Package 'mditools'

August 20, 2026

Type Package

Title Microdata Infrastructure Tools for Firm-Level Microdata Research

Version 0.1.0

License GPL-3

URL https://github.com/Secretariat-CompNet/mditools

BugReports https://github.com/Secretariat-CompNet/mditools/issues

Description Supports the full analysis pipeline for researchers working with firm-level microdata. Provides data tools for panel preparation (import, outlier detection, classification harmonization), analytical methods (production function estimation, capital stock measurement, markups, intensity measures, distributions, regression, clustering), and disclosure tools for tagging outputs with dominance and observation counts before aggregation and publication.
Production function estimation implements methods by Ackerberg, Caves and Frazer (2015) <doi:10.3982/ECTA13408>, Levinsohn and Petrin (2003) <doi:10.1111/1467-937X.00246>, Wooldridge (2009) <doi:10.1016/j.econlet.2009.04.026>, Petrin, Poi and Levinsohn (2004) <doi:10.1177/1536867X0400400202>, and Arellano and Bond (1991) <doi:10.2307/2297968> with the ``too many instruments'' correction by Roodman (2009) <doi:10.1111/j.1468-0084.2008.00542.x>. Markup estimation follows De Loecker and Warzynski (2012) <doi:10.1257/aer.102.6.2437>. Cost-share production function estimation follows Basu and Fernald (1997) <doi:10.1086/262073>. Capital stock estimation via the Perpetual Inventory Method follows OECD (2009) <doi:10.1787/9789264068476-en>.

Encoding UTF-8

Imports data.table, fixest, haven, readxl, Matrix, cluster, dbscan, mclust, stats, utils, graphics, grDevices

Suggests testthat (>= 3.0.0), arrow

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation no

Author Daniele Aglio [aut],
Eric Bartelsman [aut],
Mirja Hälbig [aut],
Marco Miorandi [aut],
Johanna Weiss [aut, cre],
Alessandro Zona Mattioli [aut],
Julián Díaz-Acosta [ctb],
Alberto Ferreira [ctb],
Javier Miranda [ctb],
Marcelo Piemonte Ribeiro [ctb],
Reetuparna Vishwanath [ctb],
Chengzi Yi [ctb]

Maintainer Johanna Weiss <johanna.weiss@iwhesl.onmicrosoft.com>

Repository CRAN

Date/Publication 2026-08-20 14:50:02 UTC

Contents

`mditools-package` *mditools: Microdata Infrastructure Tools for Firm-Level Microdata Research*

Description

Supports the full analysis pipeline for researchers working with firm-level microdata. Provides data tools for panel preparation (import, outlier detection, classification harmonization), analytical methods (production function estimation, capital stock measurement, markups, intensity measures, distributions, regression, clustering), and disclosure tools for tagging outputs with dominance and observation counts before aggregation and publication. Production function estimation implements methods by Ackerberg, Caves and Frazer (2015) doi:10.3982/ECTA13408, Levinsohn and Petrin (2003) doi:10.1111/1467937X.00246, Wooldridge (2009) doi:10.1016/j.econlet.2009.04.026, Petrin, Poi and Levinsohn (2004) doi:10.1177/1536867X0400400202, and Arellano and Bond (1991) doi:10.2307/2297968 with the "too many instruments" correction by Roodman (2009) doi:10.1111/j.14680084.2008.00542.x. Markup estimation follows De Loecker and Warzynski (2012) doi:10.1257/aer.102.6.2437. Cost-share production function estimation follows Basu and Fernald (1997) doi:10.1086/262073. Capital stock estimation via the Perpetual Inventory Method follows OECD (2009) doi:10.1787/9789264068476-en.

Author(s)

Maintainer: Johanna Weiss <johanna.weiss@iwhesl.onmicrosoft.com>

Authors:

- Johanna Weiss <johanna.weiss@iwhesl.onmicrosoft.com>
- Daniele Aglio
- Eric Bartelsman
- Mirja H<U+00E4>lbig
- Marco Miorandi
- Alessandro Zona Mattioli

Other contributors:

- Juli<U+00E1>n D<U+00ED>az-Acosta [contributor]
- Alberto Ferreira [contributor]
- Javier Miranda [contributor]
- Marcelo Piemonte Ribeiro [contributor]
- Reetuparna Vishwanath [contributor]
- Chengzi Yi [contributor]

See Also

Useful links:

- https://github.com/Secretariat-CompNet/mditools
- Report bugs at https://github.com/Secretariat-CompNet/mditools/issues

`mdi_acf_prodest` *ACF Production Function Estimation*

Description

Estimates a production function using the Ackerberg-Caves-Frazer (ACF) method (Ackerberg, Caves & Frazer 2015, Econometrica). Runs a two-stage GMM procedure: a first-stage OLS (with optional polynomial and time fixed effects) to recover the productivity proxy Phi, followed by GMM minimization to identify input elasticities.

tfp = (Phi or y) - X*beta depending on `TFP_minuend`; with `TFP_minuend = "y"` this matches the y - X*beta convention of the other estimators.

Usage

```
mdi_acf_prodest(
  DT,
  y,
  endog,
  exog,
  instr,
  id,
  time,
  spec = "cd",
  degree = 3,
  lower_bound_theta = 0,
  upper_bound_theta = 1,
  TFP_demeaned = TRUE,
  TFP_minuend = c("Phi", "y"),
  Omega_estimates = TRUE,
  time_FE = FALSE,
  extended_instr = FALSE
)
```

Arguments

`DT`	A `data.table` (or coercible object) containing the panel data.
`y`	Character. Name of the output variable column.
`endog`	Character vector. Names of endogenous input columns (e.g. labour).
`exog`	Character vector. Names of exogenous input columns (e.g. capital).
`instr`	Character vector. Names of instrument columns for the first stage polynomial.
`id`	Character. Name of the firm/unit identifier column.
`time`	Character. Name of the time period column.
`spec`	Character. Functional form. Only `"cd"` (Cobb-Douglas) is implemented. Default `"cd"`.

`degree` Integer. Degree of the polynomial used in the first stage and the Omega law of motion. Default `3`.

`lower_bound_theta` Numeric. Lower bound for elasticity estimates in the GMM optimisation. Default `0`.

`upper_bound_theta` Numeric. Upper bound for elasticity estimates in the GMM optimisation. Default `1`.

`TFP_demeaned` Logical. If `TRUE`, TFP is demeaned by subtracting the period mean (using `mdi_aggregate`). Default `TRUE`.

`TFP_minuend` Character. Whether TFP is computed as residual from `"Phi"` (first-stage fitted values) or `"y"` (raw output). Default `"Phi"`.

`Omega_estimates` Logical. If `TRUE`, attaches the estimated law-of-motion parameters (`g_b_slopes`, `g_b_intercept`) to the output. Default `TRUE`.

`time_FE` Logical. If `TRUE`, period dummies are included in the first stage regression. Default `FALSE`.

`extended_instr` Logical. If `TRUE`, augments the second-stage instrument set from `{k_t, l_{t-1}}` to `{k_t, l_{t-1}, Phi_hat_{t-1}}` as in ACF (2015) eq (28). Produces overidentification by one moment and enables a Hansen J test. Useful as a robustness check when the default exactly-identified system hits the optimiser bounds. Default `FALSE`.

Value

A `data.table` with one row per observation in the GMM sample, containing: - `id`, `time` columns (using the names supplied) - `tfp`: total factor productivity - `el_<input>`: estimated input elasticity for each input - `TFP_demeaned` (if `TFP_demeaned = TRUE`): period-demeaned TFP - `g_b_slopes`, `g_b_intercept` (if `Omega_estimates = TRUE`) - `NumObs`: number of observations used in the GMM stage - `convergence`: optimiser convergence code (0 = converged)

Examples

```
library(data.table)
set.seed(1)
n <- 200
DT <- data.table(
  id   = rep(1:50, each = 4),
  year = rep(2000:2003, times = 50),
  y    = rnorm(n, 5, 1),
  l    = rnorm(n, 3, 0.5),
  k    = rnorm(n, 4, 0.5),
  m    = rnorm(n, 2, 0.5)
)
result <- mdi_acf_prodest(
  DT, y = "y", endog = "l", exog = "k", instr = "m",
  id = "id", time = "year", degree = 2, TFP_demeaned = FALSE
)
```

mdi_aggregate *Generic Aggregation Function*

Description

Aggregates variables from a sub-aggregate level to a higher aggregate level within a ‘data.table‘. Supports multiple aggregation types, including sum, standard deviation, mean, quantiles (25 missing values, number of non-missing values, number of empty strings, number of zeros, number of positive values, and the Herfindahl-Hirschman Index (HHI).

