Package: dcce (via r-universe)

October 24, 2026

Title Dynamic Common Correlated Effects Estimation for Panel Data

Version 0.4.2

Description Estimates heterogeneous coefficient models for large panels with cross-sectional dependence. Implements the Mean Group (MG) estimator of Pesaran and Smith (1995) <doi:10.1016/0304-4076(94)01644-F>, the Common Correlated Effects (CCE) and Dynamic CCE (DCCE) estimators of Pesaran (2006) <doi:10.1111/j.1468-0262.2006.00692.x> and Chudik and Pesaran (2015) <doi:10.1016/j.jeconom.2015.03.007>, the regularized CCE of Juodis (2022), the Augmented Mean Group (AMG) of Eberhardt and Teal (2010), the Interactive Fixed Effects (IFE) estimator of Bai (2009) <doi:10.3982/ECTA6135>, and long-run estimators including Cross-Sectionally augmented Distributed Lag (CS-DL), Cross-Sectionally augmented Autoregressive Distributed Lag (CS-ARDL), and Pooled Mean Group (PMG) (Chudik et al. 2016; Shin et al. 1999). Also provides rolling-window estimation, high-dimensional fixed effect absorption, spatial CCE via user-supplied weight matrices, and structural break tests (Chow and sup-Wald) following Andrews (1993), Bai and Perron (1998), and Ditzen, Karavias and Westerlund (2024). Supplies a comprehensive cross-sectional dependence (CD) test suite including the Pesaran (2015) CD test <doi:10.1080/07474938.2014.956623>, the Juodis and Reese (2022) randomized weighted CD (CDw) test, the Baltagi et al. (2012) bias-adjusted weighted CD (CDw+) test, the Fan et al. (2015) Power Enhancement Approach (PEA) test, and the Pesaran and Xie (2021) bias-corrected CD (CD*) test. Further diagnostics include the Pesaran (2007) Cross-sectionally Augmented IPS (CIPS) panel unit root test <doi:10.1002/jae.951>, the Westerlund (2007) panel cointegration tests, the Dumitrescu and Hurlin (2012) panel Granger causality test, the Im-Pesaran-Shin (IPS) and Levin-Lin-Chu (LLC) panel unit root tests, the Pedroni (2004) and Kao (1999) residual cointegration tests, the Swamy (1970) and Pesaran and Yamagata (2008) slope homogeneity tests, a Hausman-type test for MG versus pooled, the exponent of cross-sectional dependence from Bailey et al. (2016)

<doi:10.1002/jae.2490>, information criteria for Cross-Sectional Average (CSA) selection, the rank condition classifier, impulse response functions, cross-section and wild bootstrap inference, and 'broom'-compatible methods.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports stats, Matrix, collapse (>= 2.0.0), sandwich, generics, rlang (>= 1.1.0), cli (>= 3.0.0), tibble, Rcpp (>= 1.0.0)

LinkingTo Rcpp, RcppArmadillo

Suggests broom, ggplot2, lifecycle, plm, testthat (>= 3.0.0), knitr, rmarkdown, marginaleffects, parallel

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Mustapha Wasseja [aut, cre]

Maintainer Mustapha Wasseja <muswaseja@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-05 20:36:34 UTC

RemoteUrl https://github.com/cran/dcce

RemoteRef HEAD

RemoteSha 8d28b601d40a6f43f926f0cfbfefefafecd2d157

Contents

bootstrap . . . 3
cips_test . . . 5
coef.dcce_fit . . . 7
cointegration_test . . . 7
confint.dcce_fit . . . 9
csd_exp . . . 10
D . . . 11
dcce . . . 12
dcce_bootstrap . . . 14
dcce_rolling . . . 15
dcce_sim . . . 17
dcce_sim_truth . . . 18
dcce_workflow . . . 18
fitted.dcce_fit . . . 20
glance.dcce_fit . . . 20

granger_test . . . 21
hausman_test . . . 22
irf . . . 23
L . . . 24
Lrange . . . 24
panel_coint_test . . . 25
panel_ur_test . . . 26
pcd_test . . . 28
plot.dcce_fit . . . 30
plot.dcce_irf . . . 30
plot.dcce_rolling . . . 31
predict.dcce_fit . . . 31
print.dcce_boot . . . 32
print.dcce_break . . . 32
print.dcce_cd . . . 33
print.dcce_cips . . . 33
print.dcce_cointegration . . . 34
print.dcce_cointegration_extra . . . 34
print.dcce_csd . . . 35
print.dcce_fit . . . 35
print.dcce_granger . . . 36
print.dcce_hausman . . . 36
print.dcce_irf . . . 37
print.dcce_rolling . . . 37
print.dcce_swamy . . . 38
print.dcce_unit_root . . . 38
print.dcce_workflow . . . 39
pwt8 . . . 39
rank_condition . . . 40
residuals.dcce_fit . . . 41
structural_break_test . . . 41
summary.dcce_fit . . . 43
swamy_test . . . 44
tidy.dcce_fit . . . 45
update.dcce_fit . . . 45
vcov.dcce_fit . . . 46

Index **47**

`bootstrap` *Bootstrap Inference for DCCE Models*

Description

Computes bootstrap standard errors and confidence intervals for `dcce_fit` objects using either cross-section or wild bootstrap.

Usage

```
bootstrap(
  object,
  type = c("crosssection", "wild"),
  reps = 500L,
  percentile = TRUE,
  cfresiduals = FALSE,
  seed = NULL
)
```

Arguments

object	A dcce_fit object.
type	Character: "crosssection" (default) or "wild".
reps	Integer: number of bootstrap repetitions. Default 500.
percentile	Logical: compute percentile CIs? Default TRUE.
cfresiduals	Logical: for wild bootstrap, use common-factor residuals instead of defactored residuals? Default FALSE.
seed	Integer: random seed for reproducibility. Default NULL.

Value

An object of class dcce_boot with elements:

se_boot Bootstrap standard errors.

ci_lower Percentile CI lower bound (if percentile=TRUE).

ci_upper Percentile CI upper bound.

b_boot B x K matrix of bootstrap coefficient draws.

reps Number of repetitions.

type Bootstrap type.

Note

Naming conflict with broom::bootstrap. The broom package also exports a function called bootstrap() (for resampling data frames), with a completely different signature. If you load broom after dcce, broom::bootstrap will mask dcce::bootstrap on the search path and calls to bootstrap(fit, type = ..., reps = ...) will fail with an "unused arguments" error. To avoid the conflict you can either (a) use the namespace prefix dcce::bootstrap(fit, ...), (b) load broom **before** dcce so dcce ends up higher in the search path, or (c) use the conflict-free alias dcce_bootstrap(fit, ...) which is exported by dcce and has the same semantics.

