

# Package ‘iotables’

January 8, 2024

**Type** Package

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

**Date** 2024-01-08

**Version** 0.9.3

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

**Description** Pre-processing and basic analytical tasks related to working with Eurostat's symmetric input-output tables and provide basic input-output economics calculations. The package is part of rOpenGov <<http://ropengov.github.io/>> to open source open government initiatives.

**URL** <https://iotables.dataobservatory.eu/>

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

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

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

**Suggests** testthat,  
rmarkdown,  
spelling,  
covr,  
roxyglobals

**Depends** R(>= 3.5.0)

**VignetteBuilder** knitr

**Language** en-US

**Roxygen** list(roclets = c("`collate", ``namespace", ``rd",  
``roxyglobals::global\_roclet"))

**Config/roxyglobals/filename** globals.R

**Config/roxyglobals/unique** FALSE

## R topics documented:

airpol_get . . . . .	3
backward_linkages . . . . .	4
coefficient_matrix_create . . . . .	5
conforming_vector_create . . . . .	6
croatia_2010_1700 . . . . .	7
croatia_2010_1800 . . . . .	8
croatia_2010_1900 . . . . .	9
croatia_employment_2013 . . . . .	10
croatia_employment_aggregation . . . . .	10
direct_effects_create . . . . .	11
employment_get . . . . .	12
employment_metadata . . . . .	13
empty_remove . . . . .	14
equation_solve . . . . .	14
forward_linkages . . . . .	15
germany_1995 . . . . .	16
germany_airpol . . . . .	17
ghosh_inverse_create . . . . .	18
household_column_find . . . . .	19
household_column_get . . . . .	19
indirect_effects_create . . . . .	20
input_coefficient_matrix_create . . . . .	20
input_flow_get . . . . .	21
input_indicator_create . . . . .	22
input_multipliers_create . . . . .	23
iotables_download . . . . .	24
iotables_metadata_get . . . . .	26
iotables_read_tempdir . . . . .	27
iotable_get . . . . .	28
iotable_year_get . . . . .	30
is_html_output . . . . .	31
is_latex_output . . . . .	31
key_column_create . . . . .	32
leontief_inverse_create . . . . .	32
leontief_matrix_create . . . . .	33
matrix_round . . . . .	34
metadata . . . . .	34
metadata_uk_2010 . . . . .	35
multiplier_create . . . . .	36
netherlands_2006 . . . . .	37

output_coefficient_matrix_create . . . . .	38
output_get . . . . .	39
output_multiplier_create . . . . .	39
primary_inputs . . . . .	40
primary_input_get . . . . .	41
rows_add . . . . .	41
supplementary_add . . . . .	42
total_tax_add . . . . .	44
uk_2010_data . . . . .	45
uk_2010_results_get . . . . .	46
uk_test_results . . . . .	46
vector_transpose_longer . . . . .	47
vector_transpose_wider . . . . .	48

<b>Index</b>	<b>50</b>
--------------	-----------

---

airpol_get	<i>Get air pollutant data</i>
------------	-------------------------------

---

## Description

Get air emissions accounts by NACE Rev. 2 activity for environmental impact assessments.

## Usage

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

## Arguments

airpol	The code of the air pollutant. Defaults GHG, ACG, CH4, CH4_CO2E, CH4_NMVOCE, CO, CO2, CO2_BIO, CO_NMVOCE, GHG, HFC_CO2E, N2O, N2O_CO2E, NF3_SF6_CO2E, NH3, NH3_SO2E, NMVOC, NOX, NOX_NMVOCE, NOX_SO2E, O3PR, PFC_CO2E, PM10, PM2_5, SOX_SO2E.
geo	The country code. The special value 'germany_1995' will return the replication dataset <a href="#">germany_airpol</a> .
year	The year. The average employment will be created for the given year, starting with 2008, when the NACE Rev 2 was introduced in employment statistics.
unit	Defaults to "THS_T" (thousand tons.)
data_directory	Defaults to NULL, if a valid directory, it will try to save the pre-processed data file here with labelling.
force_download	Defaults to TRUE. If FALSE it will use the existing downloaded file in the data_directory or the temporary directory, if it exists.

**Details**

Currently tested only with product x product tables. The dataset air emissions accounts by NACE Rev. 2 activity [env\_ac\_ainah\_r2] has five dimensions: The Air pollutant airpol variables are collected on the emissions of the following pollutants: carbon dioxide without emissions from biomass (CO2), carbon dioxide from biomass (Biomass CO2), nitrous oxide (N2O), methane (CH4), perfluorocarbons (PFCs), Hydrofluorocarbons (HFCs), sulphur hexafluoride (SF6) including nitrogen trifluoride (NF3), nitrogen oxides (NOx), Non-methane volatile organic compounds, (NMVOC), carbon monoxide (CO), Particulate matter smaller than 10 micrometre (PM10), Particulate matter smaller than 2,5 micrometre (PM2,5), Sulphur dioxide (SO2), Ammonia (NH3).