The function can optionally: - apply a weight column to variables before aggregation, - merge aggregated statistics back into the original dataset, - compute the number of unique firms in the input data, - and apply disclosure control criteria.

Usage

```
mdi_aggregate(
  DT,
  var_list,
  bygroups,
  agg_type = c("sum"),
  weight_col = NULL,
  mrg = FALSE,
  disclosure = TRUE,
  count_firms = FALSE,
  dom_formula = c("top_share", "residual"),
  minNumObs = 5L
)
```

Arguments

DT	A ‘data.table‘ containing the data to aggregate.
var_list	A character vector of variable names to aggregate.
bygroups	A character vector of grouping variables defining the aggregation level.
agg_type	A character vector specifying the type(s) of aggregation to perform. Supported types: ‘"sum"‘, ‘"sd"‘, ‘"mean"‘, ‘"q10"‘,‘"q25"‘, ‘"median"‘, ‘"q75"‘, ‘"q90"‘, ‘"count"‘, ‘"nmiss"‘, ‘"n_nonmiss"‘, ‘"nempty"‘, ‘"nzero"‘, ‘"npos"‘, and ‘"HHI"‘. Default is ‘"sum"‘.
weight_col	Optional character string naming a weight column in ‘DT‘ for weighted aggregates. Default is ‘NULL‘.
mrg	Logical. If ‘TRUE‘, aggregated statistics are merged back into the original dataset as new variables. If ‘FALSE‘, a new aggregated ‘data.table‘ is returned. Default is ‘FALSE‘.
disclosure	Logical. If ‘TRUE‘ and ‘mrg = FALSE‘, disclosure criteria are applied, adding dominance indicators and number of observations for disclosure control. Default is ‘TRUE‘.

count_firms	Logical. If 'TRUE', adds a column 'NumFirms' containing the number of unique firms in the input dataset. A firm identifier column ('plantid', 'firmid', 'entid', or 'entgrp') must be present. Default is 'FALSE'.
dom_formula	Character. Dominance formula passed to ['mdi_disclose_crit()'] when 'disclosure = TRUE'. '"top_share"' (default) computes the share of the top 'domNr' firms; '"residual"' computes $(Total - x_1 - x_2)/x_1$.
minNumObs	Integer. Minimum number of observations used for the quantile smoothing window in '"q10"', '"q25"', '"median"', '"q75"', '"q90"' aggregation types. Default is '5'.

Value

- If 'mrg = FALSE': An aggregated 'data.table' containing the requested statistics, optionally with disclosure variables and number of firms. - If 'mrg = TRUE': The input 'data.table' with new columns containing the aggregated statistics.

Examples

```
library(data.table)
DT <- data.table(
  firmid = rep(1:5, each = 2),
  year   = rep(2020:2021, 5),
  nace   = rep(c("A", "B"), 5),
  emp    = c(10, 12, 5, 6, 20, 22, 8, 9, 15, 16),
  rev    = c(100, 110, 50, 55, 200, 210, 80, 85, 150, 155)
)

# Sum by nace
mdi_aggregate(DT, "emp", "nace", "sum", disclosure = FALSE)

# Multiple agg types
mdi_aggregate(DT, "emp", "nace", c("sum", "mean"), disclosure = FALSE)

# Merge back into original DT
mdi_aggregate(DT, "emp", "nace", "sum", mrg = TRUE, disclosure = FALSE)

# Count unique firms
mdi_aggregate(DT, "emp", "nace", "sum",
  count_firms = TRUE, disclosure = FALSE)
```

---

mdi_clustering *Clustering*

Description

This tool clusters observations using one of the following methods: - k-means; - hierarchical clustering with Ward linkage; - hierarchical clustering with complete linkage; - hierarchical clustering

with average linkage; - hierarchical clustering with single linkage; - PAM; - Gaussian mixture; - DBSCAN. The tool can optionally compute: - total within-cluster sum of squares (WSS); - average silhouette width; - bootstrap-style ARI stability; - the WSS or silhouette selection plot used to choose the number of clusters.

Usage

```
mdi_clustering(
  DT,
  id_vars,
  cluster_vars,
 method = c("hc_ward", "hc_complete", "hc_average", "hc_single", "kmeans", "pam",
    "mclust", "dbscan"),
  k_selection = c("fixed", "automatic"),
  k_fixed = NULL,
  automatic_by_wss = FALSE,
  automatic_by_silhouette = FALSE,
  compute_wss = TRUE,
  compute_silhouette = TRUE,
  compute_stability = FALSE,
  plot_selection = FALSE,
  k_grid = 2:25,
  exclude_noise = TRUE,
  B_boot = 200,
  nstart = 100,
  seed = 123,
  minPts = 4,
  q = 0.95,
  eps = NULL,
  G = 1:10,
  standardize = TRUE,
  na_action = c("stop", "omit"),
  cluster_col = "cluster",
  overwrite_cluster_col = FALSE,
  bootstrap_reselect_parameters = FALSE,
  verbose = TRUE
)
```

Arguments

DT	A data.table or data.frame with observations to be clustered. Data frames are silently converted to data.table.
id_vars	Character vector with one or more id variables.
cluster_vars	Character vector with the numeric variables used for clustering.
method	Clustering method. One of ‘"kmeans"‘, ‘"hc_ward"‘, ‘"hc_complete"‘, ‘"hc_average"‘, ‘"hc_single"‘, ‘"pam"‘, ‘"mclust"‘, ‘"dbscan"‘.
k_selection	Selection method for k: ‘"fixed"‘ or ‘"automatic"‘.
k_fixed	Fixed number of clusters. Used only when ‘k_selection = "fixed"‘.

`automatic_by_wss`
: Logical. If ‘TRUE‘, automatic k selection is based on WSS elbow. Default ‘FALSE‘.

`automatic_by_silhouette`
: Logical. If ‘TRUE‘, automatic k selection is based on average silhouette. Default ‘FALSE‘.

`compute_wss`
: Logical. If ‘TRUE‘, computes final WSS. Default ‘TRUE‘.

`compute_silhouette`
: Logical. If ‘TRUE‘, computes final average silhouette. Default ‘TRUE‘.

`compute_stability`
: Logical. If ‘TRUE‘, performs bootstrap-style ARI stability analysis. Default ‘FALSE‘.

`plot_selection`
: Logical. If ‘TRUE‘, plots the WSS or silhouette curve used to select k. Default ‘FALSE‘.

`k_grid`
: Candidate values of k for automatic selection. Default ‘2:25‘.

`exclude_noise`
: Logical. Mainly relevant for DBSCAN; if ‘TRUE‘, observations labelled as noise (cluster 0) are excluded from WSS and silhouette calculations. Default ‘TRUE‘.

`B_boot`
: Number of bootstrap repetitions. Default ‘200‘.

`nstart`
: Number of random starts for k-means. Default ‘100‘.

`seed`
: Integer seed for reproducibility. Default ‘123‘.

`minPts`
: DBSCAN ‘minPts‘ parameter. Default ‘4‘.

`q`
: Quantile used to choose DBSCAN ‘eps‘ automatically. Default ‘0.95‘.

`eps`
: Numeric or ‘NULL‘. DBSCAN radius parameter. If a numeric value is provided, DBSCAN uses it directly. If ‘NULL‘, ‘eps‘ is chosen automatically from the ‘q‘ quantile of k-nearest-neighbour distances with ‘k = minPts‘.

`G`
: Candidate number of mixture components for mclust. Default ‘1:10‘.

`standardize`
: Logical. If ‘TRUE‘, clustering variables are standardised via ‘scale()‘ before clustering. Recommended for distance-based methods when variables are on different scales. Default ‘TRUE‘.

`na_action`
: Character. How to handle missing values in ‘cluster_vars‘. ‘"stop"‘ raises an error; ‘"omit"‘ removes incomplete rows before clustering.

`cluster_col`
: Character. Name of the output column for the cluster assignment. Default ‘"cluster"‘.

`overwrite_cluster_col`
: Logical. If ‘FALSE‘, stops when ‘cluster_col‘ already exists in ‘DT‘. If ‘TRUE‘, the existing column is replaced. Default ‘FALSE‘.

`bootstrap_reselect_parameters`
: Logical. Only used when ‘compute_stability = TRUE‘. If ‘FALSE‘, each bootstrap sample uses the same parameters as the final model. If ‘TRUE‘, automatic parameters are re-selected within each bootstrap sample. Default ‘FALSE‘.