Examples

```
set.seed(42)
df <- data.frame(
  id = rep(1:10, each = 30),
```

```
  t  = rep(1:30, 10),
  y  = rnorm(300),
  x  = rnorm(300)
)
fit <- dcce(df, "id", "t", y ~ x, model = "mg", cross_section_vars = NULL)
boot_res <- bootstrap(fit, reps = 50)
print(boot_res)
```

`cips_test` *Pesaran CIPS Panel Unit Root Test*

Description

Implements the cross-sectionally augmented IPS (CIPS) panel unit root test of Pesaran (2007), which allows for cross-sectional dependence through a single unobserved common factor. For each unit, a cross-sectionally augmented Dickey-Fuller (CADF) regression is run:

Usage

```
cips_test(x, ...)

## S3 method for class 'matrix'
cips_test(x, ..., lags = 0L, trend = FALSE)

## S3 method for class 'dcce_fit'
cips_test(x, ..., lags = 0L, trend = FALSE)

## Default S3 method:
cips_test(
  x,
  ...,
  data = NULL,
  unit_index = NULL,
  time_index = NULL,
  lags = 0L,
  trend = FALSE
)
```

Arguments

`x`	A numeric vector, numeric matrix (N x T), data.frame, or `dcce_fit` object. If a vector, `data`, `unit_index`, and `time_index` must also be supplied.
`...`	Additional arguments passed to methods.
`lags`	Integer: number of lags of Δy to include in the CADF regression. Default 0 (pure CADF without augmentation).
`trend`	Logical: include a linear time trend? Default `FALSE`.

`data`	A data.frame containing the panel structure (when `x` is a vector).
`unit_index`	Character: name of the unit variable in `data`.
`time_index`	Character: name of the time variable in `data`.

Details

$$\Delta y_{it} = a_i + b_i y_{i,t-1} + c_i \bar{y}_{t-1} + d_i \Delta \bar{y}_t + \sum_{j=1}^{p} \rho_{ij} \Delta y_{i,t-j} + \sum_{j=0}^{p} \delta_{ij} \Delta \bar{y}_{t-j} + u_{it}.$$

The CIPS statistic is the cross-sectional average of the unit-level t-statistics for $b_i = 0$ (the CADF statistic). Critical values come from Pesaran (2007, Table II(b), constant case) and Pesaran (2007, Table II(c), constant + trend case). The null hypothesis is that all series contain a unit root.

Value

An object of class `dcce_cips` with elements:

statistic The CIPS statistic (truncated cross-sectional average).

p_value Approximate p-value from Pesaran (2007) critical values.

unit_stats Per-unit truncated CADF t-statistics.

N Number of units.

T Time dimension.

lags Number of augmentation lags.

trend Whether a trend was included.

References

Pesaran, M. H. (2007). A simple panel unit root test in the presence of cross-section dependence. *Journal of Applied Econometrics*, 22(2), 265-312.

Examples

```
set.seed(1)
N <- 20; T <- 30
f <- cumsum(rnorm(T))
X <- matrix(NA, N, T)
for (i in seq_len(N)) X[i, ] <- cumsum(rnorm(T)) + 0.5 * f
cips_test(X)
```

coef.dcce_fit *Extract coefficients from a dcce_fit object*

Description

Extract coefficients from a dcce_fit object

Usage

```
## S3 method for class 'dcce_fit'
coef(object, type = c("mg", "unit"), ...)
```

Arguments

object	A dcce_fit object.
type	Character: "mg" for Mean Group coefficients (default), "unit" for unit-level coefficients as a tibble.
...	Ignored.

Value

A named numeric vector (for "mg") or a tibble (for "unit").

cointegration_test *Westerlund (2007) Panel Cointegration Tests*

Description

Implements the four error-correction-based panel cointegration tests of Westerlund (2007): two group-mean statistics (Ga, Gt) and two panel statistics (Pa, Pt). The null hypothesis is **no cointegration** between the dependent variable and the regressors.

Usage

```
cointegration_test(
  data,
  unit_index,
  time_index,
  formula,
  lags = 1L,
  leads = 1L,
  test = c("ga", "gt", "pa", "pt"),
  n_bootstrap = 0L,
  seed = NULL,
  show_progress = FALSE
)
```

Arguments

`data`	A panel data.frame.
`unit_index`	Character scalar: unit identifier column.
`time_index`	Character scalar: time identifier column.
`formula`	Two-sided formula of the form `y ~ x1 + x2` in **levels** (not first differences).
`lags`	Integer scalar: ADF lag order for Δy lags in the ECM regression. Default `1L`.
`leads`	Integer scalar: number of leads of Δx. Default `1L`.
`test`	Character vector: which statistics to compute. Subset of `c("ga", "gt", "pa", "pt")`. Default: all four.
`n_bootstrap`	Integer: number of bootstrap replications for p-values. Default `0L` (asymptotic p-values).
`seed`	Integer: random seed for the bootstrap.
`show_progress`	Logical: print progress? Default `FALSE`.

Details

For each unit i, an error-correction regression is fitted

$$\Delta y_{it} = \delta_i d_t + \alpha_i y_{i,t-1} + \lambda_i' x_{i,t-1} + \sum_{j=1}^{p_i} \alpha_{ij} \Delta y_{i,t-j} + \sum_{j=-q_i}^{p_i} \gamma_{ij}' \Delta x_{i,t-j} + e_{it},$$

and the t-statistic for $\alpha_i = 0$ is used to construct the four test statistics:

- **Gt**: group-mean of the $\hat{\alpha}_i / SE(\hat{\alpha}_i)$.
- **Ga**: group-mean of $T\hat{\alpha}_i / \hat{\alpha}_i(1)$ (unnormalised form).
- **Pt**: pooled t-statistic.
- **Pa**: pooled alpha-statistic.

A negative, large-magnitude statistic rejects the null of no cointegration. Asymptotic p-values are obtained by linear interpolation on the critical values in Westerlund (2007, Table 3). A bootstrap path (`n_bootstrap > 0`) is also provided, resampling cross-sectional units with replacement to account for cross-sectional dependence.

Value

An object of class `dcce_cointegration` with elements `statistics` (a tibble with columns `test`, `statistic`, `p_value`, `method`), `lags`, `leads`, `N`, `T_bar`, and `call`.

References

Westerlund, J. (2007). Testing for Error Correction in Panel Data. *Oxford Bulletin of Economics and Statistics*, 69(6), 709-748.

Examples

```
data(pwt8)
result <- cointegration_test(
  data       = pwt8,
  unit_index = "country",
  time_index = "year",
  formula    = log_rgdpo ~ log_hc + log_ck,
  test       = c("ga", "gt"),
  lags       = 1L
)
print(result)
```

`confint.dcce_fit` *Confidence intervals for a dcce_fit object*

Description

Confidence intervals for a dcce_fit object

Usage

```
## S3 method for class 'dcce_fit'
confint(
  object,
  parm = NULL,
  level = 0.95,
  type = c("mg", "lr", "adjustment"),
  ...
)
```

Arguments

`object`	A `dcce_fit` object.
`parm`	Character vector of parameter names (default: all).
`level`	Confidence level. Default `0.95`.
`type`	Character: `"mg"` (default) for the main MG coefficients, `"lr"` for long-run coefficients (CS-ARDL, CS-DL, PMG), `"adjustment"` for the speed of adjustment (CS-ARDL, PMG).
`...`	Ignored.

