

Package: iotables (via r-universe)

September 24, 2026

Type Package

Title Reproducible Input–Output Economics Analysis, Economic and
Environmental Impact Assessment with Empirical Data

Version 0.9.5

Date 2025-11-23

Maintainer Daniel Antal <daniel.antal@dataobservatory.eu>

Description Pre-processing and basic analytical tasks for working with
Eurostat's symmetric input–output tables, and basic
input–output economics calculations. Part of rOpenGov
<<https://ropengov.github.io/>> for open source open government
initiatives.

License MIT + file LICENSE

URL <https://iotables.dataobservatory.eu/>,
<https://github.com/rOpenGov/iotables>

BugReports <https://github.com/rOpenGov/iotables/issues>

LazyData true

Depends R (>= 3.5.0)

Imports assertthat, dplyr, eurostat, forcats, glue, httr2, kableExtra,
knitr, lubridate, magrittr, readxl, rlang, stats, tibble,
tidyr, tidyselect, utils

Suggests covr, mockery, rmarkdown, spelling, testthat (>= 3.0.0)

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/pak/sysreqs

cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-
dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev

Repository <https://ropengov.r-universe.dev>
Date/Publication 2025-11-23 17:44:59 UTC
RemoteUrl <https://github.com/rOpenGov/iotables>
RemoteRef HEAD
RemoteSha 84e420fca6d6d1447db820f738ff8db7d13faf20

Contents

airpol_get	3
align_to_io_codes	4
backward_linkages	6
coefficient_matrix_create	7
conforming_vector_create	10
convert_industry_to_product	11
cpa2_1	12
croatia_2010_1700	13
croatia_2010_1800	14
croatia_2010_1900	15
croatia_employment_2013	16
croatia_employment_aggregation	16
direct_effects_create	17
employment_get	18
employment_metadata	19
empty_remove	20
equation_solve	21
forward_linkages	22
germany_1995	23
germany_airpol	24
ghosh_inverse_create	25
household_column_find	26
household_column_get	27
ind_ava	28
ind_use	29
indirect_effects_create	30
input_coefficient_matrix_create	31
input_flow_get	32
input_indicator_create	34
input_multipliers_create	35
iotable_get	36
iotable_get_builtin	38
iotable_get_eurostat	40
iotable_year_get	41
iotables	43
iotables_download	43
iotables_download_old	44
iotables_metadata_get	46
iotables_read_tempdir	47

is_html_output	48
is_latex_output	48
key_column_create	48
leontief_inverse_create	49
leontief_matrix_create	50
matrix_round	52
metadata_uk_2010	53
multiplier_create	54
netherlands_2000	55
output_coefficient_matrix_create	56
output_get	58
output_multiplier_create	59
prd_ava	60
prd_use	61
primary_input_get	62
primary_inputs	63
rows_add	64
supplementary_add	65
total_tax_add	67
uk_2010_data	69
uk_2010_results_get	70
uk_test_results	71
vector_transpose_longer	72
vector_transpose_wider	73
Index	75

airpol_get	<i>Get air pollutant data (Eurostat env_ac_ainah_r2)</i>
------------	--

Description

Retrieves Air Emissions Accounts (env_ac_ainah_r2) by NACE Rev. 2 activity for environmental impact assessments. Automatically manages Eurostat caching via `tempdir()` or a user-specified `data_directory`.

Usage

```
airpol_get(
  airpol = "GHG",
  geo = "BE",
  year = 2020,
  unit = "THS_T",
  data_directory = NULL,
  force_download = FALSE
)
```

Arguments

<code>airpol</code>	Pollutant code (e.g. "GHG", "CO2", "CH4", "NOX", "NH3", etc.).
<code>geo</code>	Country code (e.g. "BE", "DE"). The special value "germany_1995" returns the built-in example dataset germany_airpol .
<code>year</code>	Reference year (2008 for NACE Rev. 2).
<code>unit</code>	Unit of measure (default "THS_T" = thousand tonnes).
<code>data_directory</code>	Directory to save processed data (default: <code>tempdir()</code>).
<code>force_download</code>	Logical, whether to force a new Eurostat download.

Details

The Eurostat dataset *Air emissions accounts by NACE Rev. 2 activity* (`env_ac_ainah_r2`) contains emissions of major pollutants, including: CO2, biomass CO2, N2O, CH4, PFCs, HFCs, SF6, NOx, NMVOC, CO, PM10, PM2.5, SO2, and NH3. See Eurostat metadata for definitions of aggregated indicators (GHG, ACG, O3PR, etc.).

Value

A tidy data frame with air pollutant emissions aligned to IO classifications.

Note

The function adjusts the invalid industry codes, like C10-C12 to C10-12.

Source

Eurostat dataset: `env_ac_ainah_r2`

<code>align_to_io_codes</code>	<i>Align External Industry-Based Data to an Input–Output Table</i>
--------------------------------	--

Description

`align_to_io_codes()` harmonises an external dataset (such as air pollution data, energy use, employment, or satellite accounts) to the industry structure of an input–output (IO) table.

The function standardises industry identifiers, drops irrelevant aggregate categories, pads missing industries with zeros, and reorders columns so that identifiers appear first followed by IO industry codes in the correct order.

This enables consistent integration of satellite accounts with IO tables for downstream operations such as indicator creation, coefficient matrix construction, or Leontief multiplier analysis.

Usage

```
align_to_io_codes(ext_data, io_data)
```

Arguments

- | | |
|-----------------|--|
| ext_data | A data frame or tibble containing external data whose column names correspond to NACE or NACE-like industry codes. Must include at least one identifier column, such as "indicator", "multiplier", "prod_na", "industry", "sector", or another variable name ending in "_id" or "_code". |
| io_data | A data frame or tibble representing the structured IO table whose column names define the target industry codes. These will be used to reorder and pad the external dataset. |

Details

The function performs six harmonisation steps:

1. Detect identifier columns using a combination of known names ("indicator", "multiplier", "prod_na", etc.) and regular expressions matching "_id\$", "_code\$", or "indicator".
2. Remove high-level aggregates (e.g., "TOTAL", "HH", "TOTAL_HH") that do not correspond to IO industries.
3. Harmonise naming conventions including common underscore/ hyphen inconsistencies (e.g., "J62-J63" → "J62_63").
4. Add missing industry columns that exist in **io_data** but not in **ext_data**. These are initialised to zero.
5. Reorder the resulting dataset so that identifier columns appear first, followed by industry columns in IO order.
6. Return the cleaned tibble ready for combination with the IO table (e.g., via **supplementary_add()**).

Value

A tibble with:

- the detected identifier columns first
- followed by all industry codes from **io_data** in the same order

Industry columns missing from **ext_data** are added and filled with zeros.

Typical use cases

- Align NAMEA air emission accounts to IO industries
- Align energy-use accounts, employment, material flows
- Prepare satellite accounts for Leontief multiplier analysis
- Harmonise any NACE-like external dataset to IO structure

See Also

[iotable_get](#), [supplementary_add](#), [input_indicator_create](#), [leontief_inverse_create](#)

Examples

```
# Minimal fictitious IO table with three industries
io_example <- data.frame(
  prod_na = c("A", "B", "C31_32"),
  A = c(10, 2, 1),
  B = c(1, 15, 3),
  C31_32 = c(0, 1, 20),
  check.names = FALSE
)

# External satellite account:
# * includes an identifier column
# * matches industries A and B
# * uses a hyphenated code C31-32 that needs harmonisation
# * includes TOTAL which should be dropped
ext_example <- data.frame(
  indicator = "GHG_emission",
  A = 0.5,
  B = 0.2,
  `C31-32` = 0.3,
  TOTAL = 0.7,
  check.names = FALSE
)

# Align external data to the IO industry codes
aligned <- align_to_io_codes(ext_example, io_example)

# The aligned table contains:
# * a single identifier column named 'prod_na'
# * only IO industries (A, B, C31_32)
# * renamed and reordered industry columns
aligned
```

backward_linkages *Backward Linkages*

Description

Compute the backward linkages of each industry or product sector from a Leontief inverse matrix. Backward linkages indicate how strongly a sector is interconnected on the demand side: when a sector increases its output, it will increase intermediate demand on all other sectors.

Usage

```
backward_linkages(Im)
```

Arguments

Im A Leontief inverse matrix created by `leontief_inverse_create()`.

Details

Backward linkages are defined as the column sums of the Leontief inverse, in line with the *Eurostat Manual of Supply, Use and Input-Output Tables* (pp. 506–507) and the *United Nations Handbook on Supply and Use Tables and Input-Output Tables with Extensions and Applications* (p. 636).

Value

A one-row `data.frame` containing the backward linkage values for each column (industry or product) of the Leontief inverse. The first column is the sector key column, and the remaining columns correspond to the linkage values.

See Also

Other linkage functions: `forward_linkages()`

Examples

```
de_coeff <- input_coefficient_matrix_create(iotable_get(), digits = 4)
I <- leontief_inverse_create(de_coeff)
backward_linkages(I)

# Trivial example: identity matrix gives linkages = 1
I <- diag(3)
colnames(I) <- rownames(I) <- c("A", "B", "C")
I_df <- data.frame(sector = rownames(I), I, check.names = FALSE)
backward_linkages(I_df)
```

`coefficient_matrix_create`

Create a technology (input) coefficient matrix

Description

Creates an **input (technical) coefficient matrix** from a symmetric input–output table (SIOT) or similar product $\times$ product table.

The function divides every element of the inter-industry use block by an output (or other) total, producing the classical technology matrix $\backslash(A\backslash)$ used for Leontief models.

In addition, when `households = TRUE`, the function constructs the **Type-II extension**:

- a *household consumption column* (typically `P3_S14`) rescaled as coefficients, and
- a *household earnings row* (typically `D1`) identifying labour income.

This produces an augmented coefficient matrix that can be used directly for Type-II multipliers (closed household sector).

Usage

```
coefficient_matrix_create(
  data_table,
  total = "output",
  digits = NULL,
  remove_empty = TRUE,
  households = FALSE,
  return_part = NULL
)
```

Arguments

<code>data_table</code>	A symmetric input-output table or any wide-format product $\times$ product table returned by <code>iotable_get()</code> or built-in datasets. Must contain a key row column (e.g. <code>prod_na</code> , <code>t_rows2</code> , <code>uk_row</code> , or <code>iotables_row</code>).
<code>total</code>	Character. Name of the row used for denominator (output). Defaults to "output". Recognised values include: "output", "P1", "output_bp", "total", "cpa_total", and synonyms found in the data.
<code>digits</code>	Integer. Number of digits to round the result to. Defaults to NULL (no rounding).
<code>remove_empty</code>	Logical. Remove empty rows/columns (default TRUE). Recommended for avoiding division by zero.
<code>households</code>	Logical. When TRUE, constructs a Type-II extended coefficient matrix including a household consumption column and household earnings row. Default FALSE.
<code>return_part</code>	Optional. One of: • "products" or "industries" – return only the technology matrix • "primary_inputs" – return only primary input coefficients • NULL – return the full matrix (default)

Details

The coefficient matrix is computed as:

$$A_{ij} = Z_{ij}/x_j$$

where Z is the intermediate use matrix and x is the chosen denominator row (usually output P1).

The denominator row is detected case-insensitively using the vocabulary: "output", "p1", "output_bp", "total output", or "total" / "cpa_total" depending on the `total` argument.

The function supports returning either:

- the full matrix (`return_part = NULL`),
- only the technology block (`return_part = "products" or "industries"`),
- only the primary inputs (`return_part = "primary_inputs"`).

When `households = TRUE`:

- the function identifies the household final-consumption column (e.g. `P3_S14` or `final_consumption_household`),
- divides it by the corresponding output totals,
- constructs the appropriate coefficient column,
- includes the household earnings row (e.g. `D1`) for Type-II multiplier calculations.

This behaviour matches traditional Type-II IO modelling practice.

Value

A `data.frame` containing:

- the key row identifier (first column),
- technology coefficients (columns 2...n),
- and, when `households = TRUE`, the household coefficient column.

If `return_part` is used, only the requested block is returned.

References

Beutel, J. (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*, ch. 15. Luxembourg: Publications Office of the European Union.

See Also

Other indicator functions: `direct_effects_create()`

Examples

```
# Technology matrix (Germany 1995 example)
cm <- coefficient_matrix_create(
  data_table = iotable_get(source = "germany_1995"),
  total = "output",
  digits = 4
)

# Type-II coefficient matrix with household column
cm_hh <- coefficient_matrix_create(
  data_table = iotable_get(source = "germany_1995"),
  households = TRUE
)
```

```
conforming_vector_create
```

Create an Empty Conforming Vector

Description

Create a named vector (in wide format) that conforms to the structure of a given analytical object, such as a use table, coefficient matrix, or Leontief matrix. This helps avoid mistakes when manually defining large vectors (e.g., for 60×60 matrices).

Usage

```
conforming_vector_create(data_table)
```

Arguments

data_table A use table, coefficient matrix, Leontief matrix, or other named matrix or data frame.

Details

The empty conforming vector can also be exported to `.csv` format and used as a template for importing scenarios from a spreadsheet application.

Value

A one-row `data.frame` with the same column names as `data_table`, but with all values set to zero.

