Package 'valueprhr'

December 9, 2025

Type Package

Title Value-Price Analysis with Bayesian and Panel Data Methods

Version 0.1.0

Description Provides tools for analyzing the relationship between direct prices (based on labor values) and prices of production using Bayesian generalized linear models, panel data methods, partial least squares regression, canonical correlation analysis, and panel vector autoregression. Includes functions for model comparison, out-of-sample validation, and structural break detection. Here, methods use raw accounting data with explicit temporal structure, following Gomez Julian (2023) <doi:10.17605/OSF.IO/7J8KF> and standard econometric techniques for panel data analysis.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports stats, utils, Metrics

Suggests rstanarm, loo, plm, lme4, pls, vars, panelvar, strucchange, lmtest, sandwich, dplyr, tidyr, tibble, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/isadorenabi/valueprhr

BugReports https://github.com/isadorenabi/valueprhr/issues

NeedsCompilation no

Author Jose Mauricio Gomez Julian [aut, cre] (ORCID: <https://orcid.org/0009-0000-2412-3150>)

Maintainer Jose Mauricio Gomez Julian <isadore.nabi@pm.me>

Repository CRAN

Date/Publication 2025-12-09 08:10:02 UTC

Contents

`aggregate_to_timeseries` *Create Time-Series Aggregated Data*

Description

Aggregates panel data to time series by computing means across sectors.

Usage

```
aggregate_to_timeseries(panel_data, vars = c("log_direct", "log_production"))
```

Arguments

`panel_data`	Data frame with panel data.
`vars`	Character vector of variables to aggregate. Default c("log_direct", "log_production").

Value

Data frame with one row per year containing aggregated values.

Examples

```
set.seed(123)
panel <- data.frame(
  year = rep(2000:2005, 3),
  sector = rep(c("A", "B", "C"), each = 6),
  log_direct = rnorm(18, mean = 5),
  log_production = rnorm(18, mean = 5)
)
ts_agg <- aggregate_to_timeseries(panel)
head(ts_agg)
```

`bayesian_glm` *Bayesian Generalized Linear Models for Sector Analysis*

Description

Functions for fitting Bayesian GLMs sector by sector.

calculate_mode *Calculate Mode Using Kernel Density Estimation*

Description

Estimates the mode of a numeric vector using kernel density estimation. Falls back to median if density estimation fails.

Usage

```
calculate_mode(x)
```

Arguments

x	A numeric vector.

Value

A single numeric value representing the estimated mode.

Examples

```
set.seed(123)
x <- rnorm(100, mean = 5, sd = 1)
calculate_mode(x)
```

cca_pvar *Canonical Correlation Analysis and Panel VAR*

Description

Functions for sparse CCA and panel vector autoregression.

check_package *Check if Required Package is Available*

Description

Checks for package availability and provides informative error message.

Usage

```
check_package(pkg, reason = "this functionality")
```

Arguments

pkg	Character string with package name.
reason	Character string explaining why the package is needed.

Value

TRUE invisibly if package is available, otherwise stops with error.

Examples

```
check_package("stats", "basic statistics")
```

compare_models *Compare All Models*

Description

Generates a comprehensive comparison table of all fitted models.

Usage

```
compare_models(
  twoway_result = NULL,
  mundlak_result = NULL,
  bayes_result = NULL,
  pls_result = NULL
)
```

Arguments

twoway_result	Result from fit_twoway_fe (or NULL).
mundlak_result	Result from fit_mundlak_cre (or NULL).
bayes_result	Result from fit_bayesian_hierarchical (or NULL).
pls_result	Result from fit_pls_multivariate (or NULL).

Value

A data frame with model comparison metrics.

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2019, 5),
    sector = rep(LETTERS[1:5], each = 20),
    log_direct = rnorm(100, 5, 0.5),
    log_production = rnorm(100, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(100, 0, 0.1)

  twoway <- fit_twoway_fe(panel)
  mundlak <- fit_mundlak_cre(panel)

  comparison <- compare_models(
    twoway_result = twoway,
    mundlak_result = mundlak
  )
  print(comparison)
}
```

---

compute_oos_degradation

Compute OOS Degradation

Description

Compares out-of-sample metrics to in-sample metrics.

Usage

```
compute_oos_degradation(insample_metrics, cv_results)
```

Arguments

insample_metrics	Named list with in-sample metrics.
cv_results	Data frame from rolling_window_cv.

Value

Data frame with degradation percentages.

Examples

```
insample <- list(rmse_fe = 0.05, rmse_m = 0.04)
cv_res <- data.frame(
  rmse_fe_all = c(0.06, 0.055, 0.058),
  rmse_m_all = c(0.045, 0.042, 0.044)
)
degradation <- compute_oos_degradation(insample, cv_res)
print(degradation)
```

`compute_r2` *Compute In-Sample R-squared*

Description

Calculates the coefficient of determination (R-squared) for predictions.

Usage

```
compute_r2(actual, predicted)
```

Arguments

`actual`	Numeric vector of actual values.
`predicted`	Numeric vector of predicted values.

Value

A single numeric R-squared value, or NA if not computable.

Examples

