation components.

Usage

```
run_lewbel_monte_carlo(
  config = NULL,
  run_verification = TRUE,
  run_bootstrap_demo = TRUE,
  run_sample_analysis = TRUE,
  run_sensitivity = TRUE,
  generate_plots = TRUE,
  verbose = TRUE
)
```

Arguments

`config`	If NULL, uses default configuration.
`run_verification`	Logical. Whether to run assumption verification (default: TRUE).
`run_bootstrap_demo`	Logical. Whether to run bootstrap demonstration (default: TRUE).
`run_sample_analysis`	Logical. Whether to run sample size analysis (default: TRUE).
`run_sensitivity`	Logical. Whether to run sensitivity analysis (default: TRUE).
`generate_plots`	Logical. Whether to generate visualization plots (default: TRUE).
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Details

This function runs a complete Monte Carlo evaluation including:

- Verification of Lewbel's identifying assumptions
- Main simulation comparing OLS vs 2SLS (Lewbel) estimators
- Bootstrap demonstration for set identification bounds
- Sample size consistency analysis
- Sensitivity analysis to heteroscedasticity strength
- Comprehensive result analysis and visualization

The simulation uses parallel processing for efficiency and includes proper seed management for reproducibility.

Value

A list containing:

- config: Configuration used
- results_main: Main simulation results
- results_by_n: Sample size analysis results (if run)
- results_by_delta: Sensitivity analysis results (if run)
- bootstrap_demo: Bootstrap demonstration results (if run)
- analysis: Summary analysis
- plots: Visualization plots (if generated)

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`run_single_lewbel_simulation`, `calculate_lewbel_bounds`

Examples

```
## Not run:
# Run with default settings
results <- run_lewbel_monte_carlo()

# Run with custom configuration
custom_config <- create_default_config(
  num_simulations = 500,
  gamma1 = -0.5,
  delta_het = 1.5
)
results <- run_lewbel_monte_carlo(custom_config)

# Run only main simulation without extras
results <- run_lewbel_monte_carlo(
  run_bootstrap_demo = FALSE,
  run_sample_analysis = FALSE,
  run_sensitivity = FALSE
)

## End(Not run)
```

`run_main_simulation` *Run Main Lewbel Monte Carlo Simulation*

Description

Executes the main Monte Carlo simulation to evaluate the performance of Lewbel's (2012) heteroscedasticity-based identification strategy.

Usage

```
run_main_simulation(config, seeds, verbose = TRUE)
```

Arguments

`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`seeds`	List. Seeds for reproducibility, typically generated by `generate_all_seeds`.
`verbose`	Logical. Whether to show progress information during execution (default: TRUE).

Value

A data.frame containing results from all simulation runs.

Examples

```
## Not run:
config <- create_default_config(num_simulations = 100)
seeds <- generate_all_seeds(config)
results <- run_main_simulation(config, seeds)

## End(Not run)
```

`run_prono_demo` *Run Prono demonstration*

Description

Run Prono demonstration

Usage

```
run_prono_demo(n = 500, print_results = TRUE)
```

Arguments

`n`	Sample size
`print_results`	Whether to print results

Value

Results from single simulation (invisibly)

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`run_single_prono_simulation` for the underlying simulation `create_prono_config` for configuration `run_prono_monte_carlo` for full Monte Carlo analysis

Examples

```
## Not run:
# Quick demonstration with reduced sample size
run_prono_demo(n = 200, print_results = TRUE)

## End(Not run)
```

`run_prono_monte_carlo` *Run Prono Monte Carlo simulation*

Description

Run Prono Monte Carlo simulation

Usage

```
run_prono_monte_carlo(
  config,
  n_sims = 1000,
  parallel = FALSE,
  n_cores = NULL,
  progress = TRUE
)
```

Arguments

`config`	Configuration list
`n_sims`	Number of simulations
`parallel`	Whether to use parallel processing
`n_cores`	Number of cores for parallel processing
`progress`	Whether to show progress bar

Value

Data frame with simulation results

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`run_single_prono_simulation` for single simulation `create_prono_config` for configuration setup

Examples

```
## Not run:
# Time-consuming Monte Carlo simulation
# For actual research, use n_sims = 1000+
config <- create_prono_config(n = 500)
mc_results <- run_prono_monte_carlo(config, n_sims = 100)

## End(Not run)
```

`run_rigobon_analysis` *Rigobon (2003) Regime-Based Heteroskedasticity Identification*

Description

This file implements the Rigobon (2003) procedure for using discrete regime indicators to generate instruments for identification in triangular systems with endogenous regressors. The method exploits heteroskedasticity across different regimes (e.g., policy periods, market conditions).