See Also

Other iotables processing functions: [empty_remove\(\)](#), [household_column_find\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
de_input_flow <- input_flow_get(data_table = iotable_get())
conforming_vector_create(de_input_flow)
```

`convert_industry_to_product`*Convert industry columns to product columns*

Description

A utility function to adjust column names for supplementary data, for example, air pollution data, where the industry codes are denoted with `ind_ava` or `ind_use` (or similar codes).

Usage

```
convert_industry_to_product(data_table)
```

Arguments

`data_table` A table where the names of the columns are denoted with the `ind_ava` or `ind_use` or `ind_use` vocabularies, for example, agriculture is denoted with A01.

Details

For example, A01 (agriculture) is converted to CPA_A01 (product of agriculture).

This corresponds to the bridge matrix (B) defined in Eurostat Manual §15.4.2, used to transform industry-based extensions to product-based form.

Value

A converted, named vector or table where the columns are denoted with `prd_ava` codes, i.e., A01 is converted to CPA_A01.

See Also

[`supplementary\_add\(\)`](#)

Examples

```
data_table <- data.frame(  
  ind_use = "A01",  
  A01 = 1, A02 = 3, TOTAL = 4,  
  HH = 2  
)  
convert_industry_to_product(data_table)
```

cpa2_1	<i>Eurostat Product Classification (CPA 2.1 — Summary and Aggregate Levels)</i>
--------	---

Description

A reference vocabulary derived from Eurostat's **CPA 2.1 — Statistical Classification of Products by Activity**, restricted to high-level (summary) items, extensions, and 0-, 1-, and 2-digit product aggregates.

Usage

cpa2_1

Format

A tibble (data frame) with 10 variables:

id Canonical Eurostat concept identifier (SKOS URI fragment).

label Official English label from the Eurostat CPA 2.1 vocabulary.

status Concept validity flag ("**valid**" for all entries).

status_modified Date of the most recent status update.

notation Official CPA 2.1 notation (e.g. "CPA_A01", "CPA_C10").

quadrant Integer flag used internally in **iotables** to identify the analytical block (10 = intermediate, 20 = value added, 30 = control totals).

numeric_order Ordinal position key ensuring correct row sequencing in reshaped input–output matrices.

iotables_label Simplified legacy label used internally by the **iotables** package.

block Semantic block category inferred from **quadrant** (**intermediate**, **primary_inputs**, **control_total**, **extension**).

uri Stable URI for the SKOS concept at https://dd.eionet.europa.eu/vocabulary/eurostat/cpa2_1/.

Details

This vocabulary provides the **product-side** codes used in Eurostat's **supply–use** and **input–output** frameworks, consistent with NACE Rev. 2 activity groupings. It is used by the **iotables** package to order and label the **product** × **product** blocks of symmetric input–output tables (SIOTs) and their supply–use counterparts.

The **cpa2_1** vocabulary was obtained from the official Eurostat SKOS registry (https://dd.eionet.europa.eu/vocabulary/eurostat/cpa2_1/, filter: `conceptStatusInt = 128`, i.e. active concepts).

From the full CPA 2.1 hierarchy, this dataset retains:

- **0-digit** sections (e.g. "CPA_L", "CPA_0"),

- **1-digit** aggregates,
- **2-digit** divisions, and
- published **summary** and **extension** items.

These levels correspond to the structure of Eurostat's aggregated supply-use tables and are suitable for analytical applications such as cultural, tourism, or environmental satellite accounts.

All entries are marked with status "**valid**", reflecting their current publication state in the Eurostat metadata repository.

Source

Eurostat metadata registry (DD EIONET vocabulary: https://dd.eionet.europa.eu/vocabulary/eurostat/cpa2_1/)

See Also

Other Eurostat vocabularies included in the package: [ind_use](#), [ind_ava](#), [prd_use](#), [prd_ava](#).

Examples

```
data(cpa2_1)
```

croatia_2010_1700	<i>Input-output table for Croatia, 2010</i>
-------------------	---

Description

Symmetric input-output table at basic prices (product $\times$ product). Original code: **1700**. Values are expressed in thousand kunas (T_NAC).

Usage

```
croatia_2010_1700
```

Format

A data frame with 13 variables:

t_rows2 Technology codes in row names, following Eurostat conventions.

t_rows2_lab Longer labels for **t_rows2**.

t_cols2 Technology codes in column names, following Eurostat conventions.

t_cols2_lab Longer labels for **t_cols2**.

iotables_col Standardized column labels for easier reading.

col_order Ordering index to keep the matrix legible.

row_order Ordering index to keep the matrix legible.

iotables_row Standardized row labels for easier reading.
unit Unit of measure. Here: thousand national currency units (kunas).
geo ISO/Eurostat country code for Croatia.
geo_lab ISO/Eurostat country name, "Croatia".
time Date of the SIOT.
values Observed values in thousand kunas.

Source

Državni zavod za statistiku

See Also

Other Croatia 2010 datasets: [croatia_2010_1800](#), [croatia_2010_1900](#), [croatia_employment_2013](#), [croatia_employment_aggregation](#), [primary_inputs](#)

croatia_2010_1800	<i>Input-output table for Croatia, 2010 (domestic production)</i>
-------------------	---

Description

Symmetric input-output table (SIOT) for domestic production (product $\times$ product), code **1800**.

Values are expressed in thousand kunas (T_NAC).

Usage

```
croatia_2010_1800
```

Format

A data frame with 13 variables:

t_rows2 Technology codes in row names, following Eurostat conventions.
t_rows2_lab Longer labels for **t_rows2**.
values Actual values of the table in thousand kunas.
t_cols2 Column labels, following Eurostat conventions. A CPA_ suffix was added to original DZS column names.
t_cols2_lab Longer labels for **t_cols2**.
iotables_col Standardized **iotables** column labels for easier reading.
col_order Column ordering to keep the matrix legible.
iotables_row Standardized **iotables** row labels for easier reading.
row_order Row ordering to keep the matrix legible.
unit Different from Eurostat tables, in thousand national currency units.
geo ISO/Eurostat country code for Croatia.
geo_lab ISO/Eurostat country name, Croatia.
time Date of the SIOT.

Source

Državni zavod za statistiku

See Also

Other Croatia 2010 datasets: [croatia_2010_1700](#), [croatia_2010_1900](#), [croatia_employment_2013](#), [croatia_employment_aggregation](#), [primary_inputs](#)

croatia_2010_1900	<i>Input-output table for Croatia, 2010 (imports)</i>
-------------------	---

Description

Symmetric input-output table (SIOT) for imports (product $\times$ product), code **1900**.
 Values are expressed in thousand kunas (T_NAC).

Usage

```
croatia_2010_1900
```

Format

A data frame with 13 variables:

t_rows2 Technology codes in row names, following Eurostat conventions.
t_rows2_lab Longer labels for **t_rows2**.
values Actual values of the table in thousand kunas.
t_cols2 Column labels, following Eurostat conventions. A CPA_ suffix was added to original DZS column names.
t_cols2_lab Longer labels for **t_cols2**.
iotables_col Standardized **iotables** column labels for easier reading.
col_order Column ordering to keep the matrix legible.
iotables_row Standardized **iotables** row labels for easier reading.
row_order Row ordering to keep the matrix legible.
unit Different from Eurostat tables, in thousand national currency units.
geo ISO/Eurostat country code for Croatia.
geo_lab ISO/Eurostat country name, Croatia.
time Date of the SIOT.

Source

Državni zavod za statistiku

See Also

Other Croatia 2010 datasets: [croatia_2010_1700](#), [croatia_2010_1800](#), [croatia_employment_2013](#), [croatia_employment_aggregation](#), [primary_inputs](#)

`croatia_employment_2013`

Croatian employment data for 2013

Description

Aggregated employment statistics for Croatia in 2013, formatted to match the Eurostat standard symmetric input-output table (SIOT) structure.

Usage

```
data(croatia_employment_2013)
```

Format

A data frame with 107 observations and 3 variables:

code Short labels for industries or sectors.

iotables_row iotables-style row labels.

employment Employment in the sector (absolute values, not in thousands).

See Also

Other Croatia 2010 datasets: [croatia_2010_1700](#), [croatia_2010_1800](#), [croatia_2010_1900](#), [croatia_employment_aggregation](#), [primary_inputs](#)

`croatia_employment_aggregation`

Aggregation Table for Croatian Employment Statistics

Description

A mapping table to aggregate detailed Croatian employment statistics into the Croatian (EU-standard) symmetric input-output table (SIOT) format.

Usage

```
croatia_employment_aggregation
```

Format

A data frame with 105 rows (including empty rows) and 2 variables:

employment_label Labels from the DZS (Croatian Bureau of Statistics) English-language export.

t_cols2 Labels used in EU/DZS symmetric input-output tables (SIOTs).

Details

This dataset provides a concordance between Croatian employment classifications and the EU/DZS SIOT framework, enabling consistent integration of employment data into input-output analysis.

See Also

Other Croatia 2010 datasets: [croatia_2010_1700](#), [croatia_2010_1800](#), [croatia_2010_1900](#), [croatia_employment_2013](#), [primary_inputs](#)

`direct_effects_create`

Create direct effects

Description

The function creates the effects.

Usage

```
direct_effects_create(input_requirements, inverse, digits = NULL)
```

Arguments

`input_requirements`

A matrix or vector created by [input_indicator_create](#)

`inverse`

A Leontief-inverse created by [leontief_inverse_create](#).

`digits`

Rounding digits, defaults to NULL, in which case no rounding takes place.

Value

A data.frame containing the direct effects and the necessary metadata to sort them or join them with other matrixes.

See Also

Other indicator functions: [coefficient_matrix_create\(\)](#)

Examples

```
n1 <- netherlands_2000

input_coeff_n1 <- input_coefficient_matrix_create(
  data_table = netherlands_2000,
  households = FALSE
)

compensation_indicator <- input_indicator_create(netherlands_2000, "compensation_employees")
```

```
I_nl <- leontief_inverse_create(input_coeff_nl)

direct_effects_create(
  input_requirements = compensation_indicator,
  inverse = I_nl
)
```

employment_get	<i>Get Eurostat employment data for SIOTs</i>
----------------	---

Description

Download employment data from Eurostat (dataset [lfsq_egan22d](#)) and arrange it to match 64×64 symmetric input-output tables (SIOTs).

Usage

```
employment_get(
  geo,
  year = "2010",
  sex = "Total",
  age = "Y_GE15",
  labelling = "iotables",
  data_directory = NULL,
  force_download = FALSE
)
```

Arguments

geo	A two-letter country code (Eurostat style). "GB" and "GR" are automatically converted to "UK" and "EL".
year	Year of employment data (≥ 2008 , when NACE Rev. 2 was introduced).
sex	Employment by sex. Defaults to "Total". Alternatives are "Female"/"F", "Male"/"M". Case-insensitive.
age	Eurostat age code. Defaults to "Y_GE15". Any valid Eurostat code may be supplied (see Eurostat metadata).
labelling	Controls output row/column labelling: • "iotables": iotables manual-style labels • "prod_na": product $\times$ product (CPA codes) • "induse": industry $\times$ industry (NACE codes)
data_directory	Optional path to save/load pre-processed employment data (.rds files). If NULL, only downloads are used.
force_download	Logical. If TRUE, forces a fresh Eurostat download even if local cache files exist.

Details

- Currently implemented only for product $\times$ product tables.
- Country codes are harmonized: "GB" $\rightarrow$ "UK", "GR" $\rightarrow$ "EL".
- Sex is normalized internally to Eurostat codes "T", "F", "M".
- Results are cached as `.rds` files in `data_directory` if supplied.
- An imputed rent column (L68A/CPA_L68A) with zero is always added.

Value

A one-row `data.frame` containing employment input values aligned with the chosen SIOT labelling, including an imputed rent column set to 0.

Source

Eurostat dataset `lfsq_egan22d`

See Also

Other import functions: `iotables_download()`, `iotables_download_old()`, `iotables_read_tempdir()`

Examples

```
## Not run:
employment <- employment_get(
  geo = "CZ",
  year = "2010",
  sex = "Total",
  age = "Y_GE15",
  data_directory = NULL,
  force_download = TRUE
)

## End(Not run)
```

employment_metadata	<i>Employment Metadata</i>
---------------------	----------------------------

Description

A reference dataset linking Eurostat national accounts vocabulary with employment statistics data.

Usage

```
employment_metadata
```

Format

- A data frame with 6 variables:
- emp_code** Codes used in the employment statistics.
 - code** Eurostat labels for SIOTs corresponding to **emp_code**.
 - label** Eurostat label descriptions for SIOTs corresponding to **emp_code**.
 - variable** Eurostat vocabulary source (e.g., **t_rows**, **t_cols**, **prod_na**, **induse**).
 - group** Grouping of accounts (different from Eurostat tables), in thousands of national currency units.
 - iotables_label** Custom machine-readable snake_case variable names.

Details

This dataset provides a mapping between employment statistics codes and the vocabulary used in Eurostat input-output tables, ensuring compatibility when joining employment and national accounts data.

See Also

Other metadata datasets: [metadata](#), [metadata_uk_2010](#)

<code>empty_remove</code>	<i>Remove Empty Rows and Columns Symmetrically</i>
---------------------------	--

Description

Remove columns and their corresponding rows if they contain only zeros or missing values. This ensures that the resulting table remains symmetric (same dimensions in rows and columns).

Usage

```
empty_remove(data_table)
```

Arguments

- data_table** A symmetric input-output table, or the symmetric quadrant of a use or supply table.

Details

The function first identifies columns that contain only zeros or missing values, then removes both those columns and the rows with matching labels in the first (key) column. A message is printed listing the removed columns.

Value

A `data.frame` (or tibble) with a key column and symmetric matrix, after removing all-zero (or all-missing) columns and their corresponding rows.

See Also

Other iotables processing functions: `conforming_vector_create()`, `household_column_find()`, `household_column_get()`, `iotable_year_get()`, `key_column_create()`, `matrix_round()`, `output_get()`, `primary_input_get()`, `rows_add()`, `supplementary_add()`, `total_tax_add()`, `vector_transpose_longer()`, `vector_transpose_wider()`

Examples

```
# Example using the built-in demo table
test_table <- input_coefficient_matrix_create(
  iotable_get(source = "germany_1995")
)

# Set one column to zero, then remove it
test_table[, 2] <- 0
empty_remove(test_table)
```

equation_solve

Solve a basic matrix equation

Description

Match a left-hand side (LHS) vector to a Leontief inverse by column names and compute the matrix product $\text{LHS} \times \text{Im}$.

Usage

```
equation_solve(LHS = NULL, Im = NULL)
```

Arguments

LHS	A one-row data frame (or matrix) with a key column first and numeric columns named to match the Leontief inverse.
Im	A Leontief inverse with a key column first and a square numeric block whose column names match the LHS numeric names.

Details

This helper is used by higher-level wrappers such as `multiplier_create()`. It assumes both inputs have a first key column, followed by numeric columns whose names define the alignment. The function multiplies the numeric row of **LHS** by the numeric block of **Im** after a basic conformity check.

Value

A numeric $1 \times N$ matrix containing the solution $LHS \times Im$.

Examples

```
Im <- data.frame(
  a = c("row1", "row2"),
  b = c(1, 1),
  c = c(2, 0)
)
LHS <- data.frame(
  a = "lhs",
  b = 1,
  c = 0.5
)
equation_solve(Im = Im, LHS = LHS)
```

forward_linkages	<i>Forward linkages</i>
------------------	-------------------------

Description

Forward linkages capture how the increased output of a sector provides additional inputs to other sectors, enabling them to expand production.

Usage