```
actual <- c(1, 2, 3, 4, 5)
predicted <- c(1.1, 1.9, 3.1, 4.0, 4.9)
compute_r2(actual, predicted)
```

create_mundlak_data *Create Mundlak-Transformed Panel Data*

Description

Adds sector-level means and within-deviations for Mundlak/CRE estimation.

Usage

```
create_mundlak_data(panel_data, x_var = "log_direct")
```

Arguments

panel_data	Data frame with panel data.
x_var	Character string. Name of the explanatory variable. Default "log_direct".

Value

Panel data with additional columns:

x_mean_sector Sector-level mean of x_var

x_within Within-sector deviation (x - sector mean)

Examples

```
set.seed(123)
panel <- data.frame(
  year = rep(2000:2002, 3),
  sector = rep(c("A", "B", "C"), each = 3),
  log_direct = rnorm(9, mean = 5),
  log_production = rnorm(9, mean = 5)
)
panel_mundlak <- create_mundlak_data(panel)
head(panel_mundlak)
```

data_preparation *Data Preparation Functions*

Description

Functions for preparing and transforming price data for analysis.

evaluate_insample *Evaluate In-Sample Model Performance*

Description

Computes various error metrics comparing predictions to actual values.

Usage

```
evaluate_insample(predicted, actual)
```

Arguments

predicted	Numeric vector of predicted values (log scale).
actual	Numeric vector of actual values (log scale).

Value

A list containing:

mae_log MAE in log scale

rmse_log RMSE in log scale

mae_orig MAE in original scale (after exp transformation)

rmse_orig RMSE in original scale

mae_rel_range MAE as percentage of range

Examples

```
set.seed(123)
actual <- log(runif(50, 100, 200))
predicted <- actual + rnorm(50, 0, 0.1)
evaluate_insample(predicted, actual)
```

export_results_csv *Export Results to CSV*

Description

Exports analysis results to CSV files.

Usage

```
export_results_csv(
  comparison_table = NULL,
  cv_results = NULL,
  sector_summary = NULL,
  output_dir = tempdir(),
  prefix = "valueprhr"
)
```

Arguments

comparison_table	Data frame from compare_models.
cv_results	Data frame from rolling_window_cv.
sector_summary	Data frame from fit_bayesian_glm_sectors.
output_dir	Directory for output files. Default tempdir().
prefix	Filename prefix. Default "valueprhr".

Value

Character vector of created file paths.

Examples

```
comparison <- data.frame(
  model = c("Model A", "Model B"),
  R2 = c(0.95, 0.92)
)
files <- export_results_csv(comparison_table = comparison)
print(files)
```

---

extract_cca_loadings *Extract Top CCA Loadings*

Description

Extracts the variables with highest absolute loadings for each CCA component.

Usage

```
extract_cca_loadings(
  cca_result,
  n_top = 5L,
  which_matrix = c("both", "X", "Y")
)
```

Arguments

cca_result	Result from run_sparse_cca.
n_top	Number of top variables to show. Default 5.
which_matrix	Character. "X", "Y", or "both". Default "both".

Value

Data frame with top loadings.

Examples

```
set.seed(123)
n <- 50
p <- 20
X <- matrix(rnorm(n * p), n, p)
Y <- X %*% matrix(rnorm(p * 5), p, 5) + matrix(rnorm(n * 5, 0, 0.5), n, 5)
colnames(X) <- paste0("X", 1:p)
colnames(Y) <- paste0("Y", 1:5)

cca_res <- run_sparse_cca(X, Y, n_components = 2)
top_loads <- extract_cca_loadings(cca_res)
print(top_loads)
```

---

extract_pls_importance

Extract PLS Variable Importance

Description

Extracts variable importance scores from a fitted PLS model.

Usage

```
extract_pls_importance(pls_result, ncomp = NULL)
```

Arguments

pls_result	Result from fit_pls_multivariate.
ncomp	Number of components to use. Default uses optimal.

Value

Data frame with variable names and importance scores.

Examples

```
if (requireNamespace("pls", quietly = TRUE)) {
  set.seed(123)
  n <- 50
  p <- 10
  X <- matrix(rnorm(n * p), n, p)
  colnames(X) <- paste0("X", 1:p)
  Y <- X[, 1:3] %*% diag(c(1, 0.5, 0.3)) + matrix(rnorm(n * 3, 0, 0.5), n, 3)
  colnames(Y) <- paste0("Y", 1:3)

  result <- fit_pls_multivariate(X, Y, max_components = 5)
  importance <- extract_pls_importance(result)
  print(head(importance))
}
```

---

extract_sector_coefficients

Extract Sector Coefficients

Description

Extracts intercepts and slopes from all sector models.

Usage

```
extract_sector_coefficients(sector_results)
```

Arguments

sector_results	List of sector results from fit_bayesian_glm_sectors.

Value

Data frame with sector names, intercepts, and slopes.

Examples

```
if (requireNamespace("rstanarm", quietly = TRUE)) {
  set.seed(123)
  years <- 2000:2010
  direct <- data.frame(
    Year = years,
    A = 100 + cumsum(rnorm(11)),
    B = 120 + cumsum(rnorm(11))
  )
  production <- data.frame(
    Year = years,
```