See [Reference Metadata in Single Integrated Metadata Structure \(SIMS\)](#) for further details, particularly on the calculation of Global warming potential GHG, Acidifying gases ACG and Tropospheric ozone precursors O3PR.

**Value**

A data.frame with auxiliary metadata to conform the symmetric input-output tables.

**Source**

Eurostat folder [Air emissions accounts by NACE Rev. 2 activity](#)

**See Also**

Other import functions: [employment\\_get\(\)](#), [iotables\\_download\(\)](#), [iotables\\_metadata\\_get\(\)](#), [iotables\\_read\\_tempdir\(\)](#)

**Examples**

```
airpol_get(airpol = "CO2", geo="germany_1995", year = 1995, unit = "THS_T")
```

---

backward_linkages	<i>Backward linkages</i>
-------------------	--------------------------

---

**Description**

Indicate the interconnection of a particular sector to other sectors from which it purchases inputs (demand side). When a sector increases its output, it will increase the total (intermediate) demand on all other sectors, which is measured by backward linkages.

**Usage**

```
backward_linkages(Im)
```

**Arguments**

Im                    A Leontief inverse matrix created by the [leontief\\_inverse\\_create](#) function.

**Details**

Backward linkages are defined as the column sum of the Leontief inverse, in line with the Eurostat Manual of Supply, Use and Input-Output Tables (see p506-507.) and the Handbook on Supply and Use Tables and Input-Output Tables with Extensions and Applications of the United Nations (see p636,)

**Value**

The vector of industry (product) backward linkages in a wide data.frame class, following the column names of the Leontief inverse matrix.

**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)
```

---

coefficient\_matrix\_create

*Create a coefficient matrix*

---

**Description**

Create a coefficient matrix from a Symmetric Input-Output Table.

**Usage**

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

**Arguments**

data_table	A symmetric input-output table, a use table, a margins or tax table retrieved by the <a href="#">iotable_get</a> function.
total	Usually an output vector with a key column, defaults to "output" which equals "P1" or "output_bp". You can use other rows for comparison, for example "TS_BP" if it exists in the matrix.
digits	An integer showing the precision of the technology matrix in digits. Default is NULL when no rounding is applied.
remove_empty	Defaults to TRUE. If you want to keep empty primary input rows, choose FALSE. Empty product/industry rows are always removed to avoid division by zero error in the analytic functions.
households	Defaults to NULL. Household column can be added with TRUE.
return_part	Defaults to NULL. You can choose "product" or "industry" to return an input coefficient matrix or "primary_inputs" to get only the total intermediate use and proportional primary inputs.

**Details**

The coefficient matrix is related by default to output, but you can change this to total supply or other total aggregate if it exists in your table.

**Value**

A data.frame that contains the matrix of data\_table divided by total with a key column. Optionally the results are rounded to given digits.

**References**

See [United Kingdom Input-Output Analytical Tables 2010](#) for explanation on the use of the Coefficient matrix.

**See Also**

Other indicator functions: [direct\\_effects\\_create\(\)](#), [input\\_indicator\\_create\(\)](#)

**Examples**

```
coefficient_matrix_create(data_table = iotable_get(source = "germany_1995"),
                          total = "output",
                          digits = 4 )
```

---

conforming\_vector\_create

*Create an empty conforming vector*

---

**Description**

This helper function creates you a named vector that conforms your analytical objects, such as the use table, the Leontief-matrix, etc. With 60x60 matrixes it is easy to make mistakes with manual definition. The empty effects vector can be used in .csv format as a sample to import scenarios from a spreadsheet application.

**Usage**

```
conforming_vector_create(data_table)
```

**Arguments**

data\_table      A use table, Leontief-matrix, Leontief-inverse, a coefficient matrix or other named matrix / vector.

**Value**

A wide-format conforming vector of data frame class, with column names matching the metadata of the data\_table.

**See Also**

Other iotables processing functions: [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 (data_table = de_input_flow)
```

---

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

---

**Description**

1700 - Symmetric input-output table at basic prices (product x product) In thousand kunas (T\_NAC)

**Usage**

```
data(croatia_2010_1700)
```

**Format**

A data frame with 13 variables.

**t\_rows2** Technology codes in row names, following the Eurostat convention.

**t\_rows2\_lab** Longer labels for t\_rows2

**t\_cols2** Technology codes in column names, following the Eurostat convention.

**t\_cols2\_lab** Longer labels for t\_cols2

**iotables\_col** The standardized iotables column labelling for easier reading.

**col\_order** The column ordering to keep the matrix legible.

**row\_order** The row ordering to keep the matrix legible.

**iotables\_row** The standardized iotables row labelling for easier reading.

**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

**values** The actual values of the table 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      *Input-output table for Croatia, 2010.*

---

### Description

1800 - Symmetric input-output table for domestic production (product x product) In thousand kunas (T\_NAC)

### Usage

```
data(croatia_2010_1800)
```

### Format

A data frame with 13 variables.