```
forward_linkages(output_coefficient_matrix, digits = NULL)
```

Arguments

output_coefficient_matrix	An output coefficient matrix created with <code>output_coefficient_matrix_create()</code> .
digits	Integer. Number of decimals for rounding. Defaults to NULL (no rounding).

Details

Defined as the row sums of the Ghosh inverse, in line with the Eurostat Manual of Supply, Use and Input-Output Tables (pp. 506–507) and the United Nations *Handbook on Supply and Use Tables and Input-Output Tables with Extensions and Applications* (p. 637).

Value

A `data.frame` with two columns:

- The metadata column from the input matrix (sector/product names)
- `forward_linkages`: the forward linkage indicator values

See Also

Other linkage functions: [backward_linkages\(\)](#)

Examples

```
data_table <- iotable_get()

de_out <- output_coefficient_matrix_create(
  data_table, "tfu",
  digits = 4
)

forward_linkages(
  output_coefficient_matrix = de_out,
  digits = 4
)
```

germany_1995	<i>Germany 1995 symmetric input-output table (ESA 2010 codes)</i>
--------------	---

Description

Reproduction of Table 15.4 in the Eurostat Manual (*Input-output table of domestic output at basic prices, Version A*). This is a small, well-documented benchmark dataset that accompanies the **iotables** package. It is reformatted into the same long structure as Eurostat warehouse SIOs, so that functions and tests can work identically on this example and on real Eurostat downloads.

Usage

```
germany_1995
```

Format

A data frame with 247 rows and 11 columns:

prod_na Row code (ESA 2010 / CPA aggregate).

prod_na_lab Row label, long description.

iotables_row Row identifier used internally.

iotables_col Column identifier (factor with 13 levels).

values Cell value, in millions of euros (integer).

induse Column code (ESA 2010 / CPA aggregate or national accounts item).

geo Country code, always "DE".

geo_lab Country name, "Germany".

time Reference year, as a Date ("1995-01-01").

unit Unit code, "MIO_EUR".

unit_lab Unit label, "Million euro".

Details

The values come from Beutel (2008), *Eurostat Manual of Supply, Use and Input–Output Tables*, Table 15.4. Labels and codes follow ESA 2010 conventions (e.g. CPA_A, CPA_B–E, P3_S14), allowing direct comparison with modern Eurostat releases.

This dataset underpins many unit tests in **iotables**: multipliers, coefficients, and linkage indices are validated against the published benchmark. Because it is small (247 rows), it is also used in vignettes and examples to demonstrate workflows.

Source

Beutel, J. (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*, Table 15.4. Luxembourg: Office for Official Publications of the European Communities.

See Also

`iotable_get()` for extracting comparable tables from Eurostat.

Examples

```
data(germany_1995)
head(germany_1995)
# Verify against the Eurostat manual:
subset(germany_1995, prod_na == "CPA_A" & iotables_col == "agriculture_group")
```

germany_airpol	<i>Air pollution table for Germany, 1995</i>
----------------	--

Description

Air pollution values for validation and cross-checking with the Eurostat Manual.

Usage

```
germany_airpol
```

Format

A data frame with 72 observations and 4 variables:

airpol Abbreviation of the air pollutant.

induse Column labels, following Eurostat conventions with minor differences.

iotables_col Column labels using **iotables** abbreviations.

value Values in thousand tons.

Details

This dataset is provided for testing purposes. Labels were slightly adjusted to reflect the transition from ESA95 to ESA2010 vocabulary since the publication of the Eurostat Manual.

Source

Eurostat (2008). *Eurostat Manual of Supply, Use and Input-Output Tables*, p. 482. Available at <https://ec.europa.eu/eurostat/documents/3859598/5902113/KS-RA-07-013-EN.PDF>

See Also

Other validation datasets: [netherlands_2000](#), [uk_2010_data](#), [uk_test_results](#)

`ghosh_inverse_create` *Create Ghosh inverse from output coefficients*

Description

Compute the Ghosh inverse from an output-coefficient matrix.

Usage

```
ghosh_inverse_create(output_coefficients_matrix, digits = NULL)
```

Arguments

output_coefficients_matrix A technology-coefficient matrix created by [output_coefficient_matrix_create\(\)](#).

digits Optional integer precision for rounding. Default NULL (no rounding).

Details

The Ghosh inverse is defined as $G = (I - B)^{-1}$, where B is the output-coefficient matrix created by [output_coefficient_matrix_create\(\)](#).

See the United Nations *Handbook on Supply and Use Tables and Input-Output Tables with Extensions and Applications* (2018, Rev. 1), pp. 622–639 ([PDF](#)).

For the analogous inverse based on input coefficients, see [leontief_inverse_create\(\)](#).

Value

A `data.frame` with the original key column and the Ghosh inverse in the remaining columns. If `digits` is provided, values are rounded.

See Also

Other analytic object functions: [input_flow_get\(\)](#), [leontief_inverse_create\(\)](#), [leontief_matrix_create\(\)](#), [output_coefficient_matrix_create\(\)](#)

Examples

```
# Minimal example
om <- output_coefficient_matrix_create(iotable_get())
ghosh_inverse_create(om)

# Using the Germany 1995 benchmark table (Eurostat manual)
# data(germany_1995)
# om_de <- output_coefficient_matrix_create(germany_1995)
# ghosh_inverse_create(om_de)
```

household_column_find

Find Household Expenditure Column

Description

Identify the column position corresponding to final household expenditure in a symmetric input-output table or related table.

Usage

```
household_column_find(data_table)
```

Arguments

data_table A symmetric input-output table, a use table, or a supply table.

Details

The function searches column names case-insensitively. It first looks for exact matches among the following alternatives:

- "households"
- "p3_s14"
- "final_consumption_households"
- "final_consumption_household"
- "consumption_expenditure_household"
- "consumption_expenditure_households"

If none of these are found, it falls back to any column name that contains "households".

Value

An integer vector giving the position(s) of household expenditure columns. Returns `NULL` if none are found.

See Also

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
# German SIOT includes a household final consumption column
household_column_find(iotable_get(source = "germany_1995"))

# Custom example
df <- data.frame(
  sector = c("A", "B"),
  households = c(100, 200)
)
household_column_find(df)
```

`household_column_get` *Return Final Household Expenditure*

Description

Extracts the column of final household expenditure from a symmetric input-output table, a use table, or a supply table. If no household expenditure column is detected, returns NULL.

Usage

```
household_column_get(data_table)
```

Arguments

`data_table` A symmetric input-output table, a use table, or a supply table.

Value

A tibble/data frame with the key column and the household expenditure column. Returns NULL if no household column is found.

See Also

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_find\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
household_column_get(iotable_get(source = "germany_1995"))
```

ind_ava	<i>Eurostat Industry Supply, Adjustments and Value Added Vocabulary (ind_ava)</i>
---------	---

Description

A reference codelist used in Eurostat’s national accounts framework for the **industry** × **industry** (NACE-based) symmetric input–output tables (SIOTs). This vocabulary enumerates the **rows** of the table, covering both the intermediate (Quadrant 1) and value-added (Quadrant 3) blocks, and includes the control total “*Total supply at basic prices*” published by Eurostat.

Usage

ind_ava

Format

A tibble (data frame) with 10 variables:

id Canonical Eurostat concept identifier.

label Official Eurostat label (preferred English term).

status "Valid" or "Adopted from CPA2_1".

status_modified Date of the most recent status update.

notation Alternate human-readable code or short form (e.g. CPA_A01).

group Optional grouping of related industries or aggregates.

quadrant Integer designating the analytical block: 10 = Intermediate (technology block, Quadrant 1); 20 = Value added (primary inputs, Quadrant 3); 30 = Control total(s).

numeric_order Ordinal position key for sorting within quadrant.

iotables_label Simplified legacy label used in the [iotables](#) package.

block Semantic block category inferred from **quadrant** (*intermediate*, *primary_inputs*, *control_total*, *extension*).

uri Stable URI linking to the Eurostat SKOS concept at <https://dd.eionet.europa.eu/>.

Details

The **ind_ava** vocabulary aligns with the official Eurostat SDMX codelist “*Industries, adjustments and value added*” and is used by the [iotables](#) package to order the rows of industry-by-industry tables (datasets such as **naio_10_cp1750**).

Codes correspond to NACE Rev. 2 divisions and ESA 2010 accounting aggregates. Each entry includes its quadrant classification and an ordinal numeric order to preserve correct row sequencing when reshaping long-form data.

The dataset retains the published control total (“Total supply at basic prices”) for consistency with Eurostat’s metadata, but this row is normally excluded from analytical matrix operations.

Because the Eurostat SIOT datasets sometimes contain more detailed industry breakup than their vocabulary, we adopted industry coding from [cpa2_1](#). These codes are marked the "Adopted from CPA2_1" status.

Source

Eurostat metadata registry (DD EIONET vocabulary: https://dd.eionet.europa.eu/vocabulary/eurostat/ind_ava/)

See Also

Other Eurostat vocabularies included in the package: [ind_use](#), [prd_use](#), [prd_ava](#), [cpa2_1](#).

Examples

```
data(ind_ava)
```

ind_use	<i>Eurostat Industry Uses Vocabulary (ind_use)</i>
---------	--

Description

A reference codelist used in Eurostat's national accounts framework for **industry** × **industry** (NACE-based) symmetric input-output tables (SIOTs). This vocabulary enumerates the **columns** of the table, covering the intermediate-use block (Quadrant 1), the value-added block (Quadrant 3), and the final-use and balancing items (Quadrant 2/50).

Usage

```
ind_use
```

Format

A tibble (data frame) with 10 variables:

id Canonical Eurostat concept identifier.

label Official Eurostat label (preferred English term).

status "Valid" or "Adopted from CPA2_1".

status_modified Date of last status modification.

notation Alternate human-readable code or short form (e.g. CPA_A01).

quadrant Integer designating the analytical block: 10 = Intermediate (technology block, Q1); 20 = Primary inputs / GVA (Q3); 30 = Final use / control totals (Q2); 50 = Extensions / diagnostics.

numeric_order Ordinal key for ordering columns within quadrants.

iotables_label Simplified legacy label used in the [iotables](#) package.

block Semantic block category inferred from **quadrant** (**intermediate**, **primary_inputs**, **final_use**, **extension**).

uri Stable URI linking to the Eurostat SKOS concept at <https://dd.eionet.europa.eu/>.

Details

The `ind_use` vocabulary corresponds to the official Eurostat SDMX codelist “*Industry uses*”. It is used by the `iotables` package to order the **column dimension** of industry-by-industry tables (datasets such as `naio_10_cp1750`).

Codes follow **NACE Rev. 2** for industries and **ESA 2010** for accounting aggregates. Each entry includes its quadrant classification and an ordinal numeric sequence for reproducible column ordering when reshaping long-form data.

The vocabulary also includes recent Eurostat additions introduced in 2023 (*ITTM* – International trade and transport margins, *MCH* – Merchanting, and *OP* – Purchases by residents abroad). These appear in the final-use block (`quadrant = 30`) and are retained for completeness and balancing, though they are not part of the analytical technology matrix.

Because the Eurostat SIOT datasets sometimes contain more detailed industry breakup than their vocabulary, we adopted industry coding from `cpa2_1`. These codes are marked the “Adopted from CPA2_1” status.

Source

Eurostat metadata registry (https://dd.eionet.europa.eu/vocabulary/eurostat/ind_use/)

See Also

Related Eurostat vocabularies: `ind_ava`, `prd_ava`, `prd_use`, `cpa2_1`.

Examples

```
data(ind_use)
dplyr::count(ind_use, quadrant)
```

```
indirect_effects_create
      Create indirect effects
```

Description

Compute an indirect-effects vector from an input requirement indicator and a Leontief inverse.

Usage

```
indirect_effects_create(input_requirements, inverse, digits = NULL)
```

Arguments

<code>input_requirements</code>	A data frame or matrix produced by <code>input_indicator_create()</code> , with a key column first and numeric columns thereafter.
<code>inverse</code>	A Leontief inverse created by <code>leontief_inverse_create()</code> , with a key column first and numeric columns thereafter.
<code>digits</code>	Integer number of decimal places for rounding. Defaults to NULL (no rounding).

Value

A `data.frame` containing the indirect effects and the first (key) column to allow sorting or joins with other tables.

Examples

```
data(netherlands_2000)

input_coeff_nl <- input_coefficient_matrix_create(
  data_table = netherlands_2000,
  households = FALSE
)

compensation_indicator <- input_indicator_create(
  netherlands_2000, "compensation_employees"
)

I_nl <- leontief_inverse_create(input_coeff_nl)

indirect_effects_create(
  input_requirements = compensation_indicator,
  inverse = I_nl
)
```

`input_coefficient_matrix_create`

Create an input coefficient matrix

Description

Computes the **input (technical) coefficient matrix** from a symmetric input–output table or use table. Each element expresses the share of a product’s output used as input in another production activity.

Usage

```
input_coefficient_matrix_create(data_table, households = FALSE, digits = NULL)
```

Arguments

<code>data_table</code>	A symmetric input–output or use table retrieved by <code>iotable_get()</code> .
<code>households</code>	Logical; include the household column if available. Default <code>FALSE</code> .
<code>digits</code>	Optional integer precision for rounding the coefficients. Default <code>NULL</code> (no rounding).

Details

This is a wrapper around `coefficient_matrix_create()` with `total = "output"` and `return_part = "products"`. The resulting matrix (A) satisfies $(A_{ij} = z_{ij} / x_j)$, where (z_{ij}) is intermediate use of product i by industry j and (x_j) is total output. Terminology follows the [Eurostat Manual of Supply, Use and Input–Output Tables](#), where this is also called the *technological coefficients matrix*.

Value

A `data.frame` containing the input coefficient matrix (products $\times$ products), with the key (row label) as the first column. Rows and columns labelled `"cpa_total"` or `"total"` are removed.

References

Beutel, J. (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*. Luxembourg: Office for Official Publications of the European Communities, Table 15.6 (page 485). Validation against this reference confirms that `input_coefficient_matrix_create()` reproduces the standard Eurostat input–output coefficients for Germany 1995.

United Nations et al. (2009). *System of National Accounts 2008*, §§14.90–14.91.

Examples