Usage

```
run_rigobon_analysis(
  n_obs = .hetid_const("N_DEFAULT"),
  params = NULL,
  data = NULL,
  regime_var = "regime",
  endog_var = "Y2",
  exog_vars = "Xk",
  verbose = TRUE,
  return_all = FALSE
)
```

Arguments

`n_obs`	Integer. Sample size (default: 1000).
`params`	List. Parameters for data generation. If NULL, uses default parameters suitable for demonstration.
`data`	Data.frame. Optional. Pre-existing data with regime indicators. If provided, skips data generation.
`regime_var`	Character. Name of regime variable in data (default: "regime").

`endog_var`	Character. Name of endogenous variable (default: "Y2").
`exog_vars`	Character vector. Names of exogenous variables (default: "Xk").
`verbose`	Logical. Whether to print progress messages (default: TRUE).
`return_all`	Logical. Whether to return all intermediate results (default: FALSE).

Details

The Rigobon method is a special case of Lewbel's (2012) approach where the heteroskedasticity drivers are discrete regime indicators rather than continuous functions of exogenous variables. Run Complete Rigobon Analysis

This is the main function that performs a complete Rigobon (2003) heteroskedasticity-based identification analysis. It combines data generation, estimation, and diagnostic testing in a single workflow.

Value

A list containing:

- estimates: Data frame comparing OLS and Rigobon 2SLS estimates
- diagnostics: Heteroskedasticity test results and first-stage F-stats
- data: The data used (generated or provided)
- models: Fitted model objects (if return_all = TRUE)

References

Rigobon, R. (2003). "Identification Through Heteroskedasticity." The Review of Economics and Statistics, 85(4), 777-792.

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792. doi:10.1162/003465303772815727

See Also

`generate_rigobon_data`, `run_rigobon_estimation`, `validate_rigobon_assumptions`

Examples

```
## Not run:
# Quick analysis with default parameters
results <- run_rigobon_analysis()

# Custom parameters for stronger identification
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0,
  regime_probs = c(0.3, 0.7),
  sigma2_regimes = c(1.0, 3.0) # Large difference in variances
)
results <- run_rigobon_analysis(n_obs = 2000, params = params)

# Using existing data
# Assume you have data with a regime indicator
results <- run_rigobon_analysis(data = my_data, regime_var = "period")
```

```
## End(Not run)
```

`run_rigobon_demo` *Run Rigobon (2003) Identification Demo*

Description

Demonstrates Rigobon's regime-based heteroskedasticity identification method with example data and analysis.

Usage

```
run_rigobon_demo(
  n_obs = .hetid_const("N_DEFAULT"),
  n_regimes = 2,
  verbose = TRUE
)
```

Arguments

`n_obs`	Integer. Sample size (default: 1000).
`n_regimes`	Integer. Number of regimes (default: 2).
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Value

A list containing:

- data: Generated data
- params: True parameters used
- results: Estimation results
- comparison: Comparison of OLS vs Rigobon estimates

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792. doi:10.1162/003465303772815727

See Also

`generate_rigobon_data`, `run_rigobon_estimation`

Examples

```
## Not run:
# Basic two-regime demo
demo <- run_rigobon_demo()

# Three-regime example with larger sample
demo_3reg <- run_rigobon_demo(n_obs = 2000, n_regimes = 3)

# Silent demo
demo_quiet <- run_rigobon_demo(verbose = FALSE)

## End(Not run)
```

`run_rigobon_estimation` *Run Rigobon (2003) Regime-Based Estimation*

Description

Implements Rigobon's heteroskedasticity-based identification using discrete regime indicators. This is a wrapper around the Lewbel estimation framework that automatically constructs instruments from regime dummies.

Usage

```
run_rigobon_estimation(
  data,
  endog_var = "Y2",
  exog_vars = "Xk",
  regime_var = "regime",
  df_adjust = "asymptotic",
  return_diagnostics = FALSE
)
```

Arguments

`data`	Data.frame. Must contain Y1, Y2, X variables, and regime indicator.
`endog_var`	Character. Name of endogenous variable (default: "Y2").
`exog_vars`	Character vector. Names of exogenous variables (default: "Xk").
`regime_var`	Character. Name of regime indicator variable (default: "regime").
`df_adjust`	Character. Degrees of freedom adjustment: "asymptotic" or "finite" (default: "asymptotic").
`return_diagnostics`	Logical. Whether to return additional diagnostic information (default: FALSE).

Details

The function:

1. Creates centered dummy variables for each regime
2. Estimates first-stage residuals
3. Constructs instruments as (centered regime dummy) * (first-stage residual)
4. Performs 2SLS estimation using all regime-based instruments

This implements the procedure described in Rigobon (2003) for identification through heteroskedasticity across discrete regimes.