`verbose`
: Logical. If ‘TRUE‘, prints progress messages. Default ‘TRUE‘.

Value

a list with: - data: the original input data.table with an additional clustering column, by default called "cluster". - chosen_k: final selected number of clusters, when relevant. - wss: final WSS result, if requested. - silhouette: final silhouette result, if requested. - stability: bootstrap-style ARI stability result, if requested. - selection_plot: recorded plot object, if plot_selection = TRUE.

Examples

```
library(data.table)
set.seed(1)
DT <- data.table(
  firmid = 1:30,
  x1     = c(rnorm(15, 0, 0.5), rnorm(15, 5, 0.5)),
  x2     = c(rnorm(15, 0, 0.5), rnorm(15, 5, 0.5))
)

result <- mdi_clustering(DT, id_vars = "firmid",
                     cluster_vars = c("x1", "x2"),
                     method = "kmeans", k_selection = "fixed", k_fixed = 2,
                     compute_wss = TRUE, compute_silhouette = TRUE,
                     compute_stability = FALSE, verbose = FALSE)
```

---

mdi_cs_prodest *Cost-Shares Production Function Estimator (Cobb-Douglas)*

Description

Implements a cost-shares approach to production function estimation: - Build firm-time input shares from expenditures in levels. - Average shares by (bygroup, time). - Compute TFP as a log-index residual using averaged shares. - Optionally demean TFP by (bygroup, time).

Usage

```
mdi_cs_prodest(
  DT,
  y,
  endog,
  exog,
  id,
  time,
  bygroup,
  log_values = TRUE,
  TFP_demeaned = TRUE
)
```

Arguments

DT	A data.table or coercible data.frame containing panel data.
y	Character scalar. Name of the output variable column.
endog	Character scalar. Name of the free input (expenditure) column.
exog	Character scalar. Name of the state input (expenditure) column.
id	Character scalar. Name of the entity identifier column.
time	Character scalar. Name of the time period column.
bygroup	Character scalar. Name of the grouping variable column (e.g., industry code).
log_values	Logical. If TRUE (default), y, endog, and exog are treated as log values and exponentiated before share construction.
TFP_demeaned	Logical. If TRUE (default), returns a TFP_demeaned column equal to tfp minus its mean within (bygroup, time). This is mechanically zero by construction of the cost-shares index.

Value

A data.table with one row per observation (after cost-share filtering), containing: - id, time, bygroup columns - el_<endog>, el_<exog>: averaged input elasticities - tfp: total factor productivity (log-index residual) - TFP_demeaned (if TFP_demeaned = TRUE) - NumObs: total number of observations used (matches the convention of the other estimators in this suite)

Returns NULL if no valid observations remain after filtering.

Examples

```
library(data.table)
set.seed(1)
n <- 120
DT <- data.table(
  id      = rep(1:30, each = 4),
  year    = rep(2000:2003, times = 30),
  nace    = rep(c("A", "B"), each = 60),
  y       = log(runif(n, 5, 50)),
  labour  = log(runif(n, 1, 10)),
  capital = log(runif(n, 2, 20))
)
mdi_cs_prodest(DT, y = "y", endog = "labour", exog = "capital",
               id = "id", time = "year", bygroup = "nace")
```

`mdi_disclose_crit` *Add Disclosure Criteria to Aggregated Data*

Description

Computes and attaches disclosure-control variables to an aggregated dataset. The function calculates a dominance measure and the number of non-missing observations per group.

Two dominance formulas are available via ‘dom_formula‘: - ‘"top_share"‘ (default): share of the top ‘domNr‘ firms in the group total. - ‘"residual"‘: $(Total - x_1 - x_2)/x_1$, where x_1 and x_2 are the two largest values. Used when the dominance criterion is defined as the residual relative to the largest firm.

This function is normally called internally by [‘mdi_aggregate()‘] when ‘disclosure = TRUE‘, but can also be used standalone.

Usage

```
mdi_disclose_crit(
  DT,
  domVar = "var",
  domNr = 2,
  bygroups,
  var_list = NULL,
  dom_formula = c("top_share", "residual"),
  count_firms = FALSE,
  firm_col = "firmid",
  ent_col = "entid"
)
```

Arguments

`DT`	A ‘data.table‘ containing the aggregated dataset.
`domVar`	Character. Variable used for the dominance criterion. Use ‘"var"‘ (default) to compute dominance for all variables in ‘var_list‘, or supply the name of a single numeric column already present in ‘DT‘ (e.g. ‘"emp"‘, ‘"nq"‘).
`domNr`	Numeric. Number of top firms to consider in the dominance criterion (e.g. top 1, 2, or 3). Default ‘2‘.
`bygroups`	Character vector of grouping variables, as in [‘mdi_aggregate()‘].
`var_list`	Character vector of variables to include when ‘domVar = "var"‘. Usually the same as in [‘mdi_aggregate()‘]. Default ‘NULL‘.
`dom_formula`	Character. Formula used to compute the dominance share. ‘"top_share"‘ (default) computes the share of the top ‘domNr‘ firms in the group total. ‘"residual"‘ computes $(Total - x_1 - x_2)/x_1$. Only applies when ‘domVar = "var"‘.
`count_firms`	Logical. If ‘TRUE‘, the number of unique firms and enterprises per group are computed and added to the output as ‘NumFirms‘ and ‘NumEnt‘, using ‘firm_col‘ and ‘ent_col‘. Default ‘FALSE‘.

firm_col	Character. Column name used to count unique firms when ‘count_firms = TRUE‘. Default ‘"firmid"‘.
ent_col	Character. Column name used to count unique enterprises when ‘count_firms = TRUE‘. Default ‘"entid"‘.

Details

- For ‘domVar != "var"‘, the named column must already be present in ‘DT‘. One dominance column (‘domPerc‘) is returned and ‘dom_formula‘ is ignored. - For ‘domVar = "var"‘, separate dominance columns are created for each variable in ‘var_list‘ (‘domPerc_<var>‘), using the formula specified by ‘dom_formula‘. - When ‘count_firms = TRUE‘, ‘firm_col‘ and ‘ent_col‘ must be present in ‘DT‘; the function stops with an error if either is missing.

Value

A ‘data.table‘ with the same grouping structure as the input, plus: - One or more ‘domPerc_*‘ columns: dominance share per group. - A column ‘NumObs‘: number of non-missing observations per group. - ‘NumFirms‘ and ‘NumEnt‘ (only when ‘count_firms = TRUE‘).

Examples

```
library(data.table)
DT <- data.table(
  nace   = rep(c("A", "B"), each = 5),
  year   = rep(2020L, 10),
  emp    = c(10, 20, 5, 15, 8, 12, 25, 6, 14, 9),
  firmid = 1:10,
  entid  = c(1,1,2,2,3,4,4,5,5,6)
)

# Standard top-share formula
mdi_disclose_crit(DT, domVar = "var", domNr = 2L,
            bygroups = c("nace", "year"), var_list = "emp")

# Residual formula with firm counts
mdi_disclose_crit(DT, domVar = "var", domNr = 2L,
            bygroups = c("nace", "year"), var_list = "emp",
            dom_formula = "residual", count_firms = TRUE)

# Single-column dominance (column must be present in DT)
mdi_disclose_crit(DT, domVar = "emp", domNr = 2L,
            bygroups = c("nace", "year"))
```

---

mdi_disclose_reg_tab *Disclosure Check and Masking for Regression Tables*

Description

Applies disclosure control rules to regression output tables. Rows that do not meet minimum thresholds for degrees of freedom, number of observations, or (for Germany) number of firms are flagged and sensitive regression statistics are masked.

Usage

```
mdi_disclose_reg_tab(
  DT,
  min_obs = 5,
  show_disclosed = FALSE,
  disc_method = c("obs_df", "firm_count")
)
```

Arguments

DT	A ‘data.table‘ containing regression output. Must include the columns ‘"coef"‘ and—depending on ‘disc_method‘—either ‘"df"‘ and ‘"NumObs"‘ (for ‘"obs_df"‘) or ‘"NumFirms"‘ and ‘"NumEnt"‘ (for ‘"firm_count"‘).
min_obs	Numeric. Minimum threshold used for disclosure checks. Default ‘5‘.
show_disclosed	Logical. If ‘TRUE‘, disclosed values are shown even when flagged. If ‘FALSE‘ (default), disclosed values are masked with ‘NA‘.
disc_method	Character. Disclosure rule to apply. ‘"obs_df"‘ (default) flags rows where ‘df < min_obs‘ or ‘NumObs < min_obs‘. ‘"firm_count"‘ flags rows where ‘NumFirms < min_obs‘ or ‘NumEnt < min_obs‘ (used for Germany).