```
cm <- input_coefficient_matrix_create(
  iotable_get(source = "germany_1995"),
  digits = 4
)
head(cm)
```

<code>input_flow_get</code>	<i>Create an inter-industry (intermediate-use) matrix</i>
-----------------------------	---

Description

Return the Quadrant I block (intermediate consumption) of a symmetric input–output table. Optionally append the final household consumption column for Type-II modelling.

Usage

```
input_flow_get(data_table, empty_remove = FALSE, households = TRUE)
```

Arguments

<code>data_table</code>	A symmetric input-output table (product-by-product or industry-by-industry) obtained via <code>iotable_get()</code> .
<code>empty_remove</code>	Logical. Reserved; currently ignored (no effect). Default <code>FALSE</code> .
<code>households</code>	Logical. If <code>TRUE</code> , append the <code>final_consumption_households</code> column. Default <code>TRUE</code> .

Details

In the Eurostat framework, the Quadrant I block shows *intermediate consumption* by industry (columns) and product (rows), valued at purchasers' prices. Final household consumption belongs to the final uses block (not Quadrant I); when `households = TRUE`, that column is appended for convenience in Type-II analyses that endogenise private consumption. See the *Eurostat Manual of Supply, Use and Input-Output Tables* for the quadrant layout and definitions.

Value

A data frame with the key column and the Quadrant I block; if `households = TRUE`, the household final consumption column is appended.

See Also

`input_coefficient_matrix_create()`, `leontief_inverse_create()`

Other analytic object functions: `ghosh_inverse_create()`, `leontief_inverse_create()`, `leontief_matrix_create()`, `output_coefficient_matrix_create()`

Examples

```
# Basic extraction (Quadrant I + households column)
x <- input_flow_get(
  data_table = iotable_get(),
  empty_remove = FALSE,
  households = TRUE
)

# Quadrant I only (no households column)
y <- input_flow_get(
  data_table = iotable_get(),
  empty_remove = FALSE,
  households = FALSE
)
```

```
input_indicator_create
```

Create input indicator(s)

Description

Compute input indicators (e.g., GVA, compensation of employees) by selecting specific input rows from the input-coefficient matrix.

Usage

```
input_indicator_create(
  data_table,
  input_row = c("gva_bp", "net_tax_production"),
  digits = NULL,
  households = FALSE,
  indicator_names = NULL
)
```

Arguments

<code>data_table</code>	A symmetric input–output table, use table, margins, or tax table retrieved by <code>iotable_get()</code> .
<code>input_row</code>	Character vector of input row names to extract (e.g., "gva", "compensation_employees"). Matching is case-insensitive.
<code>digits</code>	Integer number of decimal places for rounding. Default <code>NULL</code> (no rounding).
<code>households</code>	Logical; include a households column if available. Default <code>FALSE</code> .
<code>indicator_names</code>	Optional character vector of names for the returned indicators. If <code>NULL</code> , names are taken from the key column in the selected rows of the coefficient matrix and suffixed with <code>"_indicator"</code> .

Details

Let A be the input-coefficient matrix (rows are inputs, columns are products/industries). An *input indicator* for a given input row r is simply the row $A_{r..}$. These indicators are used in Beutel (2012) and the Eurostat *Manual of Supply, Use and Input-Output Tables* (e.g., pp. 495–498) to derive effects and multipliers.

Internally, the function builds A via `coefficient_matrix_create()`, then keeps only the requested input rows and renames the key column to `*_indicator`. Optional rounding is applied to numeric columns.

Value

A `data.frame` whose first column is a key, followed by the selected input-indicator rows as numeric columns.

Examples

```
input_indicator_create(
  data_table = iotable_get(),
  input_row = c("gva", "compensation_employees"),
  digits = 4,
  indicator_names = c("GVA indicator", "Income indicator")
)

# Beutel/Eurostat example: GVA indicator (cf. Manual, ~p. 498)
ii <- input_indicator_create(
  data_table = iotable_get(),
  input_row = "gva",
  digits = 4
)
head(ii)
```

```
input_multipliers_create
      Create input multipliers
```

Description

The function creates the multipliers (direct + indirect effects).

Usage

```
input_multipliers_create(
  input_requirements,
  Im,
  multiplier_name = NULL,
  digits = NULL
)
```

Arguments

input_requirements	A matrix or vector created by input_indicator_create
Im	A Leontief-inverse created by leontief_inverse_create .
multiplier_name	An optional name to be placed in the key column of the multiplier. Defaults to NULL.
digits	Rounding digits, defaults to NULL, in which case no rounding takes place. Rounding is important if you replicate examples from the literature, rounding differences can add up to visible differences in matrix equations.

Value

A data frame with the vector of multipliers and the an auxiliary metadata column, containing an automatically given row identifier (for joining with other matrixes) which can be overruled with setting `multiplier_name`.

See Also

Other multiplier functions: `output_multiplier_create()`

Examples

```
nl <- netherlands_2000

input_coeff_nl <- input_coefficient_matrix_create(
  data_table = netherlands_2000,
  households = FALSE
)

compensation_indicator <- input_indicator_create(netherlands_2000, "compensation_employees")

I_nl <- leontief_inverse_create(input_coeff_nl)

input_multipliers_create(
  input_requirements = compensation_indicator,
  Im = I_nl
)
```

`iotable_get`

Get a single input-output table from built-in datasets

Description

Filter and reshape one IO/SUT table from a bulk dataset (typically a Eurostat download). In most workflows you will call this function rather than `iotables_download()`, which it invokes as needed.

Usage

```
iotable_get(
  labelled_io_data = NULL,
  source = "germany_1995",
  geo = "DE",
  year = 1990,
  unit = "MIO_EUR",
  stk_flow = "DOM",
  labelling = "iotables",
  data_directory = NULL,
  force_download = TRUE
)
```

Arguments

<code>labelled_io_data</code>	Optional nested bulk data as returned by <code>iotables_download()</code> . If <code>NULL</code> (default), data are retrieved from cache or downloaded.
<code>source</code>	Data source code (see list above).
<code>geo</code>	Country code or name, e.g. "SK" or "Slovakia".
<code>year</code>	Numeric year. Defaults to 1990 for <code>germany_1995</code> .
<code>unit</code>	Currency unit, usually "MIO_NAC" or "MIO_EUR".
<code>stk_flow</code>	Stock/flow: "DOM", "IMP", or "TOTAL". For margins/taxes (<code>cp1620</code> , <code>cp1630</code> and <code>pyp</code> variants) only "TOTAL" is used; other inputs are coerced with a warning.
<code>labelling</code>	Column naming scheme: "iotables" (default) for consistent names; "short" for original short codes; "eurostat" is treated as "short".
<code>data_directory</code>	Optional directory to save the processed wide table (RDS). If <code>NULL</code> , nothing is saved.
<code>force_download</code>	Logical. If <code>TRUE</code> , force a fresh download when <code>labelled_io_data</code> is not supplied. Defaults to <code>TRUE</code> .

Details

The Eurostat bulk tables arrive in long form and are not ordered for matrix algebra. This function selects the requested country (`geo`), year, unit and stock/flow (`stk_flow`), joins `iotables` metadata for consistent row/column labelling, and returns a **wide** table ready for analysis.

Supported sources include:

- `naio_10_cp1700` — symmetric IO, basic prices (prod × prod)
- `naio_10_pyp1700` — previous years' prices
- `naio_10_cp1750` — symmetric IO, basic prices (ind × ind)
- `naio_10_pyp1750` — previous years' prices
- `naio_10_cp15` — supply at basic prices incl. margins/taxes
- `naio_10_cp16` — use at purchasers' prices
- `naio_10_cp1610` — use at basic prices
- `naio_10_cp1620` — trade & transport margins (basic prices)
- `naio_10_cp1630` — taxes less subsidies on products (basic prices)
- `naio_10_pyp*` — corresponding previous years' prices variants
- `germany_1995` — packaged Beutel example
- `croatia_2010_1700/1800/1900` — packaged examples
- `uk_2010_*` — packaged UK 2010 variants

Value

A **wide data.frame** representing the selected IO table, with a key column followed by ordered numeric columns.

Note

This is a wrapper around the legacy getter function now only used for built-in datasets.

See Also

Other iotables import functions: `iotable_get_builtin()`

Examples

```
germany_table <- iotable_get(
  source = "germany_1995",
  geo = "DE", year = 1990, unit = "MIO_EUR",
  labelling = "iotables"
)
```

`iotable_get_builtin` *Get a single input-output table from built-in datasets*

Description

Filter and reshape one IO/SUT table from a bulk dataset.

Usage

```
iotable_get_builtin(
  labelled_io_data = NULL,
  source = "germany_1995",
  geo = "DE",
  year = 1990,
  unit = "MIO_EUR",
  stk_flow = "DOM",
  labelling = "iotables",
  data_directory = NULL,
  force_download = TRUE
)
```

Arguments

<code>labelled_io_data</code>	Optional nested bulk data as returned by <code>iotables_download()</code> . If <code>NULL</code> (default), data are retrieved from cache or downloaded.
<code>source</code>	Data source code (see list above).

<code>geo</code>	Country code or name, e.g. "SK" or "Slovakia".
<code>year</code>	Numeric year. Defaults to 1990 for <code>germany_1995</code> .
<code>unit</code>	Currency unit, usually "MIO_NAC" or "MIO_EUR".
<code>stk_flow</code>	Stock/flow: "DOM", "IMP", or "TOTAL". For margins/taxes (<code>cp1620</code> , <code>cp1630</code> and <code>pyp</code> variants) only "TOTAL" is used; other inputs are coerced with a warning.
<code>labelling</code>	Column naming scheme: "iotables" (default) for consistent names; "short" for original short codes; "eurostat" is treated as "short".
<code>data_directory</code>	Optional directory to save the processed wide table (RDS). If NULL, nothing is saved.
<code>force_download</code>	Logical. If TRUE, force a fresh download when <code>labelled_io_data</code> is not supplied. Defaults to TRUE.

Details

The Eurostat bulk tables arrive in long form and are not ordered for matrix algebra. This function selects the requested country (`geo`), year, unit and stock/flow (`stk_flow`), joins iotables metadata for consistent row/column labelling, and returns a **wide** table ready for analysis.

Supported sources include:

- `naio_10_cp1700` — symmetric IO, basic prices (prod × prod)
- `naio_10_pyp1700` — previous years' prices
- `naio_10_cp1750` — symmetric IO, basic prices (ind × ind)
- `naio_10_pyp1750` — previous years' prices
- `naio_10_cp15` — supply at basic prices incl. margins/taxes
- `naio_10_cp16` — use at purchasers' prices
- `naio_10_cp1610` — use at basic prices
- `naio_10_cp1620` — trade & transport margins (basic prices)
- `naio_10_cp1630` — taxes less subsidies on products (basic prices)
- `naio_10_pyp*` — corresponding previous years' prices variants
- `germany_1995` — packaged Beutel example
- `croatia_2010_1700/1800/1900` — packaged examples
- `uk_2010_*` — packaged UK 2010 variants

Value

A **wide data.frame** representing the selected IO table, with a key column followed by ordered numeric columns.

Note

This is a legacy downloader that handles Eurostat datasets, too, but with inefficient cache management.

See Also

Other iotables import functions: `iortable_get()`

Examples

```
germany_table <- iortable_get(
  source = "germany_1995",
  geo = "DE", year = 1990, unit = "MIO_EUR",
  labelling = "iotables"
)
```

`iortable_get_eurostat` *Get a single Eurostat input-output table*

Description

Filters and reshapes one IO/SUT table from a bulk Eurostat dataset. This version excludes all built-in datasets (e.g. Germany 1995, Croatia 2010, UK 2010) which are retrieved with the `iortable_get()` function.

Usage

```
iortable_get_eurostat(
  source = "naio_10_cp1700",
  geo = "IT",
  year = 2022,
  unit = "MIO_EUR",
  stk_flow = "TOTAL",
  labelling = "iotables",
  data_directory = NULL,
  force_download = TRUE,
  labelled_io_data = NULL
)
```

Arguments

<code>source</code>	Data source code (see list above).
<code>geo</code>	Country code or name, e.g. "SK" or "Slovakia".
<code>year</code>	Numeric year. Defaults to 1990 for <code>germany_1995</code> .
<code>unit</code>	Currency unit, usually "MIO_NAC" or "MIO_EUR".
<code>stk_flow</code>	Stock/flow: "DOM", "IMP", or "TOTAL". For margins/taxes (<code>cp1620</code> , <code>cp1630</code> and <code>pyp</code> variants) only "TOTAL" is used; other inputs are coerced with a warning.
<code>labelling</code>	Column naming scheme: "iotables" (default) for consistent names; "short" for original short codes; "eurostat" is treated as "short".

<code>data_directory</code>	Optional directory to save the processed wide table (RDS). If NULL, nothing is saved.
<code>force_download</code>	Logical. If TRUE, force a fresh download when <code>labelled_io_data</code> is not supplied. Defaults to TRUE.
<code>labelled_io_data</code>	Optional nested bulk data as returned by <code>iotables_download()</code> . If NULL (default), data are retrieved from cache or downloaded.

Details

The Eurostat bulk tables arrive in long form and are not ordered for matrix algebra. This function selects the requested country (`geo`), year, unit and stock/flow (`stk_flow`), joins iotables metadata for consistent row/column labelling, and returns a **wide** table ready for analysis.

Value

A **wide** `data.frame` representing the selected IO table.

<code>iotable_year_get</code>	<i>Get available years from bulk IO tables</i>
-------------------------------	--

Description

Query which years are available for a given Eurostat IO product, country (`geo`), and currency unit in a bulk download.

Usage