Value

A matrix of confidence intervals with rows corresponding to parameters.

csd_exp *Exponent of Cross-Sectional Dependence*

Description

Estimates the exponent of cross-sectional dependence (alpha) using the methods of Bailey, Kapetanios & Pesaran (2016, 2019).

Usage

```
csd_exp(
  x,
  data = NULL,
  unit_index = NULL,
  time_index = NULL,
  use_residuals = FALSE,
  n_pca = 1L,
  test_size = 0.1,
  tuning = 0.5,
  n_bootstrap = 200L
)
```

Arguments

x	Either a numeric vector (variable stacked by unit), a numeric matrix (N x T), or a dcce_fit object (uses residuals).
data	A data.frame containing the panel structure. Required if x is a vector.
unit_index	Character: name of the unit variable in data.
time_index	Character: name of the time variable in data.
use_residuals	Logical: if TRUE, use the BKP (2019) residual method; if FALSE (default), use the BKP (2016) variable method.
n_pca	Integer: number of principal components. Default 1.
test_size	Numeric: significance level for thresholding. Default 0.1.
tuning	Numeric: tuning parameter for residual method threshold. Default 0.5.
n_bootstrap	Integer: bootstrap repetitions for SE (residual method only). Default 200.

Value

An object of class dcce_csd with elements alpha (estimated exponent), se (standard error), ci (confidence interval), method, N, and T_val.

Examples

```
set.seed(42)
# Matrix of cross-sectionally dependent data
N <- 20; T_val <- 50
f <- rnorm(T_val)
x <- matrix(NA, N, T_val)
for (i in 1:N) x[i,] <- rnorm(1) * f + rnorm(T_val, sd = 0.5)
result <- csd_exp(x, use_residuals = FALSE)
print(result)
```

D *Difference operator for dcce formulas*

Description

Creates a differenced version of a variable for use inside `dcce()` formulas.

Usage

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

Arguments

`x`	A numeric vector.
`k`	Integer difference order. Default 1.

Value

A numeric vector of the same length as `x` with leading `NA`s.

Examples

```
x <- c(10, 20, 30, 40, 50)
D(x, 1)   # NA 10 10 10 10
D(x, 2)   # NA NA 20 20 20
```

dcce *Dynamic Common Correlated Effects Estimation*

Description

Estimates heterogeneous coefficient panel data models with cross-sectional dependence using Mean Group (MG), Common Correlated Effects (CCE), Dynamic CCE (DCCE), and related estimators.

Usage

```
dcce(
  data,
  unit_index,
  time_index,
  formula,
 model = c("dcce", "cce", "ccep", "mg", "amg", "rcce", "ife", "pmg", "csdl", "csardl"),
  cross_section_vars = ~.,
  cross_section_lags = 0L,
  pooled_vars = NULL,
  include_constant = TRUE,
  unit_trend = FALSE,
  bias_correction = c("none", "jackknife", "recursive", "half_panel_jackknife"),
  long_run_vars = NULL,
  long_run_model = NULL,
  csdl_xlags = 3L,
  absorb = NULL,
  spatial_weights = NULL,
  fast = TRUE,
  n_cores = 1L,
  run_cd_test = FALSE,
  full_sample = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

data	A data.frame containing the panel data.
unit_index	Character: name of the unit (cross-section) variable.
time_index	Character: name of the time variable.
formula	A formula of the form y ~ x1 + x2. Supports L(), D(), and Lrange() operators for lags, differences, and lag ranges.
model	Character: estimator to use. One of "dcce", "cce", "ccep", "mg", "amg", "rcce", "ife", "pmg", "csdl", "csardl". Default "dcce". "ccep" is the Pooled CCE of Pesaran (2006) which constrains slopes to be identical across units. "ife" is the iterative PC estimator of Bai (2009).

`cross_section_vars`
: A one-sided formula specifying variables for cross-sectional averages, e.g. `~ x1 + x2`. Use `~ .` for all RHS variables plus the dependent variable. Use `NULL` for no CSAs (plain MG).

`cross_section_lags`
: Integer or named integer vector: number of lags of CSAs. Default 0. A single integer applies to all CSA variables.

`pooled_vars`
: A one-sided formula specifying which coefficients to constrain equal across units (pooled estimation). Default `NULL` (all heterogeneous).

`include_constant`
: Logical: include unit-specific intercepts? Default `TRUE`.

`unit_trend`
: Logical: include unit-specific linear trends? Default `FALSE`.

`bias_correction`
: Character: bias correction method. One of `"none"`, `"jackknife"`, `"recursive"`, `"half_panel_jackknife"`. The `"half_panel_jackknife"` implements the Chudik & Pesaran (2015) time-series half-panel jackknife that corrects the Nickell bias in dynamic CCE. Default `"none"`.

`long_run_vars`
: A one-sided formula specifying long-run variables (reserved for CSDL/CSARDL models). Default `NULL`.

`long_run_model`
: Character: long-run model specification (reserved). Default `NULL`.

`csdl_xlags`
: Integer: number of lags of Δx to include as short-run controls when `model = "csdl"`. Default 3.

`absorb`
: Either `NULL` (default), a character vector of column names, or a one-sided formula like `~ industry + region` specifying high-dimensional fixed effects to project out of y and X before the main unit loop runs. A single factor uses the within transformation; multiple factors use the alternating projections of Guimaraes & Portugal (2010) / Correia (2016). The unit fixed effects used by CCE estimators are still kept via unit intercepts; `absorb` is for *additional* categorical effects on top of the cross-section.

`spatial_weights`
: Optional $N \times N$ numeric matrix of spatial weights. When supplied, the global cross-sectional averages of classical CCE are replaced with **local**, unit-specific weighted averages $\bar{y}_{i,t}^{W} = \sum_j w_{ij} y_{j,t}$. The matrix must be square with rows and columns matching the unit identifiers (row/column names are used for alignment if present); it is row-normalised automatically and the diagonal is zeroed. This enables spatial CCE estimation that respects the topology of cross-sectional dependence (geographical contiguity, trade links, etc.).

`fast`
: Logical: use the compiled C++ (RcppArmadillo) unit-OLS fast path? Default `TRUE`. Falls back to pure R automatically if the compiled routines are not available.

`n_cores`
: Integer: number of cores for parallel unit estimation. Only effective on Unix/macOS via `parallel::mclapply`; on Windows the argument is silently ignored. Default `1L` (sequential).

`run_cd_test`
: Logical: run the Pesaran CD test on residuals? Default `FALSE`.

`full_sample`
: Logical: use the full (unbalanced) sample? Default `FALSE`.

verbose	Logical: print progress messages? Default FALSE.
...	Additional arguments passed to model-specific estimators.

Value

An object of class dcce_fit (and a model-specific subclass).

Examples