Value

If return_diagnostics = FALSE: A list with:

- ols: OLS estimates and standard errors
- tsls: 2SLS estimates and standard errors using Rigobon instruments
- first_stage_F: Vector of F-statistics for each instrument

If return_diagnostics = TRUE: Additionally returns:

- instruments: Matrix of generated instruments
- regime_props: Proportion of observations in each regime
- heteroskedasticity_test: Test for regime-based heteroskedasticity

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792. doi:10.1162/003465303772815727

See Also

`generate_rigobon_data` for generating regime-based data

Examples

```
## Not run:
# Generate Rigobon-style data
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0,
  regime_probs = c(0.4, 0.6),
  sigma2_regimes = c(1.0, 2.5)
)
data <- generate_rigobon_data(1000, params)

# Run Rigobon estimation
results <- run_rigobon_estimation(data)
print(results$tsls$estimates)

# With diagnostics
results_diag <- run_rigobon_estimation(data, return_diagnostics = TRUE)
print(results_diag$heteroskedasticity_test)
```

```
## End(Not run)
```

---

run_sample_size_analysis *Run Sample Size Analysis*

Description

Analyzes the consistency of estimators across different sample sizes.

Usage

```
run_sample_size_analysis(config, seeds, verbose = TRUE)
```

Arguments

`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`seeds`	List. Seeds for reproducibility, typically generated by `generate_all_seeds`.
`verbose`	Logical. Whether to show progress information during execution (default: TRUE).

Value

A data.frame containing results for different sample sizes.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
size_results <- run_sample_size_analysis(config, seeds)

## End(Not run)
```

---

run_sensitivity_analysis *Run Sensitivity Analysis*

Description

Analyzes sensitivity of results to heteroscedasticity strength.

Usage

```
run_sensitivity_analysis(config, seeds, verbose = TRUE)
```

Arguments

`config`	List. Configuration object created by `create_default_config` or related configuration functions.
`seeds`	List. Seeds for reproducibility, typically generated by `generate_all_seeds`.
`verbose`	Logical. Whether to show progress information during execution (default: TRUE).

Value

A data.frame containing results for different heteroscedasticity parameters.

Examples

```
## Not run:
config <- create_default_config()
seeds <- generate_all_seeds(config)
sensitivity_results <- run_sensitivity_analysis(config, seeds)

## End(Not run)
```

`run_single_lewbel_simulation`

Run Single Lewbel Simulation

Description

Executes a single Monte Carlo simulation run comparing OLS, 2SLS (Lewbel), and set identification approaches for estimating the endogenous parameter.

Usage

```
run_single_lewbel_simulation(
  sim_id,
  params,
  endog_var = "Y2",
  exog_vars = "Xk",
  compute_bounds_se = FALSE,
  return_models = FALSE,
  df_adjust = "asymptotic"
)
```

Arguments

`sim_id`	Integer. Simulation run identifier.
`params`	List. Parameters for data generation and estimation.
`endog_var`	Character. Name of endogenous variable (default: "Y2").
`exog_vars`	Character vector. Names of exogenous variables (default: "Xk").
`compute_bounds_se`	Logical. Whether to compute bootstrap SE for bounds (default: FALSE).

`return_models`	Logical. Whether to return the fitted model objects (default: FALSE).
`df_adjust`	Character. Method for degrees of freedom adjustment: • "asymptotic": No adjustment (default) • "finite": Finite sample adjustment using HC2 formula

Value

If return_models = FALSE: A data.frame with one row containing simulation results including OLS and 2SLS estimates, coverage indicators, first-stage F-statistic, and identification bounds. If return_models = TRUE: A list containing:

- results: The data.frame described above
- models: A list with ols_model, first_stage_model, tsls_model
- data: The generated data

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`calculate_lewbel_bounds` for set identification

Examples

```
## Not run:
config <- create_default_config()
params <- list(
  sample_size = config$main_sample_size,
  beta1_0 = config$beta1_0, beta1_1 = config$beta1_1, gamma1 = config$gamma1,
  beta2_0 = config$beta2_0, beta2_1 = config$beta2_1,
  alpha1 = config$alpha1, alpha2 = config$alpha2,
  delta_het = config$delta_het, tau_set_id = config$tau_set_id,
  bootstrap_reps = config$bootstrap_reps
)
result <- run_single_lewbel_simulation(1, params)

# With models
result_with_models <- run_single_lewbel_simulation(
  1, params,
  return_models = TRUE
)

## End(Not run)
```

`run_single_prono_simulation` *Run single Prono simulation with GARCH-based instruments*

Description

Run single Prono simulation with GARCH-based instruments

Usage