```
iotable_year_get(
  labelled_io_data = NULL,
  source = "germany_1995",
  geo = "DE",
  unit = "MIO_EUR",
  time_unit = "year",
  stk_flow = "TOTAL",
  data_directory = NULL,
  force_download = TRUE
)
```

Arguments

<code>labelled_io_data</code>	Optional labelled IO data from <code>iotables_download()</code> . If supplied, avoids re-reading from disk.
<code>source</code>	Character. Eurostat product code (see Details).

<code>geo</code>	Country code or name (e.g. "SK" or "Slovakia").
<code>unit</code>	Currency unit. Defaults to "MIO_NAC" (millions of national currency). Alternative: "MIO_EUR".
<code>time_unit</code>	Return mode for time. "year" (default) returns numeric years; "time" returns a vector of dates.
<code>stk_flow</code>	Flow type. Defaults to "DOM" (domestic output). Alternatives: • "IMP" for imports • "TOTAL" for total output For sources "naio_10_cp1620" (margins) and "naio_10_cp1630" (taxes), only "TOTAL" is used.
<code>data_directory</code>	Optional path used with <code>iortable_get()</code> or <code>iotables_download()</code> to persist bulk data.
<code>force_download</code>	Logical. Defaults to TRUE. If FALSE, reuse an existing file in <code>data_directory</code> or <code>tempdir()</code> when available.

Details

This function is usually called indirectly via `iortable_get()`. You normally do not need to call `iotables_download()` yourself unless working with bulk Eurostat files.

Supported Eurostat products include (non-exhaustive):

- "naio_10_cp1700" — Symmetric IO table, basic prices (product $\times$ product)
- "naio_10_cp1750" — Symmetric IO table, basic prices (industry $\times$ industry)
- "naio_10_pyp1700" — Symmetric IO table (product $\times$ product), previous years' prices
- "naio_10_pyp1750" — Symmetric IO table (industry $\times$ industry), previous years' prices
- "naio_10_cp1620" / "naio_10_pyp1620" — Trade & transport margins
- "naio_10_cp1630" / "naio_10_pyp1630" — Taxes less subsidies on products

See the [Eurostat Symmetric Input–Output Tables](#) page.

Value

A numeric vector of years, or a date vector if `time_unit = "time"`.

See Also

Other iotables processing functions: `conforming_vector_create()`, `empty_remove()`, `household_column_find()`, `household_column_get()`, `key_column_create()`, `matrix_round()`, `output_get()`, `primary_input_get()`, `rows_add()`, `supplementary_add()`, `total_tax_add()`, `vector_transpose_longer()`, `vector_transpose_wider()`

Examples

```
germany_years <- iotable_year_get(
  source = "germany_1995", geo = "DE", unit = "MIO_EUR"
)
# Return as dates
germany_dates <- iotable_year_get(
  source = "germany_1995", geo = "DE",
  unit = "MIO_EUR", time_unit = "time"
)
```

iotables	<i>iotables: Reproducible Input-Output Analysis for Eurostat and National Data</i>
----------	--

Description

Tools for downloading, structuring, and analysing input-output and supply-use tables, with metadata aligned to Eurostat's ESA 2010 framework.

Author(s)

Maintainer: Daniel Antal <daniel.antal@dataobservatory.eu> ([ORCID](#))

Other contributors:

- Pyry Kantanen ([ORCID](#)) [contributor]

See Also

Useful links:

- <https://iotables.dataobservatory.eu/>
- <https://github.com/rOpenGov/iotables>
- Report bugs at <https://github.com/rOpenGov/iotables/issues>

iotables_download	<i>Download input-output tables (Eurostat only)</i>
-------------------	---

Description

Downloads and prepares symmetric input-output (IO) or supply-use tables directly from Eurostat. This modern version excludes all built-in datasets (`germany_1995`, `croatia_2010_*`, `uk_2010_*`) which are handled internally by `iotable_get()`.

Usage

```
iotables_download(
  source = "naio_10_cp1700",
  geo = NULL,
  year = NULL,
  unit = NULL,
  stk_flow = NULL,
  data_directory = NULL,
  force_download = FALSE
)
```

Arguments

source	Character. Eurostat dataset ID (e.g. "naio_10_cp1700").
geo	Country code (optional).
year	Numeric (optional).
unit	Character, usually "MIO_EUR" or "MIO_NAC".
stk_flow	Stock/flow indicator ("DOM", "IMP", "TOTAL"), optional.
data_directory	Directory to save processed data (default: <code>tempdir()</code>).
force_download	Logical, whether to force a new Eurostat download.

Details

Files are cached under `tempdir()` as RDS (e.g. "naio_10_cp1750_processed.rds"). The temporary directory is cleared when the R session ends. To persist downloads across sessions (recommended for analytics), supply **data_directory**.

Value

A **nested tibble** with metadata columns (**geo**, **year**, **unit**, etc.) and a list-column **data** containing tidy IO tables.

See Also

Other import functions: `employment_get()`, `iotables_download_old()`, `iotables_read_tempdir()`

`iotables_download_old`

Legacy downloader for input-output tables (Eurostat)

Description

Download standard input-output (IO) and related tables. At the moment, only Eurostat products are supported. You usually do not need to call this directly; `iotable_get()` will invoke it as needed and return a filtered, tidy table.

Usage

```
iotables_download_old(
  source = "naio_10_cp1700",
  data_directory = NULL,
  force_download = FALSE
)
```

Arguments

source	Character. The Eurostat product code (see above) or "uk_2010".
data_directory	Optional directory path where the processed nested tables will be saved as "<source>_processed.rds". If NULL (default), results are saved to <code>tempdir()</code> .
force_download	Logical. If FALSE (default), reuse a cached file in data_directory or <code>tempdir()</code> when available. If TRUE, force a fresh download from Eurostat.

Details

Files are cached under `tempdir()` as RDS (e.g., "naio_10_cp1750.rds"). The temporary directory is cleared when the R session ends. To persist downloads across sessions (recommended for analytics), supply **data_directory** and the processed, **nested** output will also be written there as "<source>_processed.rds".

Supported Eurostat products include (non-exhaustive):

- **naio_10_cp1700** — Symmetric IO table, basic prices (product $\times$ product)
- **naio_10_pyp1700** — Same, previous years' prices
- **naio_10_cp1750** — Symmetric IO table, basic prices (industry $\times$ industry)
- **naio_10_pyp1750** — Same, previous years' prices
- **naio_10_cp15** — Supply table at basic prices incl. margins/taxes
- **naio_10_cp16** — Use table at purchasers' prices
- **naio_10_cp1610** — Use table at basic prices
- **naio_10_pyp1610** — Use table at basic prices (previous years' prices)
- **naio_10_cp1620** — Trade and transport margins at basic prices
- **naio_10_pyp1620** — Trade and transport margins at previous years' prices
- **naio_10_cp1630** — Taxes less subsidies on products at basic prices
- **naio_10_pyp1630** — Taxes less subsidies on products, prev. years' prices
- **uk_2010** — United Kingdom IO Analytical Tables (handled internally)

Eurostat API/format changes (e.g., **TIME_PERIOD** vs **time**) are handled for backward compatibility.

Value

A **nested data.frame** (one row per IO table) with metadata columns (**geo**, **unit**, **year**, **stk_flow**, etc.) and a list-column **data** containing the tidy table for each combination.

See Also

Other import functions: [employment_get\(\)](#), [iotables_download\(\)](#), [iotables_read_tempdir\(\)](#)

Examples

```
io_tables <- iotables_download(source = "naio_10_pyp1750")
```

```
iotables_metadata_get
```

Extract metadata from a downloaded IO table

Description

Returns only the metadata (e.g. country, year, unit, flow) from a nested input–output (IO) table created by [iotables_download\(\)](#). Removes the heavy **data** list-column, leaving just metadata rows.

Usage

```
iotables_metadata_get(dat = NULL, source = "naio_10_cp1700")
```

Arguments

dat	Optional. A nested tibble as returned by iotables_download() . If NULL , the function attempts to load it from tempdir() using the same cache naming convention (" <source>_processed.rds ").
source	Character. A valid data source code (see Details).

Value

A tibble with metadata columns only (no **data** list-column).

iotables_read_tempdir

Read input-output tables from temporary directory

Description

Validate the `source` input parameter and try to load the table from the current sessions' temporary directory.

Usage

```
iotables_read_tempdir(source = "naio_10_cp1700")
```

Arguments

<code>source</code>	See the available list of sources above in the Description. Defaults to <code>source = "naio_10_cp1700"</code> .
---------------------	--

Details

Possible `source` parameters:

`naio_10_cp1700` Symmetric input-output table at basic prices (product by product)

`naio_10_pyp1700` Symmetric input-output table at basic prices (product by product) (previous years prices)

`naio_10_cp1750` Symmetric input-output table at basic prices (industry by industry)

`naio_10_pyp1750` Symmetric input-output table at basic prices (industry by industry) (previous years prices)

`naio_10_cp15` Supply table at basic prices incl. transformation into purchasers' prices

`naio_10_cp16` Use table at purchasers' prices

`naio_10_cp1610` Use table at basic prices

`naio_10_pyp1610` Use table at basic prices (previous years prices) (`naio_10_pyp1610`)

`naio_10_cp1620` Table of trade and transport margins at basic prices

`naio_10_pyp1620` Table of trade and transport margins at previous years' prices

`naio_10_cp1630` Table of taxes less subsidies on products at basic prices

`naio_10_pyp1630` Table of taxes less subsidies on products at previous years' prices

`uk_2010_siot` United Kingdom Input-Output Analytical Tables data

Value

A nested data frame. Each input-output table is in a separate row of the nested output, where all the metadata are in columns, and the actual, tidy, ordered input-output table is in the data `data` column.

See Also

Other import functions: `employment_get()`, `iotables_download()`, `iotables_download_old()`

Examples

```
# The table must be present in the sessions' temporary directory:
iotables_download(source = "naio_10_pyp1750")

iotables_read_tempdir(source = "naio_10_pyp1750")
```

<code>is_html_output</code>	<i>Check if HTML output is required</i>
-----------------------------	---

Description

Check if HTML output is required

<code>is_latex_output</code>	<i>Check if Latex output is required</i>
------------------------------	--

Description

Check if Latex output is required

<code>key_column_create</code>	<i>Create a key columnn</i>
--------------------------------	-----------------------------

Description

Create a key column for matching the dimensions of matrixes.

Usage

```
key_column_create(key_column_name, key_column_values = NULL)
```

Arguments

- `key_column_name`
The name of the key column.
- `key_column_values`
The value(s) of the key column

Details

This function will likely be used with the creation of coefficients that need to be matched with a matrix that has a key column.

Value

A tibble with one column, named `key_column_name` and with values `key_column_values`.

See Also

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_find\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
key_column_create("iotables_row", c("CO2_multiplier", "CH4_multiplier"))
```

`leontief_inverse_create`

Create the Leontief inverse

Description

Computes the **Leontief inverse**, which measures the total (direct + indirect) requirements of all products or industries for one unit of final demand.

Usage

```
leontief_inverse_create(technology_coefficients_matrix, digits = NULL)
```

```
leontieff_inverse_create(technology_coefficients_matrix, digits = NULL)
```

Arguments

`technology_coefficients_matrix`

A technology-coefficient matrix created by [input_coefficient_matrix_create\(\)](#).

`digits`

Optional integer specifying rounding precision. Default NULL (no rounding).

Details

The Leontief inverse is defined as

$$L = (I - A)^{-1}$$

where A is the input-coefficient (or technology-coefficient) matrix created by [input_coefficient_matrix_create\(\)](#).

Each element l_{ij} of L shows the total production of product i required—both directly and indirectly—to satisfy one unit of final demand for product j .

This formulation follows the *Eurostat Manual of Supply, Use and Input–Output Tables* (Beutel 2008, Chapter 15; see equations (19), (43)) and the *UN Handbook on Supply and Use Tables and Input–Output Tables with Extensions and Applications* (2018 Rev. 1, pp. 619–621). The Leontief inverse is also called the **total requirements matrix** in both manuals.

For the analogous inverse based on output coefficients, see [ghosh_inverse_create\(\)](#).

Value

A `data.frame` with the original key column and the Leontief inverse in the remaining columns. If `digits` is supplied, values are rounded.

References

Beutel, J. (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*. Luxembourg: Office for Official Publications of the European Communities.

Validation examples: – Table 15.6 (p. 485): Direct requirements (input coefficients) – Table 15.10 (p. 488): Total requirements (Leontief inverse)

Results reproduced by `input_coefficient_matrix_create()` and `leontief_inverse_create()` using the built-in dataset `iotable_get(source = "germany_1995")`.

See Also

Other analytic object functions: [ghosh_inverse_create\(\)](#), [input_flow_get\(\)](#), [leontief_matrix_create\(\)](#), [output_coefficient_matrix_create\(\)](#)

Examples

```
# --- Minimal hand-checkable 2x2 system ---
minimal_matrix <- data.frame(
  sector = c("A", "B"),
  A = c(0.2, 0.4),
  B = c(0.1, 0.2)
)
leontief_inverse_create(minimal_matrix, digits = 3)

# --- Full Eurostat example (Germany 1995) ---
cm_de <- input_coefficient_matrix_create(
  data_table = iotable_get(source = "germany_1995")
)
leontief_inverse_create(cm_de)
```

`leontief_matrix_create`

Create the Leontief matrix ($I - A$)

Description

Builds the **Leontief matrix** ($I - A$) from a technology (input) coefficients matrix A . This is the intermediate step used before computing the Leontief inverse with [leontief_inverse_create\(\)](#).

Usage

```
leontief_matrix_create(technology_coefficients_matrix)

leontieff_matrix_create(technology_coefficients_matrix)
```

Arguments

`technology_coefficients_matrix`

A technology coefficients matrix created by `input_coefficient_matrix_create()` or `output_coefficient_matrix_create()`. The first column must be a key; remaining columns must be numeric.

Details

In Eurostat terminology (*Manual of Supply, Use and Input–Output Tables*), the technology coefficients matrix A is obtained by dividing each column of the inter-industry flows by the total output of that industry. The Leontief matrix is then $I - A$, also called the *matrix of intermediate coefficients subtracted from identity*.

This function removes any TOTAL rows/columns (e.g. "total", "cpa_total") before forming $I - A$. It returns a `data.frame` with the original key column followed by the numeric block of $I - A$.

Value

A `data.frame` whose first column is the key and whose remaining columns contain the Leontief matrix ($I - A$).

References

Beutel, J. (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*. Luxembourg: Office for Official Publications of the European Communities.

Validation examples: – Table 15.6 (p. 485): Direct requirements (input coefficients A) – **Table 15.9 (p. 487): Leontief matrix ($I - A$)** – Table 15.10 (p. 488): Leontief inverse (total requirements $L = (I - A)^{-1}$)

Results reproduced by `input_coefficient_matrix_create()`, `leontief_matrix_create()`, and `leontief_inverse_create()` using the built-in dataset `iotable_get(source = "germany_1995")`.

See Also

Other analytic object functions: `ghosh_inverse_create()`, `input_flow_get()`, `leontief_inverse_create()`, `output_coefficient_matrix_create()`

Examples

```
# From input coefficients (usual case)
tm <- input_coefficient_matrix_create(iotable_get(), households = FALSE)
L <- leontief_matrix_create(tm)
```

matrix_round	<i>Round Matrix Values</i>
--------------	----------------------------

Description

Round all numeric values in an input-output style table to a specified number of digits. The key column (first column) is preserved unchanged.