```
# Simple Mean Group estimation
df <- data.frame(
  id = rep(1:10, each = 20),
  t  = rep(1:20, 10),
  y  = rnorm(200),
  x  = rnorm(200)
)
fit <- dcce(df, unit_index = "id", time_index = "t",
            formula = y ~ x, model = "mg", cross_section_vars = NULL)
coef(fit)
```

---

dcce_bootstrap *Bootstrap alias that avoids the broom conflict*

Description

Convenience alias for bootstrap with an unambiguous name. broom::bootstrap is a data-frame resampling helper that shares a name with dcce::bootstrap but has a completely different signature. If you load broom after dcce the broom function masks ours on the search path and calls to bootstrap(fit, type = ..., reps = ...) will fail with an "unused arguments" error. dcce_bootstrap() is identical to dcce::bootstrap() but cannot be masked by any other package.

Usage

```
dcce_bootstrap(
  object,
  type = c("crosssection", "wild"),
  reps = 500L,
  percentile = TRUE,
  cfresiduals = FALSE,
  seed = NULL
)
```

Arguments

object	A dcce_fit object.
type	Character: "crosssection" (default) or "wild".
reps	Integer: number of bootstrap repetitions. Default 500.

percentile	Logical: compute percentile CIs? Default TRUE.
cfresiduals	Logical: for wild bootstrap, use common-factor residuals instead of defactored residuals? Default FALSE.
seed	Integer: random seed for reproducibility. Default NULL.

Value

See bootstrap.

See Also

bootstrap

Examples

```
set.seed(42)
df <- data.frame(
  id = rep(1:10, each = 30),
  t  = rep(1:30, 10),
  y  = rnorm(300),
  x  = rnorm(300)
)
fit <- dcce(df, "id", "t", y ~ x, model = "mg", cross_section_vars = NULL)
dcce_bootstrap(fit, reps = 50)
```

---

dcce_rolling *Rolling-Window Panel Estimation*

Description

Fits a sequence of dcce() models on overlapping time windows of the panel, producing a time path of coefficient estimates. Useful for detecting coefficient drift, parameter instability, or regime shifts in long panels.

Usage

```
dcce_rolling(
  data,
  unit_index,
  time_index,
  formula,
  model = "cce",
  window = 20L,
  step = 1L,
  min_units = 5L,
  verbose = FALSE,
  ...
)
```

Arguments

data	A panel data.frame.
unit_index	Character: unit identifier column.
time_index	Character: time identifier column.
formula	Two-sided model formula passed through to dcce().
model	Character: estimator to use (default "cce"). Same choices as dcce().
window	Integer: window length in time periods.
step	Integer: number of time periods to advance between windows. Default 1L.
min_units	Integer: minimum number of cross-sectional units a window must retain after NA handling to be kept. Default 5L.
verbose	Logical: print progress? Default FALSE.
...	Additional arguments passed to dcce() (e.g. cross_section_vars, cross_section_lags, fast).

Details

At each window the function subsets the panel to observations whose time index falls in $[t_k, t_k + \text{window} - 1]$, runs dcce() with the user-supplied arguments, and collects the Mean Group coefficient vector and its standard errors. The result is returned as a dcce_rolling object containing the full list of fits and a tidy tibble of coefficients indexed by window end-date.

Value

An object of class dcce_rolling containing:

fits List of dcce_fit objects, one per window (or NULL for windows that failed).

coefficients A tibble with one row per (window end-date, term) combination, columns window_end, window_start, term, estimate, std.error, conf.low, conf.high.

window The window length.

step The step size.

n_windows Number of successful windows.

call The original call.

Examples

```
data(pwt8)
roll <- dcce_rolling(
  data       = pwt8,
  unit_index = "country",
  time_index = "year",
  formula    = d_log_rgdpo ~ log_hc + log_ck + log_ngd,
  model      = "cce",
  cross_section_vars = ~ .,
  window     = 20,
  step       = 2
)
print(roll)
```

`dcce_sim` *Simulated Dynamic Panel Dataset*

Description

A synthetic panel dataset generated from the dynamic common correlated effects DGP of Chudik and Pesaran (2015), equation (1). Generated with N = 30 cross-sectional units and T = 50 time periods. The true mean group parameters are: autoregressive coefficient 0.50, slope on x 1.00. Factor structure uses a single AR(1) common factor with persistence 0.60.

Usage

```
dcce_sim
```

Format

A data frame with 1500 rows and 4 columns:

unit Cross-sectional unit identifier (integer, 1–30)

time Time period (integer, 1–50)

y Simulated dependent variable

x Simulated regressor (cross-sectionally dependent via common factor)

Details

The true parameter values are stored in the companion object `dcce_sim_truth` and can be used in tests to verify estimator consistency.

References

Chudik, A. and Pesaran, M. H. (2015). Common correlated effects estimation of heterogeneous dynamic panel data models with weakly exogenous regressors. *Journal of Econometrics*, 188(2), 393–420.

See Also

dcce_sim_truth for the true parameter values.

dcce_sim_truth *True Parameters for the Simulated Panel Dataset*

Description

A named list containing the true mean group parameter values used to generate dcce_sim. Use in tests to verify that `dcce()` recovers parameters close to their true values.

Usage

```
dcce_sim_truth
```

Format

A named list with elements:

beta1_mg True mean group autoregressive coefficient (~0.50)

beta2_mg True mean group slope on x (~1.00)

N Number of cross-sectional units (30)

T Number of time periods (50)

seed Random seed used for generation (20240101)

dcce_workflow *Automatic Diagnostic Workflow for Panel Data with CSD*

Description

Runs the recommended pre-estimation diagnostic sequence on a panel regression specification and returns a structured report with a suggested `dcce()` call. The workflow performs six steps:

Usage

```
dcce_workflow(
  data,
  unit_index,
  time_index,
  formula,
  max_cr_lags = NULL,
  significance = 0.05,
  verbose = TRUE,
  n_bootstrap = 0L
)
```

Arguments

data	A panel data.frame.
unit_index	Character: unit identifier column.
time_index	Character: time identifier column.
formula	Two-sided formula (levels, not differences).
max_cr_lags	Integer: maximum CSA lag order to evaluate. Default NULL uses $\lfloor T^{1/3} \rfloor$.
significance	Numeric: significance level used for decisions. Default 0.05.
verbose	Logical: print progress as each step runs. Default TRUE.
n_bootstrap	Integer: bootstrap replications for the Westerlund p-values. Default 0L (asymptotic).

Details

1. Panel summary (N, T, balance).
2. Pesaran (2007) CIPS panel unit root test on each variable.
3. Pesaran CD test on raw residuals (pooled OLS) to check for cross-sectional dependence.
4. Westerlund (2007) cointegration test, if at least one variable is non-stationary.
5. Rank condition classifier (De Vos et al. 2024) for a baseline static CCE fit.
6. Information criterion selection of the optimal CSA lag order.

Based on the results, the function chooses a recommended estimator from c("mg", "cce", "dcce", "pmg", "csardl") and returns a printable suggested dcce() call.

Value

An object of class dcce_workflow with elements panel_summary, unit_root, csd_premodel, cointegration, rank_condition, optimal_cr_lags, recommendation, and call.

Examples