Details

- **‘disc_method = "obs_df"‘:** disclosure is triggered when either ‘df < min_obs‘ or ‘NumObs < min_obs‘. - **‘disc_method = "firm_count"‘:** disclosure is based on ‘NumFirms < min_obs‘ or ‘NumEnt < min_obs‘. Used for Germany. - The following regression statistics may be masked if disclosure applies: ‘"Estimate"‘, ‘"Std. Error"‘, ‘"z value"‘, ‘"Pr(>|z|)"‘, ‘"ci.lower"‘, ‘"ci.upper"‘, ‘"R2"‘, ‘"AdjR2"‘, ‘"AIC"‘, ‘"BIC"‘, ‘"LogLik"‘.

Value

A list with three elements:

DT	A ‘data.table‘ of the regression output with disclosure rules applied. Masked cells are set to ‘NA‘ (unless ‘show_disclosed = TRUE‘).
vars	A character string listing the disclosed coefficient names (for use in output description). If all rows are masked, returns ‘"No coefficients disclosed (all masked)"‘.
redacted_n	An integer count of the number of rows flagged and masked.

Examples

```
library(data.table)
DT <- data.table(
  coef         = c("(Intercept)", "x1"),
  Estimate     = c(1.2, 0.5),
  `Std. Error` = c(0.1, 0.05),
  df           = c(20L, 20L),
  NumObs       = c(25L, 25L)
)
result <- mdi_disclose_reg_tab(DT, min_obs = 3L)
result$DT
result$redacted_n
```

`mdi_dpgmm_prodest` *Arellano-Bond Difference-GMM Production Function Estimator*

Description

Estimates the dynamic Cobb-Douglas production function

$$y_{it} = \rho\, y_{i,t-1} + \beta_l\, l_{it} + \beta_k\, k_{it} + \alpha_i + \varepsilon_{it}$$

by first-differencing to eliminate the firm fixed effect α_i, then applying one-step GMM with lagged levels of (y, l, k) as instruments for the differenced regressors. Reference: Arellano & Bond (1991, ReStud).

Capital is treated as predetermined (chosen at t-1), so $k_{i,t-s}$ for s >= 1 are valid instruments for Δk_{it}. Labour is treated as endogenous (chosen at t with knowledge of ε_{it}), so $l_{i,t-s}$ for s >= 2 are valid. Lagged output $y_{i,t-s}$ for s >= 2 instruments $\Delta y_{i,t-1}$. The lag depth is capped by `max_lag_Z` to avoid the "too many instruments" problem (Roodman 2009).

Estimation is one-step GMM with weight matrix $W = (Z'HZ)^{-1}$, where H is block-diagonal by firm with tridiagonal blocks (2 on the diagonal, -1 on the first off-diagonals) reflecting the MA(1) structure of $\Delta\varepsilon_{it}$ under iid ε. Standard errors use the cluster-robust sandwich formula (clusters = firms).

tfp is reported in levels as `tfp = y - beta_l*l - beta_k*k`, matching the convention of the other estimators (ACF/LP/OLS/WDRG). It absorbs the firm fixed effect and the lagged-y persistence; it is NOT the productivity innovation ε_{it}.

Usage

```
mdi_dpgmm_prodest(
  DT,
  y,
  endog,
  exog,
  id,
  time,
```

```
  max_lag_Z = 2,
  TFP_demeaned = TRUE
)
```

Arguments

DT	A data.table (or coercible object) containing panel data.
y	Character scalar. Output variable name (in logs).
endog	Character scalar. Free input name (e.g. "ln_labor_cost").
exog	Character scalar. State input name (e.g. "ln_capital").
id	Character scalar. Firm identifier column.
time	Character scalar. Time identifier column.
max_lag_Z	Integer >= 1. Maximum lag depth for instruments. Default 2.
TFP_demeaned	Logical. If TRUE, returns TFP_demeaned = tfp - period mean. Default TRUE.

Value

A data.table with one row per firm-year (subset where y, endog, exog are non-missing). Columns: id, time, el_<endog>, el_<exog>, se_el_<endog>, se_el_<exog>, rho, se_rho, tfp, NumObs, the diagnostics sargan_J/sargan_df/sargan_pval and ar1_z/ar1_pval/ar2_z/ar2_pval, and optionally TFP_demeaned. Under iid ε: AR(1) should reject, AR(2) should not; Sargan tests the overidentifying restrictions.

Examples

```
library(data.table)
set.seed(1)
n_firms <- 40; n_periods <- 6
n <- n_firms * n_periods
DT <- data.table(
  id   = rep(seq_len(n_firms), each = n_periods),
  year = rep(seq(2000L, length.out = n_periods), times = n_firms),
  y    = rnorm(n, 5, 1),
  l    = rnorm(n, 3, 0.5),
  k    = rnorm(n, 4, 0.5)
)
result <- mdi_dpgmm_prodest(DT, y = "y", endog = "l", exog = "k",
                            id = "id", time = "year", TFP_demeaned = FALSE)
```

`mdi_estimate_markup` *Estimate Firm-Level Markup*

Description

Estimates firm-level markup following De Loecker (2012): markup is the output elasticity of an input divided by its expenditure share of revenue.

Usage

```
mdi_estimate_markup(DT, oe = "oe_l", rev_col = "nq", input_cost = "nm")
```

Arguments

`DT`	A ‘data.table‘ containing panel data.
`oe`	Character scalar. Name of the output elasticity column. Default ‘"oe_l"‘.
`rev_col`	Character scalar. Name of the total revenue column. Default ‘"nq"‘.
`input_cost`	Character scalar. Name of the input cost column. Default ‘"nm"‘.

Value

A ‘data.table‘ with a single column ‘markup‘.

Examples

```
library(data.table)
DT <- data.table(
  oe_l = c(0.6, 0.7, 0.5),
  nq   = c(100, 200, 150),
  nm   = c(50,  80,  60)
)
mdi_estimate_markup(DT)
```

`mdi_estimate_prodfun` *Estimate production functions with multiple estimators (panel, by-group)*

Description

`mdi_estimate_prodfun()` is a wrapper that estimates Cobb-Douglas production functions on firm-level panel data using a selectable set of estimators (methods). It is designed for *grid runs* across within-method specifications (e.g., degree choices, time fixed effects, alternative demeaning) and returns standardized outputs that can be pooled across methods and industries.

The wrapper is typically run "by industry" (or another grouping variable) using `bygroup`. Within each group, the function calls one estimator at a time and binds results into a common panel-style output format including:

- firm id and year (or generic `id`, `time`)
- elasticities with standardized names `el_<var>`
- total factor productivity proxy (`tfp`) and optional de-meaned version (`TFP_demeaned`)
- book-keeping: `NumObs`, plus method/spec identifiers if requested

Usage

```
mdi_estimate_prodfun(
  DT,
  methods,
  y,
  endog,
  exog,
  instr = NULL,
  id,
  time,
  bygroup,
  acf_args = list(),
  lp_args = list(),
  wdrg_args = list(),
  dpgmm_args = list(),
  ols_args = list(),
  cs_args = list(),
  verbose = TRUE,
  drop_empty = TRUE,
  allowed_ids = NULL
)
```

Arguments

`DT`	A `data.table` or `data.frame` with firm-level panel data.
`methods`	Character vector of methods to run. Supported: `"acf"`, `"lp"`, `"wdrg"`, `"dpgmm"`, `"ols"`, `"cs"`.
`y`	Character scalar. Output variable name (typically log output/value added).
`endog`	Character vector. Free/endogenous input(s) (e.g., log labor cost).
`exog`	Character vector. State/exogenous input(s) (e.g., log capital).

instr	Character vector. Proxy/instrument variable(s) (e.g., log materials). Required by "acf", "lp", "wdrg"; ignored otherwise.
id	Character scalar. Firm identifier column.
time	Character scalar. Time identifier column.
bygroup	Character scalar. Column name of the grouping variable (e.g. industry code); estimation is performed separately for each unique value.
acf_args, lp_args, wdrg_args, dpgmm_args, ols_args, cs_args	Optional lists of tuning arguments per method. See Details.
verbose	Logical. If TRUE, progress messages and estimation warnings are printed. Default TRUE.
drop_empty	Logical. If TRUE, groups with no successful estimations are excluded from the output. Default TRUE.
allowed_ids	Optional character vector. If non-NULL, restricts the accepted values of id to this whitelist (e.g., for MDI use: c("plantid","firmid","entid","entgrp")). Default NULL means any column name is accepted.