```
  A = 102 + cumsum(rnorm(11)),
  B = 118 + cumsum(rnorm(11))
)
results <- fit_bayesian_glm_sectors(direct, production,
                                    chains = 2, iter = 1000)
coefs <- extract_sector_coefficients(results$results)
print(coefs)
}
```

`fit_aggregated_var` *Fit Aggregated VAR Model*

Description

Fits a VAR model on time-aggregated (mean across sectors) data.

Usage

```
fit_aggregated_var(panel_data, max_lags = 6L, difference = TRUE)
```

Arguments

`panel_data`	Data frame in panel format.
`max_lags`	Maximum lag order. Default 6.
`difference`	Logical. Apply first differencing. Default TRUE.

Value

A list containing:

model The fitted vars::VAR model

selected_lag Lag selected by information criteria

irf Impulse response functions (if computed)

fevd Forecast error variance decomposition (if computed)

Examples

```
if (requireNamespace("vars", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2019, 5),
    sector = rep(LETTERS[1:5], each = 20),
    log_direct = rnorm(100, 5, 0.5),
    log_production = rnorm(100, 5, 0.5)
  )
```

```
  result <- fit_aggregated_var(panel)
  print(result$selected_lag)
}
```

`fit_bayesian_glm_sectors` *Fit Bayesian GLM for Each Sector*

Description

Fits separate Bayesian generalized linear models for each sector, regressing production prices on direct prices.

Usage

```
fit_bayesian_glm_sectors(
  direct_prices,
  production_prices,
  chains = 4L,
  iter = 4000L,
  seed = 12345L,
  verbose = TRUE
)
```

Arguments

`direct_prices`	Data frame with direct prices. First column must be 'Year', remaining columns are sector values.
`production_prices`	Data frame with prices of production. Must have same structure as direct_prices.
`chains`	Number of MCMC chains. Default 4.
`iter`	Number of iterations per chain. Default 4000.
`seed`	Random seed for reproducibility. Default 12345.
`verbose`	Logical. Print progress messages. Default TRUE.

Details

This function requires the 'rstanarm' and 'loo' packages to be installed. Each sector model uses a Gaussian family with identity link and weakly informative priors.

Value

A list with two elements:

results List of results for each sector

summary_table Data frame with summary statistics for all sectors

Examples

```
if (requireNamespace("rstanarm", quietly = TRUE)) {
  set.seed(123)
  years <- 2000:2010

  direct <- data.frame(
    Year = years,
    Agriculture = 100 + cumsum(rnorm(11, 2, 1)),
    Manufacturing = 120 + cumsum(rnorm(11, 2, 1))
  )

  production <- data.frame(
    Year = years,
    Agriculture = 102 + cumsum(rnorm(11, 2, 1)),
    Manufacturing = 118 + cumsum(rnorm(11, 2, 1))
  )

  results <- fit_bayesian_glm_sectors(
    direct, production,
    chains = 2, iter = 1000
  )
  print(results$summary_table)
}
```

---

fit_bayesian_hierarchical

Fit Bayesian Hierarchical Panel Model

Description

Fits a Bayesian mixed effects model with random slopes by sector.

Usage

```
fit_bayesian_hierarchical(
  panel_data,
  include_time = TRUE,
  chains = 4L,
  iter = 4000L,
  seed = 12345L
)
```

Arguments

panel_data	Data frame in panel format.
include_time	Logical. Include time trend. Default TRUE.

chains	Number of MCMC chains. Default 4.
iter	Number of iterations. Default 4000.
seed	Random seed. Default 12345.

Value

A list containing:

model The fitted rstanarm model object

r2_bayes Bayesian R-squared (mean)

summary Model summary for fixed effects

metrics In-sample evaluation metrics

Examples

```
## Not run:
if (requireNamespace("rstanarm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2009, 5),
    sector = rep(LETTERS[1:5], each = 10),
    time = rep(1:10, 5),
    log_direct = rnorm(50, 5, 0.5),
    log_production = rnorm(50, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(50, 0, 0.1)

  result <- fit_bayesian_hierarchical(panel, chains = 2, iter = 1000)
  print(result$r2_bayes)
}

## End(Not run)
```

---

fit_mundlak_cre *Fit Mundlak Correlated Random Effects Model*

Description

Fits a Mundlak (CRE) model that decomposes effects into within and between components, allowing for correlation between unit effects and regressors.

Usage

```
fit_mundlak_cre(panel_data, include_time_fe = TRUE, robust_se = TRUE)
```

Arguments

panel_data	Data frame in panel format.
include_time_fe	Logical. Include time fixed effects. Default TRUE.
robust_se	Logical. Compute robust standard errors. Default TRUE.

Details

The Mundlak transformation adds sector-level means of the regressors to a random effects model, allowing consistent estimation even when the random effects are correlated with the regressors.

Value

A list containing:

model The fitted plm model object

summary Model summary

panel_data_augmented Panel data with Mundlak transformations

coeftest_robust Robust coefficient tests

variance_components Random effects variance components

metrics In-sample evaluation metrics

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2009, 5),
    sector = rep(LETTERS[1:5], each = 10),
    log_direct = rnorm(50, 5, 0.5),
    log_production = rnorm(50, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(50, 0, 0.1)

  result <- fit_mundlak_cre(panel)
  print(result$variance_components)
}
```

`fit_panel_var` *Fit Panel VAR Model*

Description

Fits a panel vector autoregression model with first-difference transformation.