```
data(pwt8)
wf <- dcce_workflow(
  data       = pwt8,
  unit_index = "country",
  time_index = "year",
  formula    = log_rgdpo ~ log_hc + log_ck + log_ngd,
  verbose    = FALSE
)
print(wf)
```

fitted.dcce_fit *Extract fitted values from a dcce_fit object*

Description

Extract fitted values from a dcce_fit object

Usage

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

Arguments

object	A dcce_fit object.
...	Ignored.

Value

A numeric vector of fitted values (y-hat), or NULL if not available.

glance.dcce_fit *Glance at a dcce_fit object*

Description

Returns a single-row tibble of model summary statistics, compatible with the broom package.

Usage

```
## S3 method for class 'dcce_fit'
glance(x, ...)
```

Arguments

x	A dcce_fit object.
...	Ignored.

Value

A single-row tibble.

`granger_test` *Dumitrescu-Hurlin Panel Granger Causality Test*

Description

Tests whether x Granger-causes y in a heterogeneous panel, following Dumitrescu & Hurlin (2012). For each unit i a bivariate VAR-type regression is fitted:

$$y_{it} = \alpha_i + \sum_{k=1}^{K} \gamma_{ik} y_{i,t-k} + \sum_{k=1}^{K} \beta_{ik} x_{i,t-k} + u_{it},$$

and the individual Wald statistic for $H_0^{(i)} : \beta_{i1} = \cdots = \beta_{iK} = 0$ is computed.

Usage

```
granger_test(data, unit_index, time_index, y, x, lags = 1L)
```

Arguments

`data`	A panel data.frame.
`unit_index`	Character: unit identifier column.
`time_index`	Character: time identifier column.
`y`	Character: name of the dependent variable.
`x`	Character: name of the potential Granger-causing variable.
`lags`	Integer: lag order K for both variables. Default 1.

Details

The panel statistics are:

W-bar The cross-sectional average of the N unit-level Wald statistics.

Z-bar Standardised version: $\tilde{Z} = \sqrt{N/(2K)}(W\text{-bar} - K) \xrightarrow{d} \mathcal{N}(0,1)$.

Z-bar tilde Small-sample adjusted version using $E[W_i]$ and $Var(W_i)$ from the $F(K, T-3K-1)$ distribution.

Value

An object of class `dcce_granger` with elements:

W_bar Mean Wald statistic across units.

Z_bar Standardised statistic (large-T).

Z_bar_tilde Small-sample adjusted statistic.

p_value_Z p-value for Z-bar.

p_value_Zt p-value for Z-bar tilde.

unit_wald Named vector of per-unit Wald statistics.

N Number of units used.

T_bar Average time-series length.

lags Lag order.

References

Dumitrescu, E.-I., & Hurlin, C. (2012). Testing for Granger non-causality in heterogeneous panels. *Economic Modelling*, 29(4), 1450–1460.

Examples

```
data(pwt8)
gc <- granger_test(
  data = pwt8, unit_index = "country", time_index = "year",
  y = "d_log_rgdpo", x = "log_ck", lags = 1
)
print(gc)
```

---

hausman_test *Hausman-type Test: MG vs Pooled*

Description

Tests the null that the pooled and MG estimators are both consistent (i.e. slopes are homogeneous and the pooled estimator is efficient) against the alternative that slopes are heterogeneous and only MG is consistent. The test statistic is

$$H = (\hat{\beta}_{MG} - \hat{\beta}_{pool})'[V_{MG} - V_{pool}]^{-1}(\hat{\beta}_{MG} - \hat{\beta}_{pool}) \sim \chi^2_k.$$

Usage

```
hausman_test(object)
```

Arguments

`object`	A `dcce_fit` object.

Value

An object of class `dcce_hausman` with the test statistic, degrees of freedom, and p-value.

irf *Impulse Response Functions for Dynamic Panel Models*

Description

Computes impulse response functions (IRFs) from a fitted dynamic panel model (DCCE, CS-ARDL, or PMG). The IRF traces the response of the dependent variable to a one-unit shock in a specific regressor over a given number of horizons, using the Mean Group ARDL coefficient estimates and the autoregressive lag polynomial.

Usage

```
irf(object, impulse, horizon = 10L, boot_reps = 200L, seed = NULL)
```

Arguments

object	A dcce_fit object from a dynamic model (must contain at least one L(y, k) term).
impulse	Character: the regressor that receives the shock.
horizon	Integer: number of periods to trace. Default 10.
boot_reps	Integer: bootstrap replications for confidence bands. 0 = no bands. Default 200.
seed	Integer: random seed.

Details

Confidence bands are computed via the cross-section bootstrap: units are resampled with replacement, the MG coefficients are recomputed, and the IRF is re-traced. The 2.5 and 97.5 percent quantiles of the bootstrap distribution form the 95 percent band.

Value

An object of class dcce_irf containing the impulse response path ($irf), optional bootstrap lower/upper bands, and metadata (impulse variable name and horizon length).

Examples

```
data(dcce_sim)
fit <- dcce(
  data = dcce_sim, unit_index = "unit", time_index = "time",
  formula = y ~ L(y, 1) + x,
  model = "dcce", cross_section_vars = ~ ., cross_section_lags = 3
)
ir <- irf(fit, impulse = "x", horizon = 10, boot_reps = 0)
print(ir)
```

L — *Lag operator for dcce formulas*

Description

Creates a lagged version of a variable for use inside `dcce()` formulas. This function is evaluated during formula processing by `dcce()` and should not be called directly on raw vectors outside of a dcce formula context.

Usage

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

Arguments

`x`	A numeric vector (column name evaluated within `dcce()`).
`k`	Integer lag order. Default 1. Positive values lag, negative lead.

Value

A numeric vector of the same length as `x` with leading `NA`s.

Examples

```
x <- c(10, 20, 30, 40, 50)
L(x, 1)   # NA 10 20 30 40
L(x, 2)   # NA NA 10 20 30
```

Lrange — *Lag range operator for dcce formulas*

Description

Creates multiple lagged versions of a variable from lag `k0` to lag `k1`. Useful for distributed lag specifications.

Usage

```
Lrange(x, k0, k1)
```

Arguments

`x`	A numeric vector.
`k0`	Integer start lag (inclusive).
`k1`	Integer end lag (inclusive).

Value

A matrix with columns `L[k0]` through `L[k1]`.

Examples

```
x <- c(10, 20, 30, 40, 50)
Lrange(x, 0, 2)   # 3 columns: lag 0, lag 1, lag 2
```

---

panel_coint_test *Pedroni and Kao Panel Cointegration Tests*

Description

Implements simplified versions of the Pedroni (1999, 2004) and Kao (1999) residual-based panel cointegration tests.

Usage