**t\_rows2** Technology codes in row names, following the Eurostat convention.

**t\_rows2\_lab** Longer labels for t\_rows2

**values** The actual values of the table in thousand kunas

**t\_cols2** Column labels, following the Eurostat convention with differences. CPA\_ suffix added to original DZS column names.

**t\_cols2\_lab** Longer labels for t\_cols2

**iotables\_col** The standardized iotables column labelling for easier reading.

**col\_order** The column ordering to keep the matrix legible.

**iotables\_row** The standardized iotables row labelling for easier reading.

**row\_order** The 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      *Input-output table for Croatia, 2010.*

---

### Description

1900 - Symmetric input-output table for imports (product x product) In thousand kunas (T\_NAC)

### Usage

```
data(croatia_2010_1900)
```

### Format

A data frame with 13 variables.

**t\_rows2** Technology codes in row names, following the Eurostat convention.

**t\_rows2\_lab** Longer labels for t\_rows2

**values** The actual values of the table in thousand kunas

**t\_cols2** Column labels, following the Eurostat convention with differences. CPA\_ suffix added to original DZS column names.

**t\_cols2\_lab** Longer labels for t\_cols2

**iotables\_col** The standardized iotables column labelling for easier reading.

**col\_order** The column ordering to keep the matrix legible.

**iotables\_row** The standardized iotables row labelling for easier reading.

**row\_order** The 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 the year 2013*

---

### Description

Aggregate Croatian detailed employment statistics into the Croatian (EU standard) Symmetric input-output table format.

### Usage

```
data(croatia_employment_2013)
```

### Format

A data frame with 107 observations in 2 variables:

**code** Short labels

**iotables\_row** iotables style labels

**employment** Employment in the sector in Croatia, 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

Aggregate Croatian detailed employment statistics into the Croatian (EU standard) Symmetric input-output table format.

### Usage

```
data(croatia_employment_aggregation)
```

### Format

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

**employment\_label** Labelling in DZS English language export

**t\_cols2** Labelling of EU/DZS SIOTs.

### 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\(\)](#), [input\\_indicator\\_create\(\)](#)

## Examples

```
n1 <- netherlands_2006

input_coeff_n1 <- input_coefficient_matrix_create(
  data_table = netherlands_2006,
  households = FALSE)

compensation_indicator <- input_indicator_create(netherlands_2006, 'compensation_employees')

I_n1 <- leontief_inverse_create( input_coeff_n1 )

direct_effects_create(input_requirements = compensation_indicator,
  inverse = I_n1)
```

---

employment_get	<i>Get employment data</i>
----------------	----------------------------

---

### Description

Download the employment data for a country and arrange it to the 64x64 SIOTs.

### Usage

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

### Arguments

geo	The country code.
year	The year. The average employment will be created for the given year, starting with 2008, when the NACE Rev 2 was introduced in employment statistics.
sex	Defaults to "Total". Enter "Females" or "F" for female employment, "Males" or "M" for male employment.
age	Defaults to "Y_GE15", which is the Eurostat code for employment in all age groups starting from 15-years-old. Any Eurostat code can be used as a parameter.
labelling	Either "iotables" or the applicable short code, for product x product SIOTs "prod_na" and in the case of industry x industry SIOTs "induse".
data_directory	Defaults to NULL, if a valid directory, it will try to save the pre-processed data file here with labelling.
force_download	Defaults to FALSE. It will use the existing downloaded file in the data_directory or the temporary directory, if it exists.

### Details

Currently works only with product x product tables.

### Value

A data.frame with auxiliary metadata to conform the symmetric input-output tables.

### Source

Eurostat statistic [Employment by sex, age and detailed economic activity \(from 2008 onwards, NACE Rev. 2 two digit level\) - 1 000](#)

## See Also

Other import functions: [airpol\\_get\(\)](#), [iotables\\_download\(\)](#), [iotables\\_metadata\\_get\(\)](#), [iotables\\_read\\_tempdi](#)

## Examples

```
## Not run:
io_tables <- get_employment (
  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

An arrangement of the Eurostat national accounts vocabulary to match with employment statistics data.

## Usage

```
data(employment_metadata)
```

## Format

A data frame with 6 variables.

**emp\_code** code 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, i.e. t\_rows, t\_cols, prod\_na, induse

**group** Different from Eurostat tables, in thousand national currency units.

**iotables\_label** Custom, machine\_readable snake format variable names

## See Also

Other Metadata datasets: [metadata\\_uk\\_2010](#), [metadata](#)

---

empty_remove	<i>Symmetrically remove empty rows and columns</i>
--------------	--

---

**Description**

Symmetrically remove columns with only zero values or with missing values.

**Usage**