Details

Input convention The wrapper assumes all production-function variables are in logs unless a method explicitly requires levels (e.g., cost shares with log_values=TRUE will exponentiate internally).

Minimal required columns in DT are: output y; free input(s) endog; state input(s) exog; proxy/instruments instr (ACF/LP/WRDG); panel identifiers id, time; and the grouping variable bygroup.

Each estimator performs its own NA filtering on the variables it needs; hence the effective sample can differ by method/spec.

Methods implemented Methods are selected via methods. The wrapper recognizes:

"acf" Ackerberg-Caves-Frazer (2015) control-function estimator (Cobb-Douglas). First-stage polynomial in inputs and proxy to construct Φ, then a GMM stage with lagged inputs as instruments. Returns firm-level elasticities and a residual-based tfp.

"lp" Levinsohn-Petrin (2003) proxy estimator (Cobb-Douglas).

"wdrg" Wooldridge (2009) system-GMM estimator. Stacked two-equation GMM, estimated linearly in the parameters (the polynomial coefficients on $c(x_t, m_t)$ and $c(x_{t-1}, m_{t-1})$ are free, separate vectors; this is more general than the random-walk-with-drift case but does not impose the structural AR(G) restriction of a degree-G nonlinear law of motion). Returns common elasticities, se_el_<var>, tfp = y - X*beta, and alpha_hat as a diagnostic column.

"dpgmm" Arellano-Bond (1991) difference-GMM for the dynamic PF $y_{it} = \rho y_{i,t-1} + \beta_l l_{it} + \beta_k k_{it} + \alpha_i + \varepsilon_{it}$. First-differences to remove the firm fixed effect; lagged levels of (y, l, k) instrument the differenced regressors (K predetermined, L endogenous). One-step GMM, cluster-robust SEs, Sargan/Hansen and AR(1)/AR(2) diagnostics. Returns el_<var>, rho, tfp = y - beta_l l - beta_k k.

"ols" Pooled OLS baseline with flexible polynomial controls (Cobb-Douglas). tfp = y - X*beta. Use degree = 1 for the clean naive baseline.

"cs" Cost-shares (index-number) approach for Cobb-Douglas.

Method-specific optional arguments (xxxx_args) Each method accepts an optional list of tuning arguments passed via a dedicated parameter name: acf_args, lp_args, wdrg_args, dpgmm_args, ols_args, cs_args. Unspecified fields fall back to method defaults.

ACF arguments (acf_args): spec, degree, lower_bound_theta, upper_bound_theta, TFP_demeaned, TFP_minuend ("Phi"/"y"), Omega_estimates, time_FE, and extended_instr (logical, default FALSE; if TRUE adds Φ_{t-1} to the instrument set per ACF eq 28 for overidentification).

LP arguments (lp_args): spec, degree, lower_bound_theta, upper_bound_theta, TFP_demeaned, TFP_minuend ("y"/"Phi"), Omega_estimates, time_FE.

Wooldridge arguments (wdrg_args): degree, tol, TFP_demeaned, TFP_minuend ("y" only).

Dynamic panel GMM arguments (dpgmm_args): max_lag_Z (integer >= 1, default 2; instrument lag depth), TFP_demeaned.

OLS arguments (ols_args): spec, degree, TFP_demeaned.

Cost shares arguments (cs_args): log_values, TFP_demeaned.

Value

A data.table with one row per firm-year (or per used observation), containing bygroup, method, id, time, el_<var>, tfp, and method-specific extras (TFP_demeaned, rho, se_el_<var>, diagnostics), plus NumObs.

Examples

```
library(data.table)
set.seed(42)
n <- 200
DT <- data.table(
  firmid = rep(1:50, each = 4),
  year   = rep(2000:2003, 50),
  sector = rep(c("A", "B"), each = 100),
  y      = rnorm(n, 5, 1),
  l      = rnorm(n, 3, 0.5),
  k      = rnorm(n, 4, 0.5),
  m      = rnorm(n, 2, 0.5)
)
mdi_estimate_prodfun(
  DT, methods = c("ols", "acf"),
  y = "y", endog = "l", exog = "k", instr = "m",
  id = "firmid", time = "year", bygroup = "sector"
)
```

---

mdi_hier_apply *Hierarchical Aggregation of Data*

Description

Aggregates variables in ‘var_list‘ to unique values of the hierarchical dimensions specified in ‘hhfile‘ by groups. Aggregation is performed at each level specified by ‘hier‘ using [‘mdi_aggregate()‘].

Usage

```
mdi_hier_apply(
  DT,
  hhfile,
  var_list,
  bygroups,
  hier,
  agg_type = "sum",
  weight_col = NULL,
  mrg = FALSE,
  disclosure = TRUE
)
```

Arguments

DT	A ‘data.table‘ to be aggregated. Must contain all columns in ‘var_list‘ and ‘bygroups‘.
hhfile	A ‘data.table‘ containing the hierarchy; must include a column ‘h_0‘ matching ‘bygroups[1]‘ in ‘DT‘, plus one column per aggregation level (e.g. ‘h_1‘, ‘h_2‘).
var_list	A character vector of numeric variable names in ‘DT‘ to aggregate.
bygroups	A character vector of grouping variables in ‘DT‘. The first element must match ‘h_0‘ in ‘hhfile‘.
hier	Character. Either a single node name (e.g. ‘"h_2"‘) to aggregate ‘h_0‘ up to that level, or ‘"ALL"‘ to aggregate to every available node in ‘hhfile‘.
agg_type	Character vector of aggregation types passed to [‘mdi_aggregate()‘]. Default ‘"sum"‘.
weight_col	Optional character string naming a weight column in ‘DT‘. Passed as ‘weight_col‘ to [‘mdi_aggregate()‘]. Default ‘NULL‘.
mrg	Logical. If ‘FALSE‘, returns the aggregated result. If ‘TRUE‘, merges result back into ‘DT‘. Default ‘FALSE‘.
disclosure	Logical. If ‘TRUE‘, dominance and observation-count columns are added for disclosure control (only when ‘mrg = FALSE‘). Default ‘TRUE‘.

Value

A ‘data.table‘ containing the aggregated variables from ‘var_list‘ at each requested hierarchy level, combined via ‘rbindlist‘. A ‘node‘ column identifies the aggregation level of each row.

Examples

```
library(data.table)
hhfile <- data.table(
  h_0 = c("A1", "A2", "B1", "B2"),
  h_1 = c("A",  "A",  "B",  "B")
)
DT <- data.table(
  nace = c("A1", "A2", "B1", "B2"),
  year = rep(2020L, 4),
  emp  = c(10L, 20L, 15L, 25L)
)
mdi_hier_apply(DT, hhfile, var_list = "emp",
           bygroups = c("nace", "year"), hier = "h_1",
           disclosure = FALSE)
```

`mdi_import_data` *Import Data into R data.table*

Description

Reads data from various file formats into an R 'data.table'. This function is a wrapper around multiple file-reading packages such as 'fread' (from 'data.table'), 'haven' (for Stata, SAS, SPSS), and 'readxl' (for Excel). It supports CSV, Stata (.dta), Excel (.xlsx), SAS (.sas7bdat), SPSS (.sav), and tab-delimited text files (.txt). You can also specify a list of columns to import and specify which columns should be imported as characters. However, note that for some file formats (e.g., Stata, SAS, SPSS), the function cannot directly import columns as characters during the import process. In such cases, the specified columns are converted to character types **after** the data has been loaded.

Usage

```
mdi_import_data(
  dir,
  file,
  format,
  col_list = NULL,
  char_columns = NULL,
  encoding = NULL
)
```

Arguments

`dir`	The directory path where the input file is located.
`file`	The name of the file to be imported.

format	The type of the file to be imported. Supported types include: - ‘’csv’‘ for comma-delimited files (direct import as character supported), - ‘’txt’‘ for tab-delimited text files (direct import as character supported), - ‘’gz’‘ for gzip-compressed delimited files, - ‘’dta’‘ for Stata files (post-import conversion to character), - ‘’xlsx’‘ for Excel files (post-import conversion to character), - ‘’sas7bdat’‘ for SAS files (post-import conversion to character), - ‘’sav’‘ for SPSS files (post-import conversion to character), - ‘’parquet’‘ for Apache Parquet files (requires the ‘arrow‘ package), - ‘’rdata’‘ for R workspace files (first object loaded), - ‘’rds’‘ for R serialized single-object files.
col_list	A character vector of column names to import. If ‘NULL‘ (default), all columns are imported.
char_columns	A character vector of column names to treat as character type. For ‘csv‘ and ‘txt‘, conversion happens during import; for all other formats it happens after loading. Default ‘NULL‘.
encoding	Character string passed to the underlying reader (e.g. ‘"UTF-8"‘, ‘"Latin-1"‘). If ‘NULL‘ or empty, a format-specific default is used. Default ‘NULL‘.