Usage

```
fit_panel_var(panel_data, max_lags = 2L, verbose = TRUE)
```

Arguments

`panel_data`	Data frame in panel format.
`max_lags`	Maximum lag order to consider. Default 2.
`verbose`	Logical. Print progress. Default TRUE.

Value

A list containing:

model The fitted panelvar model

best_lag Selected lag order

bic_values BIC for each lag order tested

Examples

```
if (requireNamespace("panelvar", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2019, 5),
    sector = rep(LETTERS[1:5], each = 20),
    log_direct = rnorm(100, 5, 0.5),
    log_production = rnorm(100, 5, 0.5)
  )

  result <- fit_panel_var(panel)
  print(result$best_lag)
}
```

`fit_pls_multivariate` *Fit PLS Regression with Cross-Validation Component Selection*

Description

Fits a partial least squares regression model with automatic selection of the optimal number of components via cross-validation.

Usage

```
fit_pls_multivariate(
  X_matrix,
  Y_matrix,
  max_components = NULL,
  cv_segments = 10L,
  scale = TRUE,
  center = TRUE
)
```

Arguments

`X_matrix`	Numeric matrix of predictor variables (direct prices).
`Y_matrix`	Numeric matrix of response variables (production prices).
`max_components`	Maximum number of components to consider. Default NULL uses min(ncol(X), nrow(X)-1, ncol(Y), 25).
`cv_segments`	Number of cross-validation segments. Default 10.
`scale`	Logical. Scale variables before fitting. Default TRUE.
`center`	Logical. Center variables before fitting. Default TRUE.

Details

This function uses the pls package for PLS regression. Component selection is based on minimizing cross-validated RMSE. The function handles log-transformed data and reports metrics in both log and original scales.

Value

A list containing:

model The fitted pls model object

optimal_ncomp Optimal number of components by CV-RMSE

cv_table Data frame with CV metrics by number of components

metrics_cv CV metrics at optimal component number

metrics_insample In-sample metrics at optimal component number

Examples

```
if (requireNamespace("pls", quietly = TRUE)) {
  set.seed(123)
  n <- 50
  p <- 10
  X <- matrix(rnorm(n * p), n, p)
  colnames(X) <- paste0("X", 1:p)
  Y <- X[, 1:3] %*% diag(c(1, 0.5, 0.3)) + matrix(rnorm(n * 3, 0, 0.5), n, 3)
  colnames(Y) <- paste0("Y", 1:3)

  result <- fit_pls_multivariate(X, Y, max_components = 8)
  print(result$optimal_ncomp)
  print(result$cv_table)
}
```

---

fit_twoway_fe *Fit Two-Way Fixed Effects Panel Model*

Description

Fits a two-way fixed effects model with sector and time effects, regressing log production prices on log direct prices.

Usage

```
fit_twoway_fe(
  panel_data,
  robust_se = TRUE,
  cluster_type = c("group", "time", "twoway")
)
```

Arguments

panel_data	Data frame in panel format with columns: year, sector, log_direct, log_production.
robust_se	Logical. Compute robust standard errors. Default TRUE.
cluster_type	Character. Type of cluster for robust SE. One of "group", "time", or "twoway". Default "group".

Details

This function requires the 'plm' package. The model specification is: log_production ~ log_direct with two-way (sector and time) fixed effects.

Value

A list containing:

model The fitted plm model object

summary Model summary

r2_within Within R-squared

coeftest_robust Coefficient test with robust SE (if robust_se=TRUE)

metrics In-sample evaluation metrics

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2009, 5),
    sector = rep(LETTERS[1:5], each = 10),
    log_direct = rnorm(50, 5, 0.5),
    log_production = rnorm(50, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(50, 0, 0.1)

  result <- fit_twoway_fe(panel)
  print(result$r2_within)
}
```

---

format_break_results *Format Structural Break Results*

Description

Creates a formatted summary of structural break test results.

Usage

```
format_break_results(break_results, alpha = 0.05)
```

Arguments

break_results	Result from test_structural_breaks.
alpha	Significance level for interpretation. Default 0.05.

Value

A data frame with formatted test summaries.

Examples

```
if (requireNamespace("strucchange", quietly = TRUE)) {
  set.seed(123)
  years <- 1980:2020
  panel <- data.frame(
    year = rep(years, 5),
    sector = rep(LETTERS[1:5], each = length(years)),
    log_direct = rnorm(length(years) * 5, 5, 0.5),
    log_production = rnorm(length(years) * 5, 5, 0.5)
  )

  break_tests <- test_structural_breaks(panel)
  summary_df <- format_break_results(break_tests)
  print(summary_df)
}
```

---

generate_analysis_summary

Generate Comprehensive Analysis Summary

Description

Creates a complete summary of all analysis results including models, validation, and structural break tests.

Usage