Usage

```
matrix_round(data_table, digits = 0)
```

Arguments

data_table	A symmetric input-output table, use table, supply table, tax table, or margins table.
digits	Integer number of decimal places to round to. Defaults to 0.

Details

This is useful for comparing results across software or publications that present rounded tables.

Value

A `data.frame` (or tibble) with the key column intact and all other numeric columns rounded to the given precision.

See Also

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_find\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
de_coeff <- input_coefficient_matrix_create(iotable_get())
head(matrix_round(de_coeff, digits = 2))
```

metadata_uk_2010	<i>Multipliers and Effects (Product) for Testing</i>
------------------	--

Description

A reference dataset derived from the **United Kingdom Input–Output Analytical Tables, 2010**. This version was imported from Excel and reformatted for internal testing.

Usage

```
metadata_uk_2010
```

Format

A data frame with 10 variables:

variable A constant used by `iortable_get()`.

uk_row Row identifiers from the UK tables. Dots and & were converted to -.

uk_col Column identifiers from the UK tables. Dots and & were converted to -.

uk_row_label Original UK row labels.

uk_col_label Original UK column labels.

eu_prod_na Eurostat vocabulary equivalent of **uk_row**.

row_order Ordering key for rows.

col_order Ordering key for columns.

prod_na Eurostat-like key values for rows.

induse Eurostat-like key values for columns.

Details

This dataset provides a mapping between the UK 2010 analytical input–output tables and Eurostat-compatible codes, intended mainly for testing and validation.

See Also

Other metadata datasets: [employment_metadata](#), [metadata](#)

multiplier_create *Create multipliers*

Description

Wrapper around `equation_solve()` that computes total multipliers by post-multiplying an input indicator vector with a Leontief inverse and adds a key column carrying the multiplier name for consistent joins.

Usage

```
multiplier_create(
  input_vector,
  Im,
  multiplier_name = "multiplier",
  digits = NULL
)
```

Arguments

<code>input_vector</code>	A named numeric vector (or 1-column matrix) created by <code>input_indicator_create()</code> whose names match the ordering of the Leontief inverse columns.
<code>Im</code>	A Leontief inverse matrix created by <code>leontief_inverse_create()</code> . Column names must correspond to products or industries consistent with <code>input_vector</code> .
<code>multiplier_name</code>	A string used for the key column that labels the returned multipliers. Default is "multiplier".
<code>digits</code>	Optional integer. If supplied and non-negative, round the resulting multipliers to this number of decimal places. Negative values are ignored (no rounding).

Details

In the Eurostat IO framework, multipliers measure *total* effects per unit of **final demand**, by product or industry (via the Leontief inverse $(I - A)^{-1}$). This contrasts with *direct effects*, which reflect only the immediate (first-round) impact.

The function delegates the numerical solve to `equation_solve()` and then formats the result for tidy joining with other IO tables. Ensure that the dimension ordering and names of `input_vector` and `Im` correspond; otherwise results will be misaligned.

Value

A data frame with:

- a first key column (character) named as the first column of `input_vector` and filled with `multiplier_name`, and
- one numeric column per product/industry containing the multipliers.

See Also

[equation_solve\(\)](#), [input_indicator_create\(\)](#), [leontief_inverse_create\(\)](#)

Examples

```
# Minimal workflow -----
data_table <- iotable_get()

coeff_de <- input_coefficient_matrix_create(data_table)

de_gva_indicator <- input_indicator_create(
  data_table = data_table,
  input = "gva"
)

I_de <- leontief_inverse_create(coeff_de)

de_gva_multipliers <- multiplier_create(
  input_vector = de_gva_indicator,
  Im = I_de,
  multiplier_name = "employment_multiplier",
  digits = 4
)
```

netherlands_2000	<i>Simplified input-output table for the Netherlands, 2000 (Spicosa example)</i>
------------------	--

Description

Aggregated symmetric input-output table (SIOT) for the Netherlands, reference year 2000, reproduced from the *Science Policy Integration for Coastal Systems Assessment* (Spicosa) project's multiplier specification sheet (D'Hernoncourt, Cordier & Hadley, 2011).

This dataset was originally created in the Spicosa project (circa 2006) as a simplified teaching table, based on OECD/Eurostat SIOT data. Column and row names were slightly adjusted to resemble Eurostat conventions and to align with the main example dataset [germany_1995](#).

Usage

```
netherlands_2000
```

Format

A data frame with 14 observations and 13 variables:

prod_na Simplified product/industry names.

agriculture_group Aggregated agricultural products.
mining_group Aggregated mining products.
manufacturing_group Aggregated manufacturing products.
construction_group Construction.
utilities_group Aggregated utilities products/services.
services_group Aggregated services products.
TOTAL Row/column sums; a simple summary not present in the original source.
final_consumption_private Aggregated final private consumption.
final_consumption_households Aggregated final household consumption.
final_consumption_government Aggregated final government consumption.
gross_fixed_capital_formation Gross fixed capital formation (GFCF).
exports Aggregated exports.
total_use Aggregated total use.

Details

The Spicosa specification sheet demonstrates the derivation of type I and type II multipliers step by step from this table. This dataset corresponds to Table 1 of that report, the domestic transactions input–output table (million EUR, year 2000). It is not an official Statistics Netherlands SIOT, but a simplified, aggregated example for multiplier analysis.

Source

D'Hernoncourt, J., Cordier, M. & Hadley, D. (2011). *Input–Output Multipliers: Specification sheet and supporting material*. Spicosa Project Report. <https://hal.science/hal-03233439>

See Also

Other validation datasets: [germany_airpol](#), [uk_2010_data](#), [uk_test_results](#)

output_coefficient_matrix_create

Create an output coefficient matrix

Description

Create an output-coefficient matrix from a symmetric input–output table or a use table. Output coefficients can be interpreted as the market shares of products in total output (row-wise normalization).

Usage

```
output_coefficient_matrix_create(data_table, total = "tfu", digits = NULL)
```

Arguments

<code>data_table</code>	A symmetric input–output table, use table, margins, or tax table retrieved by <code>iortable_get()</code> . If you request <code>total = "tfu"</code> (total final use), you must supply a full table from <code>iortable_get()</code> because the TFU column is in the second quadrant.
<code>total</code>	Which total to use for normalization. Use <code>"total"</code> (or the present table variant name, e.g. <code>"CPA_TOTAL"</code>) for output by product, or <code>"tfu"</code> / <code>"total_final_use"</code> / <code>"final_demand"</code> for total final use. Default: <code>"tfu"</code> .
<code>digits</code>	Integer number of decimal places for rounding. Default <code>NULL</code> (no rounding).

Details

Let Z be the inter-industry flow block and x the vector of product output (or, for final-demand shares, total final use). The output-coefficient matrix B is defined row-wise as $b_{ij} = z_{ij}/x_i$. In practice, zeros in the denominator can make equations unsolvable; this function replaces zeros with a small epsilon ($1e-6$) to avoid division by zero.

Eurostat, *Manual of Supply, Use and Input-Output Tables* (e.g., pp. 495, 507) describes output coefficients and the Ghosh framework you may use these with.

Value

A `data.frame` whose first column is the key (product labels) and the remaining columns form the output-coefficient matrix. Column order follows the input.

See Also

Other analytic object functions: `ghosh_inverse_create()`, `input_flow_get()`, `leontief_inverse_create()`, `leontief_matrix_create()`

Examples

```
data_table <- iortable_get()
output_coefficient_matrix_create(
  data_table = data_table,
  total = "tfu",
  digits = 4
)
```

output_get	<i>Get the output (P1) vector</i>
------------	-----------------------------------

Description

Convenience wrapper around `primary_input_get()` that returns the row labelled **Output (P1)** from a symmetric input-output table (SIOT) or from a use table retrieved by `iortable_get()`.

Usage

```
output_get(data_table)
```

Arguments

`data_table` A symmetric input-output table or use table retrieved by `iortable_get()`.

Details

In the Eurostat framework, **Output** is transaction **P1**, usually recorded at **basic prices** (often labelled "output" or "output_bp"). It is a balancing item of the use table / SIOT, **not** a "primary input" (primary inputs are value added components and imports, shown in the third quadrant). This helper merely selects the row labelled "output", "output_bp", "P1" or "p1" if present.

Value

A one-row data frame: the first column is the key column; remaining columns give output (P1) by product/industry.

See Also

`primary_input_get()`, `iortable_get()`

Other iotables processing functions: `conforming_vector_create()`, `empty_remove()`, `household_column_find()`, `household_column_get()`, `iortable_year_get()`, `key_column_create()`, `matrix_round()`, `primary_input_get()`, `rows_add()`, `supplementary_add()`, `total_tax_add()`, `vector_transpose_longer()`, `vector_transpose_wider()`

Examples

```
# Output (P1) from the package demo table
iot_germany <- iortable_get()
output_get(data_table = iot_germany)
```

```
output_multiplier_create
```

Create output multipliers

Description

Compute output multipliers from a Leontief inverse matrix.

Usage

```
output_multiplier_create(input_coefficient_matrix)
```

Arguments

`input_coefficient_matrix`

A technology-coefficient matrix as returned by [input_coefficient_matrix_create\(\)](#).

Details

The output multipliers are defined as the **column sums** of the Leontief inverse $(I - A)^{-1}$, where A is the input coefficient matrix. They measure the total direct and indirect output generated in each industry per unit increase in final demand.

See Eurostat (2008), *Manual of Supply, Use and Input-Output Tables*, p. 500; UN (2018), *Handbook on Supply and Use Tables and Input-Output Tables with Extensions and Applications*, §15.35.

Value

A one-row `data.frame` (or tibble) with:

- The first column a label "output_multipliers".
- Remaining columns the multipliers for each industry.

See Also

Other multiplier functions: [input_multipliers_create\(\)](#)

Examples

```
de_input_coeff <- input_coefficient_matrix_create(
  iotable_get(),
  digits = 4
)

output_multiplier_create(de_input_coeff)
```

prd_ava	<i>Eurostat Product Supply, Adjustments and Value Added Vocabulary (prd_ava)</i>
---------	--

Description

A reference codelist used in Eurostat’s national accounts framework for **product** × **product** symmetric input–output tables (SIOTs). This vocabulary enumerates the **rows** of product-based tables, covering intermediate products (Quadrant 1), value-added components (Quadrant 3), and selected control totals such as “*Total supply at basic prices*”.

Usage

prd_ava

Format

A tibble (data frame) with 10 variables:

id Canonical Eurostat concept identifier.

label Official Eurostat label (preferred English term).

status "Valid" or "Valid in CPA2_1".

status_modified Date of last status modification.

notation Alternate human-readable code or short form (e.g. CPA_A01).

ind_ava_notation Legacy comparison column retained for internal cross-checks.

quadrant Integer designating the analytical block: 10 = Intermediate (technology block, Quadrant 1); 20 = Primary inputs / GVA (Quadrant 3); 30 = Control totals; 50 = Extensions / adjustments.

numeric_order Ordinal position key for ordering within quadrants.

iotables_label Simplified legacy label used in the [iotables](#) package.

block Semantic block category inferred from **quadrant** (`intermediate`, `primary_inputs`, `control_total`, `extension`).

uri Stable URI linking to the Eurostat SKOS concept at <https://dd.eionet.europa.eu/>.

Details

The **prd_ava** vocabulary aligns with the official Eurostat SDMX codelist “*Products, adjustments and value added*”. It is used by the [iotables](#) package to order the **row dimension** of product-by-product tables (datasets such as `naio_10_cp1700`).

Codes correspond to **CPA 2.1** product classifications and ESA 2010 accounting aggregates. Each entry includes a quadrant identifier and an ordinal numeric order for consistent placement when reshaping long-form data.

The dataset retains published control totals and balancing items for completeness (e.g., *Total supply at basic prices* or *CIF/FOB adjustments*), but these rows are normally excluded from analytical matrices.

Because the Eurostat SIOT datasets sometimes contain more detailed industry breakup than their vocabulary, we adopted industry coding from [cpa2_1](#). These codes are marked the "Valid in CPA2_1" status.

Source

Eurostat metadata registry (https://dd.eionet.europa.eu/vocabulary/eurostat/prd_ava/)

See Also

Related Eurostat vocabularies: [ind_ava](#), [ind_use](#), [prd_use](#), [cpa2_1](#).

Examples

```
data(prd_ava)
```

prd_use	<i>Eurostat Product Uses Vocabulary (prd_use)</i>
---------	---

Description

A reference codelist used in Eurostat's national accounts framework for **product** × **product** symmetric input-output tables (SIOTs). This vocabulary enumerates the **columns** of product-based tables, covering intermediate product uses (Quadrant 1), value-added components (Quadrant 3), and the final-use block (Quadrant 2), as well as balancing or extension items.

Usage

```
prd_use
```

Format

A tibble (data frame) with 10 variables:

id Canonical Eurostat concept identifier.

label Official Eurostat label (preferred English term).

status "Valid" or "Valid in CPA2_1".

status_modified Date of last status modification.

notation Alternate human-readable code or short form (e.g. CPA_A01).

quadrant Integer designating the analytical block: 10 = Intermediate (technology block, Q1); 20 = Primary inputs / GVA (Q3); 30 = Final use (Q2); 50 = Extensions / diagnostics.

numeric_order Ordinal key for ordering columns within quadrants.

iotables_label Simplified legacy label used in the [iotables](#) package.

block Semantic block category inferred from **quadrant** (*intermediate*, *primary_inputs*, *final_use*, *extension*).

uri Stable URI linking to the Eurostat SKOS concept at <https://dd.eionet.europa.eu/>.

Details

The `prd_use` vocabulary corresponds to the official Eurostat SDMX codelist “*Product uses*”. It is used by the `iotables` package to order the **column dimension** of product-by-product tables (datasets such as `naio_10_cp1700`).

Codes follow **CPA 2.1** for products and **ESA 2010** for accounting aggregates. Each entry includes a quadrant identifier and a numeric order key that ensures consistent column ordering when reshaping Eurostat’s long-form data into analytical matrices.

The dataset also retains control totals and adjustment items (Quadrant 30/50) such as *Total use at purchasers’ prices*, *CIF/FOB adjustments*, or *Discrepancy*, which are part of Eurostat’s published codelist but are normally excluded from matrix algebra operations.

Because the Eurostat SIOT datasets sometimes contain more detailed industry breakup than their vocabulary, we adopted industry coding from `cpa2_1`. These codes are marked the “Valid in CPA2_1” status.

Source

Eurostat metadata registry (https://dd.eionet.europa.eu/vocabulary/eurostat/prd_use/)

See Also

Related Eurostat vocabularies: `prd_ava`, `ind_ava`, `ind_use`.

Examples