Value

A ‘data.table‘ containing the imported data.

Examples

```
tmp_dir <- paste0(tempdir(), "/")
write.csv(data.frame(id = 1:3, emp = c(10, 20, 30)),
          paste0(tmp_dir, "data.csv"), row.names = FALSE)
mdi_import_data(tmp_dir, "data.csv", "csv", char_columns = "id")
```

---

mdi_intensity *Compute Technology Adoption Intensity via Probit Propensities*

Description

Reduces a set of binary adoption indicators to a single continuous intensity score using probit regressions. For each boolean indicator a probit model is fitted with continuous firm-level predictors and optional fixed effects. The predicted propensities are then combined into a single intensity score via the geometric mean.

Usage

```
mdi_intensity(DT, uniqdim, boollist, contlist, fe)
```

Arguments

DT	A data.table containing the input data.
uniqdim	Character vector of column names that uniquely identify each observation (e.g. ‘c("firmid", "year")‘).
boollist	Character vector of column names for binary adoption indicators (0/1). A probit propensity is estimated for each.
contlist	Character vector of column names for continuous firm-level predictors used in each probit model.
fe	Character vector of column names to include as factor fixed effects in each probit model (e.g. ‘c("nace2", "year")‘).

Value

A data.table keyed on ‘uniqdim‘ with one additional column ‘intens_probit‘: the geometric mean of all predicted propensities.

Examples

```
library(data.table)
set.seed(1)
n <- 100
DT <- data.table(
  firmid = seq_len(n),
  year   = sample(2010:2012, n, replace = TRUE),
  bool1  = sample(0L:1L, n, replace = TRUE),
  cont1  = rnorm(n),
  cont2  = rnorm(n)
)
mdi_intensity(DT, uniqdim = "firmid", boollist = "bool1",
          contlist = c("cont1", "cont2"), fe = "year")
```

---

mdi_jointdist *Calculate Joint Distributions*

Description

Computes joint distributions for specified variables within a data table. Calculates distributional moments (deciles, quintiles, or quartiles) for one or more variables, aggregated by specified groups and potentially hierarchical structures.

Usage

```
mdi_jointdist(
  DT,
  hhfile,
  qnames,
```

```
  var_names,
  moment = c("decile", "quintile", "quartile"),
  bygroups,
  hier,
  agg_type,
  prefix = agg_type,
  weight_col = NULL,
  mrg = FALSE,
  disclosure = TRUE
)
```

Arguments

DT	A ‘data.table‘ containing variables for distribution calculations.
hhfile	A ‘data.table‘ containing hierarchical information for aggregation. Must have an ‘h_0‘ column matching ‘bygroups[1]‘.
qnames	Character vector. Names of variables used for calculating distributional moments.
var_names	Character vector. Names of the variable(s) whose aggregates are computed.
moment	Character scalar. Distributional moment to compute. One of ‘"decile"‘, ‘"quintile"‘, or ‘"quartile"‘. Default ‘"decile"‘.
bygroups	Character vector. Variables used for stratification.
hier	Character scalar. Hierarchical level for aggregation. ‘"ALL"‘ uses all levels in ‘hhfile‘; otherwise specify a column name.
agg_type	Character scalar. Type of aggregation (e.g. ‘"sum"‘, ‘"mean"‘).
prefix	Character scalar. Prefix for naming aggregated variables. Default is the value of ‘agg_type‘.
weight_col	Optional character scalar. Name of a weight column in ‘DT‘ for weighted aggregation.
mrg	Logical. Whether to merge results back with the original data table. Default ‘FALSE‘.
disclosure	Logical. Whether to apply disclosure control. Default ‘TRUE‘.

Value

A ‘data.table‘ with computed joint distributions including the distributional moments for specified variables aggregated by the given criteria.

Examples

```
library(data.table)
DT <- data.table(
  nace = rep(c("A", "B"), each = 5),
  year = rep(2020L, 10),
  emp  = c(10, 20, 5, 15, 8, 12, 25, 6, 14, 9)
)
```

```
hhfile <- data.table(h_0 = c("A", "B"), h_1 = c("X", "X"))
mdi_jointdist(DT, hhfile,
  qnames = "emp", var_names = "emp", moment = "quartile",
  bygroups = c("nace", "year"), hier = "h_1",
  agg_type = "sum", disclosure = FALSE)
```

`mdi_lp_prodest` *Levinsohn-Petrin (2003) Production Function Estimator*

Description

Two-stage proxy-variable estimator using an intermediate input (typically materials) to control for unobserved productivity. First stage: regress y on a flexible polynomial in (state, proxy) plus the free input, recovering Phi. Second stage: GMM on the law-of-motion residual identifying the state elasticities. Reference: Levinsohn & Petrin (2003, ReStud).

tfp = (y or Phi) - X*beta, depending on `TFP_minuend`. Defaults to `"y"` to match the convention shared by the other estimators.

Usage

```
mdi_lp_prodest(
  DT,
  y,
  endog,
  exog,
  instr,
  id,
  time,
  spec = "cd",
  degree = 3,
  lower_bound_theta = 0,
  upper_bound_theta = 1,
  TFP_demeaned = TRUE,
  TFP_minuend = c("y", "Phi"),
  Omega_estimates = TRUE,
  time_FE = FALSE
)
```

Arguments

`DT`	A `data.table` (or coercible object) containing panel data.
`y`	Character. Name of the output variable column.
`endog`	Character vector. Names of endogenous input columns (e.g. labour).
`exog`	Character vector. Names of exogenous input columns (e.g. capital).

instr	Character vector. Names of proxy/instrument columns (e.g. materials).
id	Character. Name of the firm/unit identifier column.
time	Character. Name of the time period column.
spec	Character. Functional form. Only "cd" implemented. Default "cd".
degree	Integer. Polynomial degree in proxy function and law of motion. Default 3.
lower_bound_theta	Numeric. Lower bound for state elasticities. Default 0.
upper_bound_theta	Numeric. Upper bound for state elasticities. Default 1.
TFP_demeaned	Logical. If TRUE, TFP is demeaned by subtracting the period mean (via mdi_aggregate). Default TRUE.
TFP_minuend	Character. "y" (default) or "Phi".
Omega_estimates	Logical. If TRUE, attaches law-of-motion parameters (g_b_slopes, g_b_intercept). Default TRUE.
time_FE	Logical. If TRUE, period dummies enter the first stage. Default FALSE.

Value

A data.table with one row per observation in the GMM sample, containing: - id, time columns (using the names supplied) - tfp: total factor productivity - el_<input>: estimated input elasticities - TFP_demeaned (if TFP_demeaned = TRUE): period-demeaned TFP - g_b_slopes, g_b_intercept (if Omega_estimates = TRUE) - NumObs: number of observations used in the GMM stage - convergence: optimiser convergence code (0 = converged)

Examples

```
library(data.table)
set.seed(1)
n <- 200
DT <- data.table(
  id   = rep(1:50, each = 4),
  year = rep(2000:2003, times = 50),
  y    = rnorm(n, 5, 1),
  l    = rnorm(n, 3, 0.5),
  k    = rnorm(n, 4, 0.5),
  m    = rnorm(n, 2, 0.5)
)
result <- mdi_lp_prodest(DT, y = "y", endog = "l", exog = "k", instr = "m",
                         id = "id", time = "year", degree = 2,
                         TFP_demeaned = FALSE)
```

`mdi_make_conc` *Harmonize a Classification Over Time*

Description

Creates a harmonized concordance for a classification of interest over a time period, starting from year-by-year concordance tables. Each row in ‘conc_table‘ maps a code in year ‘t-1‘ (column ‘left‘) to a code in year ‘t‘ (second column), along with a ‘year‘ column indicating which transition the row belongs to.

The function handles 1:1, m:1, 1:m, and m:m code changes, and appends ‘"D"‘ to harmonized codes that disappear before the last year.