```
generate_analysis_summary(
  comparison_table,
  cv_summary = NULL,
  break_results = NULL,
  cca_results = NULL,
  granger_results = NULL
)
```

Arguments

comparison_table	Data frame from compare_models.
cv_summary	Data frame from summarize_cv_results (or NULL).
break_results	Result from test_structural_breaks (or NULL).
cca_results	Result from run_sparse_cca (or NULL).
granger_results	Data frame from panel_granger_test (or NULL).

Value

A list with formatted summary components.

Examples

```
comparison <- data.frame(
  model = c("Model A", "Model B"),
  R2 = c(0.95, 0.92),
  RMSE_log = c(0.05, 0.06)
)
summary_list <- generate_analysis_summary(comparison)
print(summary_list$best_model)
```

---

get_r2_values *Extract R-squared Values from rstanarm Model*

Description

Attempts to extract R-squared using loo_R2, falling back to bayes_R2.

Usage

```
get_r2_values(fit, verbose = FALSE)
```

Arguments

fit	A fitted rstanarm model object.
verbose	Logical. Print messages about extraction method. Default FALSE.

Value

A named numeric vector with mean, median, and mode R-squared values.

Examples

```
if (requireNamespace("rstanarm", quietly = TRUE)) {
  data(mtcars)
  fit <- rstanarm::stan_glm(mpg ~ wt, data = mtcars,
                            chains = 2, iter = 1000, refresh = 0)
  get_r2_values(fit)
}
```

interpret_break_tests *Interpret Break Test Results*

Description

Provides textual interpretation of structural break tests.

Usage

```
interpret_break_tests(break_results, alpha = 0.05)
```

Arguments

break_results	Result from test_structural_breaks.
alpha	Significance level. Default 0.05.

Value

Character string with interpretation.

Examples

```
if (requireNamespace("strucchange", quietly = TRUE)) {
  set.seed(123)
  years <- 1980:2020
  panel <- data.frame(
    year = rep(years, 5),
    sector = rep(LETTERS[1:5], each = length(years)),
    log_direct = rnorm(length(years) * 5, 5, 0.5),
    log_production = rnorm(length(years) * 5, 5, 0.5)
  )

  break_tests <- test_structural_breaks(panel)
  interpretation <- interpret_break_tests(break_tests)
  cat(interpretation)
}
```

leave_one_sector_out *Leave-One-Sector-Out Cross-Validation*

Description

Performs LOSO CV, leaving out each sector in turn as the test set.

Usage

```
leave_one_sector_out(panel_data, verbose = TRUE)
```

Arguments

panel_data	Data frame in panel format.
verbose	Logical. Print progress. Default TRUE.

Value

A data frame with RMSE and MAE for each held-out sector.

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2019, 5),
    sector = rep(LETTERS[1:5], each = 20),
    log_direct = rnorm(100, 5, 0.5),
    log_production = rnorm(100, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(100, 0, 0.1)

  loso_results <- leave_one_sector_out(panel)
  print(loso_results)
}
```

---

panel_granger_test *Panel Granger Causality Test (Dumitrescu-Hurlin)*

Description

Performs panel Granger causality tests between direct and production prices.

Usage

```
panel_granger_test(panel_data, lags = c(1L, 2L))
```

Arguments

panel_data	Data frame in panel format.
lags	Integer vector of lag orders to test. Default c(1, 2).

Details

Tests both directions: direct -> production and production -> direct.

Value

A data frame with test results for each direction and lag.

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2019, 5),
    sector = rep(LETTERS[1:5], each = 20),
    log_direct = rnorm(100, 5, 0.5),
    log_production = rnorm(100, 5, 0.5)
  )

  granger_results <- panel_granger_test(panel)
  print(granger_results)
}
```

---

panel_models *Panel Data Models for Value-Price Analysis*

Description

Functions for fitting two-way fixed effects and Mundlak CRE models.

---

pls_analysis *Partial Least Squares Regression Analysis*

Description

Functions for multivariate PLS regression with cross-validation.

predict_pls *Predict from PLS Model*

Description

Generate predictions from a fitted PLS model.

Usage

```
predict_pls(pls_result, newdata = NULL, ncomp = NULL)
```

Arguments

pls_result	Result from fit_pls_multivariate.
newdata	Optional new data matrix for prediction.
ncomp	Number of components to use. Default uses optimal.

Value

Matrix of predictions.

Examples

```
if (requireNamespace("pls", quietly = TRUE)) {
  set.seed(123)
  n <- 50
  p <- 10
  X <- matrix(rnorm(n * p), n, p)
  colnames(X) <- paste0("X", 1:p)
  Y <- X[, 1:3] %*% diag(c(1, 0.5, 0.3)) + matrix(rnorm(n * 3, 0, 0.5), n, 3)
  colnames(Y) <- paste0("Y", 1:3)

  result <- fit_pls_multivariate(X, Y, max_components = 5)
  preds <- predict_pls(result)
  dim(preds)
}
```

`prepare_log_matrices` *Prepare Log-Transformed Matrices*

Description

Extracts numeric columns from price data frames and applies log transform.

Usage