```
data(prd_use)
```

<code>primary_input_get</code>	<i>Get a primary input row</i>
--------------------------------	--------------------------------

Description

Retrieve a named primary-input row from a symmetric input–output table, a use table, or a supply table (as returned by `iotable_get()`).

Usage

```
primary_input_get(data_table, primary_input = "compensation_employees")
```

Arguments

<code>data_table</code>	A symmetric I–O table, use table, or supply table as returned by <code>iotable_get()</code> .
<code>primary_input</code>	Character. The primary input to return. Accepts common synonyms (e.g., “compensation of employees”, “cfc”, “taxes on production”, “operating surplus”, “imports”).

Details

In I–O accounting, *primary inputs* (e.g., compensation of employees, consumption of fixed capital, taxes on production/subsidies, operating surplus/mixed income, and—when relevant—imports used for domestic production) are shown in the value-added block (third quadrant).

Value

A data frame containing the key column and the matching primary- input row.

References

Eurostat (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*, ch. 13. United Nations (2018). *Handbook on Supply and Use Tables and Input–Output Tables with Extensions and Applications (Rev. 1, “white cover”)*, ch. 10.

See Also

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_find\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
# Get the Germany 1995 demo SIOT with default labelling
de_iot <- iotable_get(source = "germany_1995")

# Select compensation of employees (row code: "compensation_employees")
primary_input_get(de_iot, "compensation_employees")

# Get the same table with Eurostat short labelling
de_iot_short <- iotable_get(source = "germany_1995", labelling = "short")

# Consumption of fixed capital (row code: "K1")
primary_input_get(de_iot_short, "K1")

# Operating surplus and mixed income, net (row code: "B2A3N")
primary_input_get(de_iot_short, "B2A3N")
```

primary_inputs	<i>Primary input abbreviations</i>
----------------	------------------------------------

Description

Only currently used primary inputs. Abbreviations for filtering.

Usage

```
data("croatia_employment_aggregation")
```

Format

- A data frame with 105 rows (including empty ones) and 2 variables.
- `t_rows2` Eurostat code of the input.
- `t_rows2_lab` Labelling of the input by Eurostat.
- `source` Eurostat / DZS
- `indicator` Human readable abbreviation

See Also

Other Croatia 2010 datasets: [croatia_2010_1700](#), [croatia_2010_1800](#), [croatia_2010_1900](#), [croatia_employment_2013](#), [croatia_employment_aggregation](#)

rows_add	Add Conforming Row(s) to an Input-Output Table
----------	--

Description

Add a conforming row, or elements of a conforming row, to a named input-output style data frame.

Usage

```
rows_add(data_table, rows_to_add, row_names = NULL, empty_fill = 0)
```

Arguments

- `data_table` A symmetric input-output table, a use table, a margins table, or a tax table retrieved by [iotable_get\(\)](#).
- `rows_to_add` A data frame or a named numeric vector containing the new row(s).
- `row_names` Optional character vector giving names for the new key column. If `NULL`, names are inferred (see *Details*).
- `empty_fill` Value used to fill missing columns. Defaults to 0.

Details

You can add rows in several ways:

- A **data frame** with one or more rows, where the first column contains row identifiers.
- A **named numeric vector**, which will be turned into a single-row data frame.

If no `row_names` are supplied and the first column of `rows_to_add` is numeric, new rows will be automatically labelled as "new_row_1", "new_row_2", etc.

Missing column values are filled with `empty_fill`, which defaults to 0. If you want to avoid division by zero in later computations, you can set this to a very small value (e.g. `1e-6`).

Value

A `data.frame` containing the original `data_table` extended with the new row(s).

See Also

Other iotables processing functions: `conforming_vector_create()`, `empty_remove()`, `household_column_find()`, `household_column_get()`, `iotable_year_get()`, `key_column_create()`, `matrix_round()`, `output_get()`, `primary_input_get()`, `supplementary_add()`, `total_tax_add()`, `vector_transpose_longer()`, `vector_transpose_wider()`

Examples

```
rows_to_add <- data.frame(
  iotables_row = "CO2_emission",
  agriculture_group = 10448,
  industry_group = 558327, # construction is omitted
  trade_group = 11194
)

rows_add(iotable_get(), rows_to_add = rows_to_add)

rows_add(iotable_get(),
  rows_to_add = c(
    industry_group = 1534,
    trade_group = 4
  ),
  row_names = "CH4_emission"
)
```

<code>supplementary_add</code>	<i>Add supplementary rows to an IO/SUT table</i>
--------------------------------	--

Description

Append supplementary indicators (e.g., emissions coefficients) as new rows to a symmetric input-output table (SIOT), use, supply, or margins table. This is a light wrapper around `rows_add()`.

Usage

```
supplementary_add(data_table, supplementary_data, supplementary_names = NULL)
```

Arguments

`data_table` A SIOT, use, supply, or margins table (key column + numeric columns).

supplementary_data

A data frame (or tibble) of one or more rows to add. It may already contain a key column (first column). Otherwise, provide **supplementary_names** or the keys will be auto-generated. All other column names must match **data_table**.

supplementary_names

Optional character vector of row names for the key column; length must equal `nrow(supplementary_data)`. Ignored if a key column is already present.

Details

Column names in **supplementary_data** must match the numeric columns of **data_table**. If the key column is missing, it is created from **supplementary_names** or auto-generated as **supplementary_row_#**.

When a household final consumption column is present (e.g., **final_consumption_households**, P3_S14), new rows get 0 in that column if the supplied values are NA.

For terminology, see Eurostat's *Manual of Supply, Use and Input-Output Tables*. (Eurostat, 2008; ISBN 978-92-79-04704-3)

Value

A **data.frame** with the rows of **supplementary_data** bound to **data_table** and aligned to its key and numeric columns.

References

Table 15.13 (p. 494) in Beutel (2008), *Eurostat Manual of Supply, Use and Input-Output Tables*, shows an environmental-emission model built by adding CO and CH rows to the German 1995 input-output coefficients. The same construction is reproduced here by **supplementary_add()** and the results are checked against this example.

See Also

[convert_industry_to_product\(\)](#)

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_find\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_longer\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
de_io <- iotable_get()
CO2_coefficients <- data.frame(
  agriculture_group = 0.2379,
  industry_group = 0.5172,
  construction = 0.0456,
  trade_group = 0.1320,
  business_services_group = 0.0127,
  other_services_group = 0.0530
```

```

)
CH4_coefficients <- data.frame(
  agriculture_group = 0.0349,
  industry_group = 0.0011,
  construction = 0,
  trade_group = 0,
  business_services_group = 0,
  other_services_group = 0.0021
)
CO2 <- cbind(
  data.frame(iotables_row = "CO2"),
  CO2_coefficients
)
CH4 <- cbind(
  data.frame(iotables_row = "CH4_coefficients"),
  CH4_coefficients
)
de_coeff <- input_coefficient_matrix_create ( iotable_get() )
emissions <- rbind (CO2, CH4)

# Check with the Eurostat Manual page 494:
supplementary_add(de_io, emissions)

```

total_tax_add	<i>Add a total tax row (D.2–D.3 and D.29–D.39)</i>
---------------	--

Description

Create and append a **total tax** row by summing selected tax rows in the primary inputs block (Quadrant III) of a SIOT or use table.

Usage

```

total_tax_add(
  data_table,
  tax_names = c("d21x31", "d29x39"),
  total_tax_name = "TOTAL_TAX"
)

```

Arguments

data_table	A symmetric input–output table (SIOT) or use table whose primary inputs include tax rows (see Details). Typically obtained via <code>iotable_get()</code> .
tax_names	Character vector of row labels to sum. Defaults to <code>c("d21x31", "d29x39")</code> , shorthand for D.2–D.3 and D.29–D.39. Matching is currently made against the lower-cased key column .
total_tax_name	Character scalar for the new row label. Default "TOTAL_TAX". (See Enhancements regarding case handling.)

Details

In Eurostat/ESA terminology, tax rows commonly include:

- **Taxes less subsidies on products** (codes D.2–D.3), and
- **Other net taxes on production** (codes D.29–D.39).

These appear in the value-added (primary inputs) section of the use/SIOT layout. The function sums the specified rows **column-wise** over all numeric columns and appends the result as `total_tax_name`. If a household final consumption column is present (e.g. `final_consumption_households` or `p3_s14`), any missing value in the new total row is replaced by zero.

Value

A data frame like `data_table`, with one additional row named `total_tax_name` that equals the element-wise sum of the rows in `tax_names` over numeric columns.

Terminology

Eurostat uses the lines “Taxes less subsidies on products” and “Other net taxes on production” in published tables; these correspond, respectively, to D.2–D.3 and D.29–D.39.

References

Eurostat (2008). *Eurostat Manual of Supply, Use and Input–Output Tables*, ch. 13. United Nations (2018). *Handbook on Supply and Use Tables and Input–Output Tables with Extensions and Applications (Rev. 1, “white cover”)*, ch. 10.

See Also

Other iotables processing functions: `conforming_vector_create()`, `empty_remove()`, `household_column_find()`, `household_column_get()`, `iotable_year_get()`, `key_column_create()`, `matrix_round()`, `output_get()`, `primary_input_get()`, `rows_add()`, `supplementary_add()`, `vector_transpose_longer()`, `vector_transpose_wider()`

Examples

```
de_io <- iotable_get()
total_tax_add(
  data_table = de_io,
  tax_names = c("net_tax_products", "net_tax_production"),
  total_tax_name = "total_tax"
)
```

uk_2010_data

United Kingdom Input–Output Analytical Tables, 2010

Description

Official **Input–Output Analytical Tables** for the United Kingdom (2010), published by the Office for National Statistics (ONS).

These replication data are harmonized to the *iotables* package format and can be accessed via `iotable_get()` by setting the `source` parameter to one of the available tables.

Usage

```
data(uk_2010_data)
```

Format

A `tibble` with **10 variables**:

uk_row Row identifier (dots and `&` converted to `-`).

uk_row_lab Original ONS row label.

uk_col Column identifier (dots and `&` converted to `-`).

uk_col_lab Original ONS column label.

geo Eurostat-style geocode (constant: "UK").

geo_lab Human-readable country label (constant: "United Kingdom").

indicator Indicator or table name, matching Excel sheet title.

unit Eurostat-style unit code (e.g. "MIO_NAC").

unit_lab Human-readable unit label (e.g. "Million national currency").

values Numeric values for each indicator.

year Constant value: 2010.

Details

The dataset includes five analytical table types from the 2010 UK Input–Output release:

- **uk_2010_siot** — Input–Output table (domestic use, basic prices, product $\times$ product)
- **uk_2010_use** — Domestic use table at basic prices (product $\times$ industry)
- **uk_2010_imports** — Imports use table at basic prices (product $\times$ product)
- **uk_2010_coeff** — Matrix of coefficients (product $\times$ product)
- **uk_2010_inverse** — Leontief inverse (product $\times$ product)

These correspond to the Excel sheets from the official ONS publication.

Source

Office for National Statistics (2010). *United Kingdom Input–Output Analytical Tables, 2010*. Retrieved from the [UK National Archives \(ONS Web Archive\)](#).

See Also

- [iotable_get\(\)](#) for accessing and filtering IO tables.
- [iotables_download\(\)](#) for downloading current Eurostat and UK datasets.

Other validation datasets: [germany_airpol](#), [netherlands_2000](#), [uk_test_results](#)

uk_2010_results_get *Get United Kingdom Multipliers and Effects, 2010*

Description

This function will retrieve the published effects and multipliers from the United Kingdom Input-Output Analytical Tables, 2010 (consistent with UK National Accounts Blue Book 2013 & UK Balance of Payments Pink Book 2013) by Richard Wild.

Usage

```
uk_2010_results_get(path = NULL)
```

Arguments

path A path to the downloaded file, if already exists, given with `file.path()` function.

Source

[ukioanalyticaltablesio1062010detailedpubversion.xls](#)

Examples

```
## Not run:
uk_results <- iotables::uk_2010_results_get()

## End(Not run)
```

uk_test_results	<i>UK multipliers and effects (product), 2010</i>
-----------------	---

Description

Published multipliers and effects from the United Kingdom Input–Output Analytical Tables, reference year 2010.

This dataset contains output, employment cost, and GVA multipliers and effects, together with their published rankings. It is imported from the official ONS Excel release and normalized for use in **iotables**. It is primarily used in the `vignette("united_kingdom_2010", package = "iotables")` to validate the package’s multiplier functions against official UK results.

Usage

```
uk_test_results
```

Format

A tibble with 127 rows and 12 variables:

uk_row_label Product/industry label.

output_multiplier Output multiplier (published).

output_multiplier_rank Ranking of output multipliers.

employment_cost_multiplier Employment cost multiplier.

employment_cost_multiplier_rank Ranking of employment cost multipliers.

employment_cost_effects Employment cost effects.

employment_cost_effects_rank Ranking of employment cost effects.

gva_multiplier GVA multiplier.

gva_multiplier_rank Ranking of GVA multipliers.

gva_effects GVA effects.

gva_effects_rank Ranking of GVA effects.

indicator Indicator label, usually "Multipliers and effects (product)".

Details

The Office for National Statistics (ONS) publishes Input–Output Analytical Tables (IOATs) for the UK. From these, Type I and Type II multipliers and effects are calculated. This dataset contains those published values at the product level for 2010, enabling direct cross-checks with **iotables** computations.

Source

Office for National Statistics (ONS), UK Input–Output Analytical Tables 2010 (Excel release).

See Also

```
vignette("united_kingdom_2010", package = "iotables")
```

Other validation datasets: [germany_airpol](#), [netherlands_2000](#), [uk_2010_data](#)

```
vector_transpose_longer
```

Transpose a Vector to Long Form

Description

Convert a wide-form vector (e.g., indicators or multipliers) into long form, which is often more useful for printing or joining. This is a thin wrapper around `tidyr::pivot_longer()`, provided so you do not need to load **tidyr** explicitly.

Usage