```
run_single_prono_simulation(config, return_details = FALSE)
```

Arguments

`config`	Configuration list with simulation parameters
`return_details`	If TRUE, return detailed results

Value

List with estimation results including gamma1_true, gamma1_ols, gamma1_iv, standard errors, biases, F-statistic, and optionally full model objects

References

Prono, T. (2014). The Role of Conditional Heteroskedasticity in Identifying and Estimating Linear Triangular Systems, with Applications to Asset Pricing Models That Include a Mismeasured Factor. Journal of Applied Econometrics, 29(5), 800-824. doi:10.1002/jae.2387

See Also

`generate_prono_data` for data generation `run_prono_monte_carlo` for Monte Carlo analysis `prono_gmm` for GMM estimation

`summary.lewbel_gmm` *Summary Method for Lewbel GMM*

Description

Summary Method for Lewbel GMM

Usage

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

Arguments

`object`	An object of class "lewbel_gmm".
`...`	Additional arguments passed to summary.gmm.

`validate_rigobon_assumptions` *Validate Rigobon Assumptions*

Description

Tests whether the data satisfies the key assumptions required for Rigobon's (2003) identification strategy.

Usage

```
validate_rigobon_assumptions(
  data,
  regime_var = "regime",
  exog_vars = "Xk",
  verbose = TRUE
)
```

Arguments

`data`	Data.frame. Must contain Y1, Y2, regime, and X variables.
`regime_var`	Character. Name of regime variable (default: "regime").
`exog_vars`	Character vector. Names of exogenous variables.
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Details

The function tests three key assumptions:

1. Heteroskedasticity across regimes in at least one equation
2. The covariance restriction (centered regime dummies uncorrelated with error product)
3. Constant covariance between errors across regimes

Value

A list containing test results:

- regime_heteroskedasticity: Test for different variances across regimes
- covariance_restriction: Test that Cov(Z, e1*e2) = 0
- constant_covariance: Test that Cov(e1, e2) is constant across regimes
- all_valid: Logical indicating if all assumptions are satisfied

References

Rigobon, R. (2003). Identification through heteroskedasticity. Review of Economics and Statistics, 85(4), 777-792. doi:10.1162/003465303772815727

Examples

```
## Not run:
# Generate test data
params <- list(
  beta1_0 = 0.5, beta1_1 = 1.5, gamma1 = -0.8,
  beta2_0 = 1.0, beta2_1 = -1.0,
  alpha1 = -0.5, alpha2 = 1.0,
  regime_probs = c(0.4, 0.6),
  sigma2_regimes = c(1.0, 2.5)
)
data <- generate_rigobon_data(1000, params)

# Validate assumptions
validation <- validate_rigobon_assumptions(data)

## End(Not run)
```

`verify_lewbel_assumptions` *Verify Lewbel's Key Identifying Assumptions*

Description

Tests whether the data generating process satisfies the key assumptions required for Lewbel's (2012) identification strategy. This includes testing the covariance restriction and instrument relevance condition.

Usage

```
verify_lewbel_assumptions(
  data = NULL,
  config = NULL,
  n_obs = 10000,
  params = NULL,
  verbose = TRUE
)
```

Arguments

`data`	Data.frame. Optional. Pre-generated data to verify (alternative to n_obs/params).
`config`	List. Optional. Configuration object (used with data parameter).
`n_obs`	Integer. Sample size for verification (default: 10000, used with params).
`params`	List. Parameters for the data generating process (same format as generate_lewbel_data).
`verbose`	Logical. Whether to print progress messages (default: TRUE).

Details

The function tests:

- Assumption A2: Cov(Z, $\epsilon_1\epsilon_2$) = 0 (covariance restriction)
- Assumption A3: Cov(Z, ϵ_2^2) != 0 (instrument relevance)
- Endogeneity: Cov(ϵ_1, ϵ_2) != 0

Can be called in two ways:

1. verify_lewbel_assumptions(data, config) - using pre-generated data
2. verify_lewbel_assumptions(n_obs = 10000, params = params) - generating new data

Value

Invisibly returns a list with verification results and data.

References

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. Journal of Business & Economic Statistics, 30(1), 67-80. doi:10.1080/07350015.2012.643126

See Also

`generate_lewbel_data` for generating data that meets assumptions

Examples

```
## Not run:
config <- create_default_config()
params <- list(
  beta1_0 = config$beta1_0, beta1_1 = config$beta1_1, gamma1 = config$gamma1,
  beta2_0 = config$beta2_0, beta2_1 = config$beta2_1,
  alpha1 = config$alpha1, alpha2 = config$alpha2,
  delta_het = config$delta_het
)
verify_lewbel_assumptions(params = params)

## End(Not run)
```

Index