```
prepare_log_matrices(
  direct_prices,
  production_prices,
  exclude_cols = c("Year")
)
```

Arguments

`direct_prices`	Data frame with direct prices.
`production_prices`	Data frame with prices of production.
`exclude_cols`	Character vector of columns to exclude. Default c("Year").

Value

A list containing:

X_log Log-transformed matrix of direct prices

Y_log Log-transformed matrix of production prices

complete_cases Logical vector indicating complete cases

X_clean Subset of X_log with complete cases

Y_clean Subset of Y_log with complete cases

Examples

```
set.seed(123)
direct <- data.frame(
  Year = 2000:2005,
  A = runif(6, 100, 200),
  B = runif(6, 100, 200)
)
production <- data.frame(
  Year = 2000:2005,
  A = runif(6, 100, 200),
  B = runif(6, 100, 200)
)
matrices <- prepare_log_matrices(direct, production)
str(matrices)
```

prepare_panel_data *Prepare Panel Data from Wide Format Matrices*

Description

Converts wide-format price matrices (with Year as first column and sectors as subsequent columns) into long-format panel data suitable for panel regression analysis.

Usage

```
prepare_panel_data(direct_prices, production_prices, log_transform = TRUE)
```

Arguments

direct_prices	Data frame with direct prices (labor value-based). First column must be 'Year', remaining columns are sector values.
production_prices	Data frame with prices of production. Must have same structure as direct_prices.
log_transform	Logical. Apply natural log transformation. Default TRUE.

Value

A data frame in panel (long) format with columns:

year Year of observation

sector Sector identifier

sector_id Numeric sector identifier

time Time index (year minus minimum year plus 1)

direct_price Direct price value

production_price Price of production value

log_direct Log of direct price (if log_transform = TRUE)

log_production Log of production price (if log_transform = TRUE)

Examples

```
set.seed(123)
years <- 2000:2010
sectors <- c("Agriculture", "Manufacturing", "Services")

direct <- data.frame(
  Year = years,
  Agriculture = 100 + cumsum(rnorm(11)),
  Manufacturing = 120 + cumsum(rnorm(11)),
  Services = 90 + cumsum(rnorm(11))
)
```

```
production <- data.frame(
  Year = years,
  Agriculture = 102 + cumsum(rnorm(11)),
  Manufacturing = 118 + cumsum(rnorm(11)),
  Services = 92 + cumsum(rnorm(11))
)

panel <- prepare_panel_data(direct, production)
head(panel)
```

`print_analysis_summary` *Print Analysis Summary*

Description

Prints a formatted summary of analysis results to console.

Usage

```
print_analysis_summary(summary_list, verbose = TRUE)
```

Arguments

`summary_list`	Result from generate_analysis_summary.
`verbose`	Logical. Print detailed output. Default TRUE.

Value

Invisible NULL.

Examples

```
comparison <- data.frame(
  model = c("Two-Way FE", "Mundlak CRE"),
  R2 = c(0.95, 0.92),
  RMSE_log = c(0.05, 0.06)
)
summary_list <- generate_analysis_summary(comparison)
print_analysis_summary(summary_list)
```

`robust_summary` *Robust Summary Statistics with Bootstrap Confidence Intervals*

Description

Computes mean, standard deviation, median, MAD, trimmed mean, and bootstrap confidence intervals for a numeric vector.

Usage

```
robust_summary(x, bootstrap_reps = 200L, trim_proportion = 0.1)
```

Arguments

`x`	Numeric vector.
`bootstrap_reps`	Number of bootstrap replications for CI. Default 200.
`trim_proportion`	Proportion to trim for trimmed mean. Default 0.10.

Value

A list containing:

mean Arithmetic mean

sd Standard deviation

median Median

mad Median Absolute Deviation (scaled)

tmean Trimmed mean

ci Vector of length 2 with 95 percent bootstrap CI bounds

Examples

```
set.seed(123)
x <- rnorm(50)
robust_summary(x)
```

`rolling_window_cv` *Rolling Window Cross-Validation*

Description

Performs time-series cross-validation using rolling windows, comparing fixed effects and Mundlak models.

Usage

```
rolling_window_cv(
  panel_data,
  window_sizes = c(20L, 30L),
  step_size = 2L,
  test_horizon = 3L,
  verbose = TRUE
)
```

Arguments

`panel_data`	Data frame in panel format.
`window_sizes`	Integer vector of training window sizes. Default c(20, 30).
`step_size`	Integer step between windows. Default 2.
`test_horizon`	Integer number of periods to forecast. Default 3.
`verbose`	Logical. Print progress. Default TRUE.

Details

For each rolling window, the function fits both a fixed effects model (by sector) and a Mundlak CRE model, then evaluates predictions on the test period. Results are separated by whether test sectors were seen during training (common) or not (new).

Value

A data frame with validation results for each window, including:

window_size Training window size

window_start Start year of training window

window_end End year of training window

rmse_fe_all RMSE for FE model on all test observations

rmse_m_all RMSE for Mundlak model on all test observations

n_test_common Number of test obs from sectors in training

n_test_new Number of test obs from new sectors

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2029, 5),
    sector = rep(LETTERS[1:5], each = 30),
    log_direct = rnorm(150, 5, 0.5),
    log_production = rnorm(150, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(150, 0, 0.1)

  cv_results <- rolling_window_cv(panel, window_sizes = c(15, 20))
  print(head(cv_results))
}
```

---

run_cca_var_analysis *Run Complete CCA and VAR Analysis*

Description

Convenience function to run both sparse CCA and panel/aggregated VAR.