```
panel_coint_test(
  data,
  unit_index,
  time_index,
  formula,
  test = c("pedroni", "kao"),
  lags = 1L
)
```

Arguments

`data`	A panel data.frame.
`unit_index`	Character: unit identifier column.
`time_index`	Character: time identifier column.
`formula`	Two-sided formula in levels: `y ~ x1 + x2`.
`test`	Character: `"pedroni"` or `"kao"`.
`lags`	Integer: ADF lag order for the residual regression. Default 1.

Details

Pedroni: Fits unit-level OLS regressions of y on $x_1, \ldots, x_K$, extracts the residuals, runs an ADF regression on each unit's residual series, and averages the t-statistics. Reports the group-mean t-statistic (`group_t`) and the group-mean rho-statistic (`group_rho`).

Kao: Similar but pools the residuals rather than averaging unit-level statistics. Reports a single ADF t-statistic on the demeaned pooled residuals.

Both tests have null hypothesis **no cointegration**. A large negative statistic rejects the null.

Value

An object of class `dcce_cointegration_extra` with elements `test`, `statistics` (a tibble), `N`, `T_bar`, `lags`.

References

Pedroni, P. (1999). Critical values for cointegration tests in heterogeneous panels with multiple regressors. *Oxford Bulletin of Economics and Statistics*, 61(S1), 653–670.

Pedroni, P. (2004). Panel cointegration: asymptotic and finite sample properties of pooled time series tests with an application to the PPP hypothesis. *Econometric Theory*, 20(3), 597–625.

Kao, C. (1999). Spurious regression and residual-based tests for cointegration in panel data. *Journal of Econometrics*, 90(1), 1–44.

Examples

```
data(pwt8)
panel_coint_test(
  data = pwt8, unit_index = "country", time_index = "year",
  formula = log_rgdpo ~ log_hc + log_ck,
  test = "pedroni", lags = 1
)
```

---

panel_ur_test *IPS and LLC Panel Unit Root Tests*

Description

Implements the Im, Pesaran & Shin (2003) IPS t-bar test and the Levin, Lin & Chu (2002) LLC common-root test for panel unit roots. Neither test corrects for cross-sectional dependence — use `cips_test` for CSD-robust testing.

Usage

```
panel_ur_test(x, ...)

## S3 method for class 'matrix'
panel_ur_test(x, ..., test = c("ips", "llc"), lags = 0L, trend = FALSE)

## Default S3 method:
panel_ur_test(
  x,
  ...,
  data = NULL,
  unit_index = NULL,
  time_index = NULL,
  test = c("ips", "llc"),
  lags = 0L,
```

```
  trend = FALSE
)
```

Arguments

x	A numeric matrix (N x T), data.frame, or vector.
...	Additional arguments.
test	Character: "ips" or "llc".
lags	Integer: ADF lag order. Default 0.
trend	Logical: include a time trend. Default FALSE.
data, unit_index, time_index	Panel structure (when x is a vector).

Details

IPS: fits unit-level ADF regressions, averages the t-statistics, and standardises using tabulated moments from Im et al. (2003, Table 2).

LLC: fits unit-level ADF regressions, extracts the t-statistic from a pooled regression of adjusted Δy on adjusted y_{t-1}, standardised to N(0,1).

Value

An object of class dcce_unit_root with elements test, statistic, p_value, N, T, lags, trend.

References

Im, K. S., Pesaran, M. H., & Shin, Y. (2003). Testing for unit roots in heterogeneous panels. *Journal of Econometrics*, 115(1), 53–74.

Levin, A., Lin, C.-F., & Chu, C.-S. J. (2002). Unit root tests in panel data: asymptotic and finite-sample properties. *Journal of Econometrics*, 108(1), 1–24.

Examples

```
set.seed(1)
X <- matrix(cumsum(rnorm(20 * 30)), 20, 30)
panel_ur_test(X, test = "ips")
panel_ur_test(X, test = "llc")
```

pcd_test *Cross-Sectional Dependence Tests*

Description

Computes cross-sectional dependence test statistics for panel data residuals. Implements the Pesaran (2015) CD test, CDw (Juodis & Reese 2022), PEA (Fan et al. 2015), and CD* (Pesaran & Xie 2021).

Usage

```
pcd_test(x, ...)

## S3 method for class 'dcce_fit'
pcd_test(
  x,
  ...,
  test = c("pesaran", "cdw", "cdwplus", "pea", "cdstar"),
  n_reps = 500L,
  n_pca = 1L
)

## S3 method for class 'data.frame'
pcd_test(
  x,
  ...,
  unit_index = NULL,
  time_index = NULL,
  test = c("pesaran", "cdw", "pea", "cdstar"),
  n_reps = 500L,
  n_pca = 1L
)

## S3 method for class 'matrix'
pcd_test(
  x,
  ...,
  test = c("pesaran", "cdw", "pea", "cdstar"),
  n_reps = 500L,
  n_pca = 1L
)

## Default S3 method:
pcd_test(
  x,
  ...,
  data = NULL,
```

```
  unit_index = NULL,
  time_index = NULL,
  test = c("pesaran", "cdw", "cdwplus", "pea", "cdstar"),
  n_reps = 500L,
  n_pca = 1L
)
```

Arguments

x	Either a numeric vector of residuals (stacked by unit), a dcce_fit object, a numeric matrix (N x T) of residuals, or a data.frame containing the panel structure.
...	Arguments passed to methods.
test	Character vector specifying which tests to compute. One or more of "pesaran", "cdw", "cdwplus", "pea", "cdstar". Default uses all five. See Pesaran (2015), Juodis & Reese (2022), Baltagi, Feng & Kao (2012), Fan et al. (2015), Pesaran & Xie (2021).
n_reps	Integer: number of Rademacher draws for CDw. Default 500.
n_pca	Integer: number of principal components for CD* bias correction. Default 1.
unit_index	Character: name of the unit variable in data.
time_index	Character: name of the time variable in data.
data	A data.frame containing the panel structure. Required if x is a vector.

Value

An object of class dcce_cd containing:

statistics A data.frame with columns test, statistic, p_value.

N Number of cross-sectional units.

T_bar Average time dimension.

rho_ij Matrix of pairwise correlations (if retained).

Examples

```
set.seed(42)
df <- data.frame(
  id = rep(1:10, each = 20),
  t  = rep(1:20, 10),
  e  = rnorm(200)
)
cd <- pcd_test(df$e, data = df, unit_index = "id", time_index = "t",
               test = "pesaran")
print(cd)
```

plot.dcce_fit *Plot method for dcce_fit objects*

Description

Produces coefficient distribution plots (one histogram per regressor) showing the unit-level estimates and the MG mean.

Usage

```
## S3 method for class 'dcce_fit'
plot(x, which = c("coef", "resid"), ...)
```

Arguments

x	A dcce_fit object.
which	Character: "coef" (default) for coefficient histograms, or "resid" for a residuals-versus-time plot by unit.
...	Passed to the underlying graphics call.

Value

Invisibly returns x.

plot.dcce_irf *Plot a dcce_irf object*

Description

Plot a dcce_irf object

Usage

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

Arguments

x	A dcce_irf object.
...	Passed to plot().

Value

Invisibly returns x.

plot.dcce_rolling *Plot a dcce_rolling coefficient path*

Description

Produces one line per regressor showing the rolling-window coefficient against the window end-date, with a 95% confidence ribbon built from the unit-loop standard errors.

Usage