Usage

```
mdi_make_conc(conc_table, year_list, code_name)
```

Arguments

`conc_table`	A ‘data.table‘ of concordance mappings with at least three columns: ‘year‘ (integer transition year), ‘left‘ (code at ‘t-1‘), and a second code column (code at ‘t‘). Rows must cover all transitions within ‘year_list‘.
`year_list`	Integer or numeric vector of years of interest, starting from the first year ‘t‘ in the first concordance table (not ‘t-1‘).
`code_name`	Character. Name of the classification variable, used to label output columns (e.g. ‘"pcc8"‘ produces columns ‘pcc8‘ and ‘pcc8_harmonized‘).

Value

A ‘data.table‘ in long format with columns:

- ‘year‘ — the year of the observation.
- ‘<code_name>‘ — the original code for that year.
- ‘<code_name>_harmonized‘ — the harmonized code mapped to the most recent non-missing code in the group.

Examples

```
library(data.table)
conc <- data.table(
  year  = c(2011L, 2011L, 2012L, 2012L),
  left  = c("A",   "B",   "A",   "C"),
  right = c("A",   "B",   "A2",  "C")
)
mdi_make_conc(conc, 2011:2012, "pcc")
```

mdi_ols_prodest *OLS Production Function Estimator*

Description

Fits a Cobb-Douglas production function by OLS on the free and state inputs together with a degree-G polynomial in the same inputs, following the prodest-package convention. Returns TFP as the residual between output and the fitted input index, with optional period demeaning.

Intended as a naive baseline for the simultaneity-bias comparison against the proxy-variable estimators. NOTE: at `degree > 1` the reported linear-input coefficients are partial regression coefficients conditional on the higher-order polynomial terms, not structural elasticities. Use `degree = 1` for the clean naive-OLS baseline.

tfp = y - X*beta (same convention as ACF/LP/WDRG/dpGMM).

Usage

```
mdi_ols_prodest(
  DT,
  y,
  endog,
  exog,
  id,
  time,
  spec = "cd",
  degree = 3,
  TFP_demeaned = TRUE
)
```

Arguments

`DT`	A `data.table` (or coercible object) containing panel data.
`y`	Character scalar. Name of the output variable column.
`endog`	Character vector. Names of the free (endogenous) input columns (e.g. log labour).
`exog`	Character vector. Names of the state input columns (e.g. log capital).
`id`	Character scalar. Name of the firm/unit identifier column.
`time`	Character scalar. Name of the time period column.
`spec`	Character. Functional form. Only `"cd"` (Cobb-Douglas) is implemented. Default `"cd"`.
`degree`	Integer. Polynomial degree for the input polynomial. Use 1 for the true naive baseline. Default `3`.
`TFP_demeaned`	Logical. If `TRUE`, TFP is demeaned by subtracting the period mean (via `mdi_aggregate`). Default `TRUE`.

Value

A data.table with one row per observation in the estimation sample, containing: - id, time columns (using the names supplied) - tfp: total factor productivity (OLS residual) - el_<endog>, el_<exog>: estimated input elasticities - TFP_demeaned (if TFP_demeaned = TRUE): period-demeaned TFP - NumObs: number of observations used

Examples

```
library(data.table)
set.seed(3)
n <- 200
DT <- data.table(
  id   = rep(1:50, each = 4),
  year = rep(2000:2003, times = 50),
  y    = rnorm(n, 5, 1),
  l    = rnorm(n, 3, 0.5),
  k    = rnorm(n, 4, 0.5)
)
result <- mdi_ols_prodest(DT, y = "y", endog = "l", exog = "k",
                          id = "id", time = "year", degree = 2,
                          TFP_demeaned = FALSE)
```

---

mdi_outlier *Outlier Routine for Data Cleaning*

Description

Executes a specified outlier handling routine (trimming, winsorizing, or flagging) on a dataset for selected continuous variables. This function can trim or winsorize the data at specified quantiles or flag observations as outliers based on the fraction provided. Trimming replaces outliers with NA, winsorizing replaces outliers with the closest value within the non-outlier range, and flagging marks outliers with a flag variable.

Usage

```
mdi_outlier(
  DT,
  var_list,
  routine = c("trim", "winsorize", "flag"),
  fraction,
  both_tails = FALSE,
  group = NULL
)
```

Arguments

DT	A ‘data.table‘ containing the variables to process.
var_list	A character vector of continuous variable names in ‘DT‘ to be processed by the outlier routine.
routine	A string specifying the outlier routine to apply: ‘"trim"‘, ‘"winsorize"‘, or ‘"flag"‘. Default ‘"trim"‘.
fraction	The fraction of data to be trimmed or winsorized; must be a numeric value between 0 and 1.
both_tails	Logical indicating whether to apply the routine to both tails of the distribution. If ‘TRUE‘, the operation affects both the upper and lower tails; otherwise, it affects only the upper tail. Default ‘FALSE‘.
group	An optional character string naming a grouping variable in ‘DT‘. When supplied, the outlier routine is applied within each group. Default ‘NULL‘.

Value

A modified copy of ‘DT‘ with the outlier routine applied to the specified variables. If ‘routine = "flag"‘, new columns named ‘flag_<var>‘ are added (value ‘1‘ for flagged observations, ‘NA‘ otherwise).

Examples

```
library(data.table)
DT <- data.table(id = 1:10, income = c(50, 55, 45, 60, 200, 40, 45, 55, 65, 1000))
mdi_outlier(DT, "income", "trim", 0.1, both_tails = TRUE)
mdi_outlier(DT, "income", "winsorize", 0.1)
mdi_outlier(DT, "income", "flag", 0.1)
```

---

mdi_pim_capital *Estimate Capital Stock Using Perpetual Inventory Method (PIM)*

Description

Estimates capital stock based on the Perpetual Inventory Method (PIM), as outlined in Halle & Mairesse (1995). This function is tailored for data structured as panel data, with indexing based on a firm identifier (‘firmid‘) and a year variable. It involves the following steps: - Reading the specified variables from the input data.table. - Depending on whether the depreciation rate is provided directly or inferred from the asset type, the function calculates the capital stock. - The output is the input data.table augmented with the new capital stock variable.

The function supports different depreciation formats and asset types, offering flexibility in estimating capital stock across various contexts.

Usage

```
mdi_pim_capital(
  DT,
  id = "firmid",
  K0 = "K0",
  I = "ni_tan",
  delta = "d_GFCF",
  output_name = "k_new",
  time = "year"
)
```

Arguments

DT	A data.table including the necessary variables.
id	Column name of the firm (unit) identifier. Default is ‘"firmid"‘.
K0	Column name for the real initial capital stock value. Default is ‘"K0"‘.
I	Column name for real investment. Default is ‘"ni_tan"‘.
delta	Column name for the depreciation rate. Default is ‘"d_GFCF"‘.
output_name	Name of the output variable for the estimated capital stock. Default is ‘"k_new"‘.
time	Name of the time variable. Default is ‘"year"‘.

Value

A modified ‘DT‘ with the new capital stock variable appended.

References

Halle, P., & Mairesse, J. (1995). "Estimation of the Perpetual Inventory Method for Capital Stock".

Examples

```
library(data.table)
DT <- data.table(
  firmid = rep(c("F1", "F2"), each = 4),
  year   = rep(2010:2013, 2),
  K0     = c(100, NA, NA, NA, 200, NA, NA, NA),
  ni_tan = c(10, 12, 11, 13, 20, 22, 21, 23),
  d_GFCF = rep(0.08, 8)
)

mdi_pim_capital(DT)
```

mdi_regress *Weighted Regression with Automatic Disclosure Check*

Description

Runs regression analysis on a ‘data.table‘ using models from the **fixest** package (‘feols‘, ‘feglm‘, etc.), with automatic disclosure control applied.

Disclosure rules ensure that regressions are only reported if they meet minimum thresholds for sample size: both the number of observations and the residual degrees of freedom must be at least ‘minNumObs‘.

When ‘count_firms = TRUE‘, the number of unique firms and enterprises in the regression sample are also computed and added to the output (‘NumFirms‘, ‘NumEnt‘). This is controlled by the caller – typically set based on the country-specific disclosure requirements of the NSI running the code.

The function supports clustered standard errors, regression weights, instrumental variables, and optional LaTeX export of results.