Usage

```
run_cca_var_analysis(
  direct_prices,
  production_prices,
  panel_data,
  cca_components = 3L,
  verbose = TRUE
)
```

Arguments

direct_prices	Data frame with direct prices.
production_prices	Data frame with production prices.
panel_data	Data frame in panel format.
cca_components	Number of CCA components. Default 3.
verbose	Logical. Print progress. Default TRUE.

Value

A list with cca, pvar, agg_var, and granger results.

Examples

```
set.seed(123)
years <- 2000:2019
sectors <- LETTERS[1:5]

direct <- data.frame(Year = years)
production <- data.frame(Year = years)
for (s in sectors) {
  direct[[s]] <- 100 + cumsum(rnorm(20, 2, 1))
  production[[s]] <- 102 + cumsum(rnorm(20, 2, 1))
}

panel <- prepare_panel_data(direct, production)

matrices <- prepare_log_matrices(direct, production)

result <- run_cca_var_analysis(
  direct, production, panel,
  cca_components = 2
)
```

---

run_full_analysis *Run Complete Analysis Pipeline*

Description

Convenience function to run the full analysis pipeline.

Usage

```
run_full_analysis(
  direct_prices,
  production_prices,
  run_bayesian = FALSE,
  run_cv = TRUE,
  run_breaks = TRUE,
  verbose = TRUE
)
```

Arguments

direct_prices	Data frame with direct prices.
production_prices	Data frame with production prices.
run_bayesian	Logical. Run Bayesian models. Default FALSE.
run_cv	Logical. Run cross-validation. Default TRUE.

run_breaks	Logical. Run structural break tests. Default TRUE.
verbose	Logical. Print progress. Default TRUE.

Value

A list with all analysis results.

Examples

```
set.seed(123)
years <- 2000:2019
sectors <- LETTERS[1:5]

direct <- data.frame(Year = years)
production <- data.frame(Year = years)
for (s in sectors) {
  direct[[s]] <- 100 + cumsum(rnorm(20, 2, 1))
  production[[s]] <- 102 + cumsum(rnorm(20, 2, 1))
}

if (requireNamespace("plm", quietly = TRUE)) {
  results <- run_full_analysis(
    direct, production,
    run_bayesian = FALSE,
    run_cv = FALSE
  )
  print(results$comparison)
}
```

---

run_sparse_cca *Run Sparse CCA with PCA Preprocessing*

Description

Performs canonical correlation analysis on PCA-reduced price matrices, with optional sparsity penalties.

Usage

```
run_sparse_cca(
  X_matrix,
  Y_matrix,
  n_components = 3L,
  variance_threshold = 0.9,
  min_pcs = 8L,
  max_pcs = 12L
)
```

Arguments

X_matrix	Numeric matrix of direct prices.
Y_matrix	Numeric matrix of production prices.
n_components	Number of canonical components to extract. Default 3.
variance_threshold	Cumulative variance threshold for PCA. Default 0.90.
min_pcs	Minimum number of PCs to retain. Default 8.
max_pcs	Maximum number of PCs to retain. Default 12.

Details

The function first reduces dimensionality using PCA, then applies CCA. Falls back to base R cancor if specialized packages unavailable.

Value

A list containing:

method Method used for CCA

correlations Canonical correlations

U_loadings X loadings in PC space

V_loadings Y loadings in PC space

W_X_original X loadings projected to original variables

W_Y_original Y loadings projected to original variables

n_pcs_x Number of PCs used for X

n_pcs_y Number of PCs used for Y

Examples

```
set.seed(123)
n <- 50
p <- 20
X <- matrix(rnorm(n * p), n, p)
Y <- X %*% matrix(rnorm(p * 5), p, 5) + matrix(rnorm(n * 5, 0, 0.5), n, 5)
colnames(X) <- paste0("X", 1:p)
colnames(Y) <- paste0("Y", 1:5)

result <- run_sparse_cca(X, Y, n_components = 2)
print(result$correlations)
```

safe_mae *Safe MAE Calculation*

Description

Calculates Mean Absolute Error handling non-finite values.

Usage

```
safe_mae(actual, predicted)
```

Arguments

actual	Numeric vector of actual values.
predicted	Numeric vector of predicted values.

Value

A single numeric value for MAE, or NA if calculation not possible.

Examples

```
actual <- c(1, 2, 3, 4, 5)
predicted <- c(1.1, 2.2, 2.9, 4.1, 5.2)
safe_mae(actual, predicted)
```

safe_pct *Safe Percentage Calculation*

Description

Calculates percentage while handling division by zero and non-finite values.