```
## S3 method for class 'dcce_rolling'
plot(x, terms = NULL, ...)
```

Arguments

x	A dcce_rolling object.
terms	Character vector of terms to plot. Default NULL plots all non-intercept terms.
...	Passed to the underlying graphics call.

Value

Invisibly returns x.

predict.dcce_fit *Predict from a dcce_fit object*

Description

Predict from a dcce_fit object

Usage

```
## S3 method for class 'dcce_fit'
predict(object, newdata = NULL, type = c("response", "xb"), ...)
```

Arguments

object	A dcce_fit object.
newdata	Optional data.frame with new observations. When supplied, predictions are computed using the Mean Group coefficients on the structural regressors (CSAs are not applied because they depend on the full cross-section). When NULL, in-sample unit-level fitted values are returned.
type	Character: "response" (default) returns predicted y; "xb" is an alias for response (kept for compatibility with **marginaleffects**).
...	Ignored.

Value

A numeric vector of predictions.

`print.dcce_boot` *Print method for dcce_boot objects*

Description

Print method for dcce_boot objects

Usage

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

Arguments

`x`	A `dcce_boot` object.
`...`	Ignored.

Value

Invisibly returns `x`.

`print.dcce_break` *Print a dcce_break object*

Description

Print a dcce_break object

Usage

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

Arguments

`x`	A `dcce_break` object.
`...`	Ignored.

Value

Invisibly returns `x`.

print.dcce_cd *Print method for dcce_cd objects*

Description

Print method for dcce_cd objects

Usage

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

Arguments

x	A dcce_cd object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_cips *Print a dcce_cips object*

Description

Print a dcce_cips object

Usage

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

Arguments

x	A dcce_cips object.
...	Ignored.

Value

Invisibly returns x.

`print.dcce_cointegration` *Print method for dcce_cointegration*

Description

Print method for dcce_cointegration

Usage

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

Arguments

`x`	A `dcce_cointegration` object.
`...`	Ignored.

Value

Invisibly returns `x`.

`print.dcce_cointegration_extra` *Print a dcce_cointegration_extra object*

Description

Print a dcce_cointegration_extra object

Usage

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

Arguments

`x`	A `dcce_cointegration_extra` object.
`...`	Ignored.

Value

Invisibly returns `x`.

print.dcce_csd *Print method for dcce_csd objects*

Description

Print method for dcce_csd objects

Usage

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

Arguments

x	A dcce_csd object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_fit *Print a dcce_fit object*

Description

Print a dcce_fit object

Usage

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

Arguments

x	A dcce_fit object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_granger *Print a dcce_granger object*

Description

Print a dcce_granger object

Usage

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

Arguments

x	A dcce_granger object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_hausman *Print a dcce_hausman object*

Description

Print a dcce_hausman object

Usage

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

Arguments

x	A dcce_hausman object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_irf *Print a dcce_irf object*

Description

Print a dcce_irf object

Usage

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

Arguments

x	A dcce_irf object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_rolling *Print a dcce_rolling object*

Description

Print a dcce_rolling object

Usage

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

Arguments

x	A dcce_rolling object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_swamy *Print a dcce_swamy object*

Description

Print a dcce_swamy object

Usage

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

Arguments

x	A dcce_swamy object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_unit_root *Print a dcce_unit_root object*

Description

Print a dcce_unit_root object

Usage

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

Arguments

x	A dcce_unit_root object.
...	Ignored.

Value

Invisibly returns x.

print.dcce_workflow *Print a dcce_workflow object*

Description

Print a dcce_workflow object

Usage

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

Arguments

x	A dcce_workflow object.
...	Ignored.

Value

Invisibly returns x.

pwt8 *Penn World Tables Growth Panel Dataset*

Description

Panel dataset from Jan Ditzen's xtdcce2 Stata package, originally derived from the Penn World Tables 8. Used in Ditzen (2018, Stata Journal 18:3, 585–617) to illustrate MG, CCE, and DCCE estimation. Contains 93 countries observed from 1960 to 2007 (balanced).

Usage

```
pwt8
```

Format

A data frame with 4464 rows and 8 columns:

id Country numeric identifier

year Year (1960–2007)

log_rgdpo Log real GDP (output-side)

log_hc Log human capital index

log_ck Log physical capital stock

log_ngd Log of population growth plus depreciation

country Country identifier (character)

d_log_rgdpo First difference of log_rgdpo (GDP growth)

Source

Downloaded from https://github.com/JanDitzen/xtdcce2.

References

Ditzen, J. (2018). Estimating dynamic common-correlated effects in Stata. *The Stata Journal*, 18(3), 585–617.

`rank_condition` *Rank Condition Classifier*

Description

Implements the De Vos et al. (2024) rank condition classifier for CCE estimators. Tests whether the rank condition holds (RC=1) which ensures CCE consistency.

Usage

```
rank_condition(object, criterion = c("er", "gr"))
```

Arguments

`object`	A `dcce_fit` object.
`criterion`	Character: `"er"` or `"gr"` for factor number selection. Default `"er"`.

Details

Note: Only valid for static panels. Emits a warning if called on dynamic or PMG models.

Value

An object of class `dcce_rank` with elements `m` (number of common factors), `g` (rank of average factor loadings), `RC` (1 if rank condition holds, 0 otherwise).

residuals.dcce_fit *Extract residuals from a dcce_fit object*

Description

Extract residuals from a dcce_fit object

Usage

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

Arguments

object	A dcce_fit object.
...	Ignored.

Value

A numeric vector.

structural_break_test *Structural Break Tests for Panels with Cross-Sectional Dependence*

Description

Tests for and estimates structural breaks in heterogeneous panel data models with cross-sectional dependence. Implements a Wald-type Chow test at a known break date, a sup-Wald test for an unknown break date (with trimmed candidate set), and a sequential procedure for multiple breaks following Bai & Perron (1998) adapted to panel CCE settings.

Usage

```
structural_break_test(
  data,
  unit_index,
  time_index,
  formula,
  model = "cce",
  type = c("unknown", "known"),
  break_date = NULL,
  trim = 0.15,
  n_breaks = 1L,
  test_terms = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

data	A panel data.frame.
unit_index	Character: unit identifier column.
time_index	Character: time identifier column.
formula	Two-sided formula passed through to dcce().
model	Character: base estimator (default "cce").
type	Character: "known" for a Chow test at a specific break date, "unknown" for a sup-Wald test over a trimmed window. Default "unknown".
break_date	Required when type = "known". Must match the time-index type.
trim	Numeric: trimming fraction at each end of the sample for the unknown-break-date search. Default 0.15 (standard Andrews 1993 choice).
n_breaks	Integer: maximum number of breaks to search for using the sequential Bai-Perron procedure. Default 1L.
test_terms	Character vector: which coefficients to include in the Wald statistic. Default NULL tests all slope coefficients (excluding the intercept).
verbose	Logical: print progress during the candidate search. Default FALSE.
...	Additional arguments passed to dcce() (e.g. cross_section_vars, cross_section_lags).