Usage

```
mdi_regress(
  DT,
  formula,
  model = "feols",
  family = NULL,
  vcov = "iid",
  cluster = FALSE,
  tex = FALSE,
  output_name = NULL,
  desc_file = NULL,
  weights = NULL,
  iv = FALSE,
  num_firms = NULL,
  num_ent = NULL,
  count_firms = FALSE,
  firm_col = "firmid",
  ent_col = "entid",
  minNumObs = 5L,
  dirOUTPUT = NULL
)
```

Arguments

DT	A ‘data.table‘ containing the dataset for regression analysis.
formula	A character vector of regression formulas.
model	Character. The regression model to use (e.g. ‘"feols"‘, ‘"feglm"‘). Default is ‘"feols"‘.

family	Family specification for GLM models (only relevant if 'model = "feglm"'). Example: 'binomial'.
vcov	Variance-covariance specification for standard errors. Can be '"iid"' (default) or a formula for clustered SE (e.g. '~clustervar').
cluster	Logical. If 'TRUE', standard errors are clustered. In this case 'vcov' must specify the clustering variable. Default is 'FALSE'.
tex	Logical. If 'TRUE', regression results are exported as a LaTeX table using 'etable()'. Default is 'FALSE'.
output_name	Character. Name of the LaTeX output file (without extension), if 'tex = TRUE'. Default is 'NULL'.
desc_file	Character. Name of a description file to which an entry for the regression output is appended (if 'tex = TRUE'). Default is 'NULL'.
weights	Optional formula specifying the weights variable to apply in the regression (e.g. '~myweights'). Default is 'NULL'.
iv	Logical. If 'TRUE', the regression is treated as an instrumental variable model and IV-specific fit statistics are reported. Default is 'FALSE'.
num_firms	Optional integer. Number of unique firms to use when 'count_firms = TRUE' but 'firm_col' is not found in the data. Default 'NULL'.
num_ent	Optional integer. Number of unique enterprises to use when 'count_firms = TRUE' but 'ent_col' is not found in the data. Default 'NULL'.
count_firms	Logical. If 'TRUE', the number of unique firms and enterprises in the regression sample are computed and added to the output as 'NumFirms' and 'NumEnt'. The columns used are controlled by 'firm_col' and 'ent_col'. Set to 'TRUE' when NSI disclosure rules require firm-level counts. Default 'FALSE'.
firm_col	Character. Name of the column in 'DT' that identifies firms, used to count unique firms when 'count_firms = TRUE'. Falls back to 'num_firms' if the column is absent. Default '"firmid"'.
ent_col	Character. Name of the column in 'DT' that identifies enterprises, used to count unique enterprises when 'count_firms = TRUE'. Falls back to 'num_ent' if the column is absent. Default '"entid"'.
minNumObs	Integer. Minimum observations and degrees of freedom required for a regression to pass disclosure. Default '5L'.
dirOUTPUT	Character. Path to the output directory (must end with '"/"'), used when 'tex = TRUE'. Default 'NULL'.

Details

- Regression output is only returned if both 'nobs >= minNumObs' and residual degrees of freedom '>= minNumObs'. - When 'count_firms = TRUE': unique firm and enterprise counts are computed from the regression sample using 'firm_col' and 'ent_col'. If a column is absent from the data, the corresponding 'num_firms' or 'num_ent' argument is used instead. If neither the column nor the manual count is provided, the function stops with an error before any regression is run. - If 'tex = TRUE', results are saved to '<dirOUTPUT>/<output_name>.tex' and an entry is appended to '<dirOUTPUT>/<desc_file>.txt'.

Value

A ‘data.table‘ of regression results, including: - Coefficients and standard errors - Confidence intervals (‘ci.lower‘, ‘ci.upper‘) - Sample size (‘NumObs‘), residual df (‘df‘) - ‘NumFirms‘ and ‘NumEnt‘ (only when ‘count_firms = TRUE‘) - Fit statistics (R2, Adj. R2, AIC, BIC, LogLik, and IV/F tests if applicable)

If disclosure criteria are not satisfied, the regression is skipped and a message is printed.

Examples

```
library(data.table)
set.seed(1)
DT <- data.table(
  y      = rnorm(100),
  x1     = rnorm(100),
  x2     = rnorm(100),
  firmid = paste0("F", sample(1:20, 100, replace = TRUE)),
  entid  = paste0("E", sample(1:15, 100, replace = TRUE))
)
# Basic regression
mdi_regress(DT, formula = "y ~ x1 + x2", minNumObs = 5L)

# With firm/enterprise counts (e.g. when NSI rules require it)
mdi_regress(DT, formula = "y ~ x1 + x2", minNumObs = 5L,
            count_firms = TRUE)


# Weighted regression with clustered SEs
DT[, w := runif(.N)]
mdi_regress(DT, formula = "y ~ x1 + x2", weights = ~w,
            cluster = TRUE, vcov = "~firmid")

# With LaTeX export (writes .tex file to tempdir)
mdi_regress(DT, formula = "y ~ x1", tex = TRUE,
            output_name = "reg1", desc_file = "log",
            dirOUTPUT = paste0(tempdir(), "/"))
```

`mdi_transition` *Count Classification Switches Between Consecutive Periods*

Description

Identifies pairs of classification codes between which a unit switched in a given year, and counts how often each specific switch is observed across units. The output has one row per (old code, new code, year) triple.

Usage

```
mdi_transition(DT, id, time, classvar)
```

Arguments

DT	A data.table containing the panel data. Modified in place by sorting on ‘id‘ and ‘time‘ before processing.
id	Character. Name of the unit identifier column (e.g. ‘"firmid"‘).
time	Character. Name of the time variable column (e.g. ‘"year"‘).
classvar	Character. Name of the classification variable column (e.g. ‘"nace"‘).

Value

A data.table with four columns: ‘old_code‘, ‘new_code‘, ‘year_shifted‘, and ‘N‘ (count of units making that switch in that year).

Examples

```
library(data.table)
DT <- data.table(
  firmid = c("A", "A", "A", "B", "B", "B"),
  year   = c(2010L, 2011L, 2012L, 2010L, 2011L, 2012L),
  nace   = c("C10", "C10", "C20", "D30", "D40", "D40")
)
mdi_transition(DT, id = "firmid", time = "year", classvar = "nace")
```

---

mdi_wdrg_prodest *Wooldridge (2009) System-GMM Production Function Estimator*

Description

Implements the stacked two-equation GMM from Wooldridge (2009, Economics Letters 104, 112-114). The system is estimated linearly in the parameters: the polynomial coefficients on (x_t, m_t) (Wooldridge eq 3.3) and on (x_{t-1}, m_{t-1}) (eq 3.4) are free, separate vectors. This is more general than the random-walk-with-drift restriction (eq 3.10) which forces them equal, but it does NOT impose the structural rho_g * lambda^g form that eq (3.4) implies for a general degree-G polynomial law of motion (G > 1, which would require nonlinear GMM and is not implemented here). The implementation matches the standard empirical Wooldridge GMM used in Petrin, Poi and Levinsohn (2004).

Returns elasticities common to all firms in the group, firm-year `tfp = y - X*beta` (the intercept `alpha_hat` is reported as a separate column for diagnostics, not subtracted), and analytical SEs.

Usage

```
mdi_wdrg_prodest(
  DT,
  y,
  endog,
  exog,
  instr,
```

```
  id,
  time,
  degree = 2,
  tol = 1e-10,
  TFP_demeaned = TRUE,
  TFP_minuend = "y"
)
```

Arguments

DT	A data.table (or coercible object) containing panel data.
y	Character. Output variable name (in logs).
endog	Character. Free input name (e.g. labour; one variable for now).
exog	Character. State input name (e.g. capital; one variable for now).
instr	Character. Proxy variable name (e.g. materials).
id	Character. Firm identifier column.
time	Character. Time identifier column.
degree	Integer. Polynomial degree for c(x, m). Default 2.
tol	Numeric. Linear-solver tolerance for the GMM normal equations. Default 1e-10.
TFP_demeaned	Logical. If TRUE, returns TFP_demeaned (tfp - period mean). Default TRUE.
TFP_minuend	Character. Currently only "y" is supported.

Value

A data.table with one row per observation in the GMM sample, containing id, time, tfp, alpha_hat, el_<input>, se_el_<input>, NumObs, and optionally TFP_demeaned.

Examples

```
library(data.table)
set.seed(1)
n <- 200
DT <- data.table(
  id   = rep(1:50, each = 4),
  year = rep(2000:2003, times = 50),
  y    = rnorm(n, 5, 1),
  l    = rnorm(n, 3, 0.5),
  k    = rnorm(n, 4, 0.5),
  m    = rnorm(n, 2, 0.5)
)
result <- mdi_wdrg_prodest(DT, y = "y", endog = "l", exog = "k", instr = "m",
                           id = "id", time = "year", TFP_demeaned = FALSE)
```

Index