Usage

```
safe_pct(numerator, denominator)
```

Arguments

numerator	Numeric vector for the numerator.
denominator	Numeric vector for the denominator.

Value

Numeric vector of percentages, with NA for invalid calculations.

Examples

```
safe_pct(c(10, 20, 30), c(100, 0, 200))
```

`safe_rmse` *Safe RMSE Calculation*

Description

Calculates Root Mean Squared Error handling non-finite values.

Usage

```
safe_rmse(actual, predicted)
```

Arguments

`actual`	Numeric vector of actual values.
`predicted`	Numeric vector of predicted values.

Value

A single numeric value for RMSE, or NA if calculation not possible.

Examples

```
actual <- c(1, 2, 3, 4, 5)
predicted <- c(1.1, 2.2, 2.9, 4.1, 5.2)
safe_rmse(actual, predicted)
```

`structural_breaks` *Structural Break Tests*

Description

Functions for testing structural breaks in the price relationship.

summarize_cv_results *Summarize Rolling Window Results*

Description

Computes summary statistics from rolling window cross-validation.

Usage

```
summarize_cv_results(cv_results, bootstrap_reps = 300L)
```

Arguments

cv_results	Data frame from rolling_window_cv.
bootstrap_reps	Number of bootstrap replications. Default 300.

Value

A data frame with summary statistics by model and partition.

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2029, 5),
    sector = rep(LETTERS[1:5], each = 30),
    log_direct = rnorm(150, 5, 0.5),
    log_production = rnorm(150, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(150, 0, 0.1)

  cv_results <- rolling_window_cv(panel, window_sizes = c(15, 20))
  summary_stats <- summarize_cv_results(cv_results)
  print(summary_stats)
}
```

summary_comparison *Model Comparison and Summary Functions*

Description

Functions for comparing models and generating summary tables.

`test_mundlak_specification` *Test Mundlak Specification*

Description

Performs Wald test on the sector-mean coefficient to test whether fixed effects would be preferred over random effects.

Usage

```
test_mundlak_specification(mundlak_result)
```

Arguments

`mundlak_result`	Result from fit_mundlak_cre.

Details

Under the null hypothesis that the sector means coefficient equals zero, random effects would be appropriate. Rejection suggests fixed effects should be used.

Value

A list with test statistic, degrees of freedom, and p-value.

Examples

```
if (requireNamespace("plm", quietly = TRUE)) {
  set.seed(123)
  panel <- data.frame(
    year = rep(2000:2009, 5),
    sector = rep(LETTERS[1:5], each = 10),
    log_direct = rnorm(50, 5, 0.5),
    log_production = rnorm(50, 5, 0.5)
  )
  panel$log_production <- panel$log_direct * 0.95 + rnorm(50, 0, 0.1)

  mundlak_fit <- fit_mundlak_cre(panel)
  test_result <- test_mundlak_specification(mundlak_fit)
  print(test_result)
}
```

`test_structural_breaks` *Test for Structural Breaks*

Description

Performs multiple structural break tests on the aggregated time series relationship between direct and production prices.

Usage

```
test_structural_breaks(panel_data, chow_years = NULL, min_segment = 10L)
```

Arguments

`panel_data`	Data frame in panel format.
`chow_years`	Integer vector of candidate break years for Chow test. Default NULL uses 1986, 1997, 2001, 2008 if present.
`min_segment`	Integer minimum observations per segment. Default 10.

Details

This function aggregates panel data to a single time series by taking means across sectors, then applies various structural break tests from the strucchange package.

Value

A list containing:

chow Chow test results for candidate years

cusum CUSUM test results

mosum MOSUM test results

supf supremum F test results

breakpoints Estimated breakpoint dates

aggregated_data The aggregated time series used

Examples

```
if (requireNamespace("strucchange", quietly = TRUE)) {
  set.seed(123)
  years <- 1980:2020
  panel <- data.frame(
    year = rep(years, 5),
    sector = rep(LETTERS[1:5], each = length(years)),
    log_direct = rnorm(length(years) * 5, 5, 0.5),
    log_production = rnorm(length(years) * 5, 5, 0.5)
```

```
  )

  break_tests <- test_structural_breaks(panel)
  print(break_tests$cusum)
}
```

`utils` *Utility Functions for Value-Price Analysis*

Description

Internal utility functions used across the package.

`validate_panel_data` *Validate Panel Data Structure*

Description

Checks that panel data has required columns and valid structure.

Usage

```
validate_panel_data(panel_data, require_log = TRUE)
```

Arguments

`panel_data`	Data frame to validate.
`require_log`	Logical. Check for log-transformed columns. Default TRUE.

Value

TRUE invisibly if valid, otherwise stops with informative error.

Examples

```
set.seed(123)
panel <- data.frame(
  year = rep(2000:2002, 3),
  sector = rep(c("A", "B", "C"), each = 3),
  log_direct = rnorm(9),
  log_production = rnorm(9)
)
validate_panel_data(panel)
```

validation *Out-of-Sample Validation Functions*

Description

Functions for rolling window and leave-one-sector-out validation.

Index