```
vector_transpose_longer(
  data_table,
  names_to = "nace_r2",
  values_to = "value",
  key_column_name = NULL,
  .keep = FALSE
)

vector_transpose(
  data_table,
  names_to = "nace_r2",
  values_to = "value",
  key_column_name = NULL,
  .keep = FALSE
)
```

Arguments

<code>data_table</code>	A <code>data.frame</code> or tibble. The first column is assumed to be a key column.
<code>names_to</code>	Name of the new column containing previous column names. Default: "nace_r2".
<code>values_to</code>	Name of the new column containing the values. Default: "value".
<code>key_column_name</code>	Optional. New name for the first (key) column. If <code>NULL</code> (default), the name is not changed.
<code>.keep</code>	Logical. If <code>TRUE</code> , keep the indicator identifier column. If <code>FALSE</code> (default), drop it.

Value

A tibble in long format with a key column and, if requested, the indicator identifier column.

See Also

Other iotables processing functions: [conforming_vector_create\(\)](#), [empty_remove\(\)](#), [household_column_find\(\)](#), [household_column_get\(\)](#), [iotable_year_get\(\)](#), [key_column_create\(\)](#), [matrix_round\(\)](#), [output_get\(\)](#), [primary_input_get\(\)](#), [rows_add\(\)](#), [supplementary_add\(\)](#), [total_tax_add\(\)](#), [vector_transpose_wider\(\)](#)

Examples

```
vector_transpose_longer(  
  data.frame(  
    indicator = "my_indicator",  
    agriculture = 0.0123,  
    manufacturing = 0.1436,  
    trade = 0.0921  
  )  
)  
  
# Keep the indicator column  
vector_transpose_longer(  
  data.frame(  
    indicator = "my_indicator",  
    agriculture = 0.0123,  
    manufacturing = 0.1436  
  ),  
  .keep = TRUE  
)
```

`vector_transpose_wider`

Transpose a Vector to Wide Form

Description

Convert a long-form vector (e.g., indicators, multipliers) into wide form, which is often more useful for binding with input-output tables. This is a thin wrapper around [tidyr::pivot_wider\(\)](#), provided so you do not need to load **tidyr** explicitly.

Usage

```
vector_transpose_wider(  
  data_table,  
  names_from,  
  values_from,  
  key_column_name = NULL,  
  key_column_values = NULL  
)
```

Arguments

- data_table** A `data.frame` or tibble, normally with a key column. If the key column must be created or replaced, use `key_column_name` and `key_column_values`.
- names_from, values_from** Columns specifying the names of the output columns (`names_from`) and the values to fill (`values_from`).
- key_column_name** The name of the key column.
- key_column_values** Optional explicit key column values. Default: `NULL`, in which case values are inferred from the long data.

See Also

Other iotables processing functions: `conforming_vector_create()`, `empty_remove()`, `household_column_find()`, `household_column_get()`, `iotable_year_get()`, `key_column_create()`, `matrix_round()`, `output_get()`, `primary_input_get()`, `rows_add()`, `supplementary_add()`, `total_tax_add()`, `vector_transpose_longer()`

Examples

```
vector_transpose_wider(  
  data_table = germany_airpol[, -2],  
  names_from = "induse",  
  values_from = "value"  
)  
  
vector_transpose_wider(  
  data_table = germany_airpol[1:8, 3:4],  
  names_from = "induse",  
  values_from = "value",  
  key_column_values = "CO2_emission"  
)
```

Index

- * .
 - input_indicator_create, [34](#)
 - multiplier_create, [54](#)
- * @references
 - multiplier_create, [54](#)
- * Beutel, J. (2008).
 - input_indicator_create, [34](#)
 - multiplier_create, [54](#)
- * CPA
 - cpa2_1, [12](#)
- * Communities.
 - multiplier_create, [54](#)
- * Croatia 2010 datasets
 - croatia_2010_1700, [13](#)
 - croatia_2010_1800, [14](#)
 - croatia_2010_1900, [15](#)
 - croatia_employment_2013, [16](#)
 - croatia_employment_aggregation,
[16](#)
 - primary_inputs, [63](#)
- * European Communities.
 - input_indicator_create, [34](#)
- * Eurostat
 - cpa2_1, [12](#)
 - ind_ava, [28](#)
 - ind_use, [29](#)
 - prd_ava, [60](#)
 - prd_use, [61](#)
- * Luxembourg: Office for Official
Publications of the European
multiplier_create, [54](#)
- * Luxembourg: Office for Official
Publications of the
input_indicator_create, [34](#)
- * Results reproduced by
 - input_indicator_create, [34](#)
 - multiplier_create, [54](#)
- * SIOT
 - cpa2_1, [12](#)
 - ind_ava, [28](#)
 - ind_use, [29](#)
 - prd_ava, [60](#)
 - prd_use, [61](#)
- * SUT
 - cpa2_1, [12](#)
 - ind_ava, [28](#)
 - ind_use, [29](#)
 - prd_ava, [60](#)
 - prd_use, [61](#)
- * Validation examples:
 - input_indicator_create, [34](#)
 - multiplier_create, [54](#)
- * #' @references
 - input_indicator_create, [34](#)
- * analytic object functions
 - ghosh_inverse_create, [25](#)
 - input_flow_get, [32](#)
 - leontief_inverse_create, [49](#)
 - leontief_matrix_create, [50](#)
 - output_coefficient_matrix_create,
[56](#)
- * datasets
 - cpa2_1, [12](#)
 - croatia_2010_1700, [13](#)
 - croatia_2010_1800, [14](#)
 - croatia_2010_1900, [15](#)
 - croatia_employment_2013, [16](#)
 - croatia_employment_aggregation,
[16](#)
 - employment_metadata, [19](#)
 - germany_1995, [23](#)
 - germany_airpol, [24](#)
 - ind_ava, [28](#)
 - ind_use, [29](#)
 - metadata_uk_2010, [53](#)
 - netherlands_2000, [55](#)
 - prd_ava, [60](#)
 - prd_use, [61](#)

- primary_inputs, 63
- uk_2010_data, 69
- uk_test_results, 71
- * **import functions**
 - employment_get, 18
 - iotables_download, 43
 - iotables_download_old, 44
 - iotables_read_tempdir, 47
- * **indicator functions**
 - coefficient_matrix_create, 7
 - direct_effects_create, 17
 - input_indicator_create, 34
- * **input-output**
 - ind_ava, 28
 - ind_use, 29
 - prd_ava, 60
 - prd_use, 61
- * **iotables import functions**
 - iotable_get, 36
 - iotable_get_builtin, 38
- * **iotables processing functions**
 - conforming_vector_create, 10
 - empty_remove, 20
 - household_column_find, 26
 - household_column_get, 27
 - iotable_year_get, 41
 - key_column_create, 48
 - matrix_round, 52
 - output_get, 58
 - primary_input_get, 62
 - rows_add, 64
 - supplementary_add, 65
 - total_tax_add, 67
 - vector_transpose_longer, 72
 - vector_transpose_wider, 73
- * **linkage functions**
 - backward_linkages, 6
 - forward_linkages, 22
- * **list(Eurostat Manual of Supply, Use and Input-Output Tables.)**
 - input_indicator_create, 34
 - multiplier_create, 54
- * **list(input_indicator_create())**
 - input_indicator_create, 34
- * **list(iotable_get(source = \germany_1995\))**
 - input_indicator_create, 34
 - multiplier_create, 54
- * **list(multiplier_create())**
 - multiplier_create, 54
- * **metadata datasets**
 - employment_metadata, 19
 - metadata_uk_2010, 53
- * **metadata**
 - cpa2_1, 12
 - ind_ava, 28
 - ind_use, 29
 - prd_ava, 60
 - prd_use, 61
- * **multiplier functions**
 - input_multipliers_create, 35
 - multiplier_create, 54
 - output_multiplier_create, 59
- * **products**
 - cpa2_1, 12
- * **using**
 - input_indicator_create, 34
 - multiplier_create, 54
- * **validation datasets**
 - germany_airpol, 24
 - netherlands_2000, 55
 - uk_2010_data, 69
 - uk_test_results, 71
- *
- input_indicator_create, 34
- multiplier_create, 54
- * – **Table 15.16 (pp. 503–504): Total multipliers (Germany 1995)**
 - multiplier_create, 54
- * – **pp. 495–498: Input indicators (GVA, compensation of employees)**
 - input_indicator_create, 34
- airpol_get, 3
- align_to_io_codes, 4
- backward_linkages, 6, 23
- coefficient_matrix_create, 7, 17
- coefficient_matrix_create(), 32, 34
- conforming_vector_create, 10, 21, 27, 42, 49, 52, 58, 63, 65, 66, 68, 73, 74
- convert_industry_to_product, 11
- convert_industry_to_product(), 66
- cpa2_1, 12, 29, 30, 61, 62

- croatia_2010_1700, [13](#), [15–17](#), [64](#)
- croatia_2010_1800, [14](#), [14–17](#), [64](#)
- croatia_2010_1900, [14](#), [15](#), [15–17](#), [64](#)
- croatia_employment_2013, [14](#), [15](#), [16](#), [17](#), [64](#)
- croatia_employment_aggregation, [14](#), [15](#), [16](#), [16](#), [64](#)
- direct_effects_create, [9](#), [17](#)
- employment_get, [18](#), [44](#), [46](#), [48](#)
- employment_metadata, [19](#), [53](#)
- empty_remove, [10](#), [20](#), [27](#), [42](#), [49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- equation_solve, [21](#)
- equation_solve(), [54](#), [55](#)
- forward_linkages, [7](#), [22](#)
- germany_1995, [23](#), [55](#)
- germany_airpol, [4](#), [24](#), [56](#), [70](#), [72](#)
- ghosh_inverse_create, [25](#), [33](#), [50](#), [51](#), [57](#)
- ghosh_inverse_create(), [50](#)
- household_column_find, [10](#), [21](#), [26](#), [27](#), [42](#), [49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- household_column_get, [10](#), [21](#), [27](#), [27](#), [42](#), [49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- ind_ava, [13](#), [28](#), [30](#), [61](#), [62](#)
- ind_use, [13](#), [29](#), [29](#), [61](#), [62](#)
- indirect_effects_create, [30](#)
- input_coefficient_matrix_create, [31](#)
- input_coefficient_matrix_create(), [33](#), [49](#), [51](#), [59](#)
- input_flow_get, [25](#), [32](#), [50](#), [51](#), [57](#)
- input_indicator_create, [6](#), [17](#), [34](#), [35](#)
- input_indicator_create(), [31](#), [54](#), [55](#)
- input_multipliers_create, [35](#), [59](#)
- iotable_get, [6](#), [36](#), [40](#)
- iotable_get(), [8](#), [24](#), [32–34](#), [40](#), [42–44](#), [53](#), [57](#), [58](#), [62](#), [64](#), [67](#), [69](#), [70](#)
- iotable_get_builtin, [38](#), [38](#)
- iotable_get_eurostat, [40](#)
- iotable_year_get, [10](#), [21](#), [27](#), [41](#), [49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- iotables, [12](#), [28–30](#), [43](#), [60–62](#)
- iotables_download, [19](#), [43](#), [46](#), [48](#)
- iotables_download(), [36–38](#), [41](#), [42](#), [46](#), [70](#)
- iotables_download_old, [19](#), [44](#), [44](#), [48](#)
- iotables_metadata_get, [46](#)
- iotables_read_tempdir, [19](#), [44](#), [46](#), [47](#)
- is_html_output, [48](#)
- is_latex_output, [48](#)
- key_column_create, [10](#), [21](#), [27](#), [42](#), [48](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- leontief_inverse_create, [6](#), [17](#), [25](#), [33](#), [35](#), [49](#), [51](#), [57](#)
- leontief_inverse_create(), [7](#), [25](#), [31](#), [33](#), [50](#), [54](#), [55](#)
- leontief_matrix_create, [25](#), [33](#), [50](#), [50](#), [57](#)
- leontieff_inverse_create
(*leontief_inverse_create*), [49](#)
- leontieff_matrix_create
(*leontief_matrix_create*), [50](#)
- matrix_round, [10](#), [21](#), [27](#), [42](#), [49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- metadata, [20](#), [53](#)
- metadata_uk_2010, [20](#), [53](#)
- multiplier_create, [54](#)
- multiplier_create(), [21](#)
- netherlands_2000, [25](#), [55](#), [70](#), [72](#)
- output_coefficient_matrix_create, [25](#), [33](#), [50](#), [51](#), [56](#)
- output_coefficient_matrix_create(), [22](#), [25](#), [51](#)
- output_get, [10](#), [21](#), [27](#), [42](#), [49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- output_multiplier_create, [36](#), [59](#)
- prd_ava, [13](#), [29](#), [30](#), [60](#), [62](#)
- prd_use, [13](#), [29](#), [30](#), [61](#), [61](#)
- primary_input_get, [10](#), [21](#), [27](#), [42](#), [49](#), [52](#), [58](#), [62](#), [65](#), [66](#), [68](#), [73](#), [74](#)
- primary_input_get(), [58](#)
- primary_inputs, [14–17](#), [63](#)
- rows_add, [10](#), [21](#), [27](#), [42](#), [49](#), [52](#), [58](#), [63](#), [64](#), [66](#), [68](#), [73](#), [74](#)
- rows_add(), [65](#)

`supplementary_add`, [6](#), [10](#), [21](#), [27](#), [42](#), [49](#),
[52](#), [58](#), [63](#), [65](#), [65](#), [68](#), [73](#), [74](#)
`supplementary_add()`, [11](#)

`tibble`, [69](#)
`tidyr::pivot_longer()`, [72](#)
`tidyr::pivot_wider()`, [73](#)
`total_tax_add`, [10](#), [21](#), [27](#), [42](#), [49](#), [52](#), [58](#),
[63](#), [65](#), [66](#), [67](#), [73](#), [74](#)

`uk_2010_data`, [25](#), [56](#), [69](#), [72](#)
`uk_2010_results_get`, [70](#)
`uk_test_results`, [25](#), [56](#), [70](#), [71](#)

`vector_transpose`
(vector_transpose_longer), [72](#)
`vector_transpose_longer`, [10](#), [21](#), [27](#), [42](#),
[49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [72](#), [74](#)
`vector_transpose_wider`, [10](#), [21](#), [27](#), [42](#),
[49](#), [52](#), [58](#), [63](#), [65](#), [66](#), [68](#), [73](#), [73](#)