Details

For each candidate break date τ the function computes

$$W_N(\tau) = (\hat{\beta}_1(\tau) - \hat{\beta}_2(\tau))'[V_1(\tau) + V_2(\tau)]^{-1}(\hat{\beta}_1(\tau) - \hat{\beta}_2(\tau))$$

where $\hat{\beta}_1$ and $\hat{\beta}_2$ are the Mean Group coefficients from running the base estimator on the pre-break and post-break sub-samples respectively, and V_1, V_2 are the corresponding MG variances. Under the null of no break the statistic is distributed χ^2_k where k is the number of tested coefficients.

The unknown-break-date sup-Wald test reports $\sup_{\tau \in [\tau_1, \tau_2]} W_N(\tau)$, where $[\tau_1, \tau_2]$ is the interior of the sample after applying the trimming parameter. Approximate p-values use the Andrews (1993) large-sample distribution.

Value

An object of class dcce_break with elements:

type "known" or "unknown".

statistic The Wald (or sup-Wald) test statistic.

p_value Approximate p-value.

df Degrees of freedom of the Wald statistic.

break_date Known or estimated break date.

candidates For type = "unknown": a tibble with one row per candidate date (date, wald).

break_dates Vector of break dates when n_breaks > 1L.

fit_pre dcce_fit object from the pre-break sub-sample.

fit_post dcce_fit object from the post-break sub-sample.

call The original call.

References

Andrews, D. W. K. (1993). Tests for parameter instability and structural change with unknown change point. *Econometrica*, 61(4), 821-856.

Bai, J., & Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica*, 66(1), 47-78.

Ditzen, J., Karavias, Y., & Westerlund, J. (2024). Multiple structural breaks in interactive effects panel data models. *Journal of Applied Econometrics*.

Examples

```
data(pwt8)
brk <- structural_break_test(
  data       = pwt8,
  unit_index = "country",
  time_index = "year",
  formula    = d_log_rgdpo ~ log_hc + log_ck + log_ngd,
  model      = "mg",
  cross_section_vars = NULL,
  type       = "unknown",
  trim       = 0.20
)
print(brk)
```

`summary.dcce_fit` *Summary for a dcce_fit object*

Description

Summary for a dcce_fit object

Usage

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

Arguments

`object`	A `dcce_fit` object.
`...`	Ignored.

Value

Invisibly returns `object`.

`swamy_test` *Swamy Slope Heterogeneity Test*

Description

Tests the null hypothesis that all slope coefficients are identical across cross-sectional units, against the alternative of heterogeneous slopes. Implements the Swamy (1970) chi-square test and the Pesaran & Yamagata (2008) standardised dispersion statistic.

Usage

```
swamy_test(object)
```

Arguments

`object`	A `dcce_fit` object (typically with `model = "mg"`, `"cce"`, or `"dcce"`).

Details

The Swamy statistic is

$$\tilde{S} = \sum_{i=1}^{N} (\hat{\beta}_i - \hat{\beta}^*)' \frac{X_i' M_\tau X_i}{\hat{\sigma}_i^2} (\hat{\beta}_i - \hat{\beta}^*),$$

where $\hat{\beta}_i$ is the unit-level OLS estimate, $\hat{\beta}^*$ is the weighted pooled estimate, and $M_\tau = I - \tau(\tau'\tau)^{-1}\tau'$ projects off the intercept. Under H_0 (homogeneous slopes), $\tilde{S}$ is asymptotically $\chi^2_{k(N-1)}$.

Pesaran & Yamagata (2008) propose a standardised version that is asymptotically standard normal:

$$\tilde{\Delta} = \sqrt{N}\, \frac{N^{-1}\tilde{S} - k}{\sqrt{2k}}.$$

Value

An object of class `dcce_swamy` with elements `S_stat`, `delta_stat`, `df`, `p_swamy`, `p_delta`, `N`, `k`.

References

Swamy, P. A. V. B. (1970). Efficient inference in a random coefficient regression model. *Econometrica*, 38(2), 311-323.

Pesaran, M. H., & Yamagata, T. (2008). Testing slope homogeneity in large panels. *Journal of Econometrics*, 142(1), 50-93.

tidy.dcce_fit *Tidy a dcce_fit object*

Description

Returns a tibble of MG coefficients with standard errors, test statistics, p-values, and confidence intervals, compatible with the broom package. For long-run estimators (CS-ARDL, CS-DL, PMG) the tibble additionally includes rows for long-run coefficients and the adjustment speed.

Usage

```
## S3 method for class 'dcce_fit'
tidy(x, include_lr = TRUE, ...)
```

Arguments

x	A dcce_fit object.
include_lr	Logical: include long-run and adjustment rows when available? Default TRUE.
...	Ignored.

Value

A tibble with columns term, estimate, std.error, statistic, p.value, conf.low, conf.high, type.

update.dcce_fit *Update a dcce_fit object*

Description

Update a dcce_fit object

Usage

```
## S3 method for class 'dcce_fit'
update(object, formula. = NULL, ..., evaluate = TRUE)
```

Arguments

object	A dcce_fit object.
formula.	Optional replacement formula (use . to keep existing parts).
...	Additional arguments passed to dcce() to override the original call.
evaluate	Logical: evaluate the updated call? Default TRUE.

Value

An updated dcce_fit object (if evaluate = TRUE) or an unevaluated call.

`vcov.dcce_fit` *Extract variance-covariance matrix from a dcce_fit object*

Description

Extract variance-covariance matrix from a dcce_fit object

Usage

```
## S3 method for class 'dcce_fit'
vcov(object, ...)
```

Arguments

`object`	A `dcce_fit` object.
`...`	Ignored.

Value

A variance-covariance matrix.

Index

∗ **datasets**
  dcce_sim, 17
  dcce_sim_truth, 18
  pwt8, 39

bootstrap, 3, *14, 15*

cips_test, 5, *26*
coef.dcce_fit, 7
cointegration_test, 7
confint.dcce_fit, 9
csd_exp, 10

D, 11
dcce, 12
dcce(), *18*
dcce_bootstrap, 14
dcce_rolling, 15
dcce_sim, 17, *18*
dcce_sim_truth, *17*, 18
dcce_workflow, 18

fitted.dcce_fit, 20

glance.dcce_fit, 20
granger_test, 21

hausman_test, 22

irf, 23

L, 24
Lrange, 24

panel_coint_test, 25
panel_ur_test, 26
pcd_test, 28
plot.dcce_fit, 30
plot.dcce_irf, 30
plot.dcce_rolling, 31
predict.dcce_fit, 31
print.dcce_boot, 32
print.dcce_break, 32
print.dcce_cd, 33
print.dcce_cips, 33
print.dcce_cointegration, 34
print.dcce_cointegration_extra, 34
print.dcce_csd, 35
print.dcce_fit, 35
print.dcce_granger, 36
print.dcce_hausman, 36
print.dcce_irf, 37
print.dcce_rolling, 37
print.dcce_swamy, 38
print.dcce_unit_root, 38
print.dcce_workflow, 39
pwt8, 39

rank_condition, 40
residuals.dcce_fit, 41

structural_break_test, 41
summary.dcce_fit, 43
swamy_test, 44

tidy.dcce_fit, 45

update.dcce_fit, 45

vcov.dcce_fit, 46