```
empty_remove(data_table)
```

**Arguments**

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

**Value**

A tibble/data.frame with a key row and a symmetric matrix, after removing all empty columns and rows at the same time.

**Examples**

```
test_table <- input_coefficient_matrix_create(iotable_get(source = "germany_1995"))
test_table[, 2] <- 0
empty_remove (test_table)
```

---

equation_solve	<i>Solve a basic (matrix) equation</i>
----------------	--

---

**Description**

The function matches to parts of the matrix equation, using the named formats with row names and solves the matrix equation.

**Usage**

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

**Arguments**

LHS	A left-hand side vector with a key column containing the industry or product names for matching, for example the employment coefficients.
Im	A Leontief-inverse with a key column containing the industry or product names for matching.

**Details**

This function is used in wrapper functions, such as [multiplier\\_create](#). to solve particular problems, but it can be used directly, too. The function only performs the lhs pairing industries and checking for exceptions.

**Value**

A data.frame with auxiliary metadata to conform the symmetric input-output tables.

**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**

The increased output of a sector indicates that additional amounts of products are available to be used as inputs by other sectors which can increase their production, which is captured in this indicator vector.

**Usage**

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

**Arguments**

output_coefficient_matrix	An output coefficient matrix created with the <a href="#">output_coefficient_matrix_create</a> function.
digits	Number of decimals for rounding, defaults to NULL.

**Details**

Forward linkages as defined by 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 p637.

**Value**

The vector of industry (product) forward linkages in a long-form data.frame, containing the meta-data column of the the row names from the output\_coefficient\_matrix.

**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

*Simple input-output table for Germany, 1995.***Description**

Replication data taken from the Eurostat Manual, Table 15.4: Input-output table of domestic output at basic prices (Version A)

**Usage**

```
data(germany_1995)
```

**Format**

A data frame with 228 observations and 10 variables.

**prod\_na** Technology codes in row names, following the Eurostat convention.

**prod\_na\_lab** Longer labels for t\_rows2.

**induse** Column labels, following the Eurostat convention with differences.

**iotables\_row** Row labels, i.e. to be used in key column, for iotables package abbreviations.

**iotables\_col** Column labels for iotables package abbreviations.

**values** The actual values of the table in million euros.

**unit** MIO\_EUR, the same as Eurostat.

**unit\_lab** Million euros. Eurostat usually has euro and national currency unit values, too.

**geo** ISO/Eurostat country code for Germany, i.e. DE.

**geo\_lab** ISO/Eurostat country name, Germany.

**time** Date of the SIOT.

**Details**

For testing and documentation purposes a well documented example is taken the Eurostat Manual. The table in the Eurostat manual is brought to the format used by the Eurostat database. It is a small dataset for examples, but it is also instructive to understand how Eurostat stores the highly structured SIOTs in long-form tidy datasets. The labels were slightly altered to reflect the transition from the vocabulary of ESA95 to ESA2010 since the publication of the Manual.

**Source**

[Eurostat Manual of Supply, Use and Input-Output Tables](#) p 482



**See Also**

Other Validation datasets: [germany\\_airpol](#), [netherlands\\_2006](#), [uk\\_2010\\_data](#), [uk\\_test\\_results](#)

---

germany\_airpol

*Air Pollution Table for Germany, 1995.*

---

**Description**

Air pollution values for validation.

**Usage**

```
data(germany_airpol)
```

**Format**

A data frame with 72 observations and 4 variables.

**airpol** The abbreviation of the air pollutant.

**induse** Column labels, following the Eurostat convention with differences.

**iotables\_col** Column labels for iotables package abbreviations.

**value** The actual values of the table in thousand tons.

**Details**

For testing purposes and cross-checking with the Eurostat manual. The labels were slightly altered to reflect the transition from the vocabulary of ESA95 to ESA2010 since the publication of the Manual.

**Source**

[Eurostat Manual of Supply, Use and Input-Output Tables](#) p 482.

**See Also**

Other Validation datasets: [germany\\_1995](#), [netherlands\\_2006](#), [uk\\_2010\\_data](#), [uk\\_test\\_results](#)

---

ghosh\_inverse\_create *Create the inverse of a Ghosh-matrix*

---

### Description

Create the Ghosh-inverse from the output coefficients.

### Usage

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

### Arguments

`output_coefficients_matrix`  
A technology coefficient matrix created by the [output\\_coefficient\\_matrix\\_create](#).

`digits`  
An integer showing the precision of the technology matrix in digits. Default is NULL when no rounding is applied.

### Details

The Ghosh-inverse is

$$G = (I - B)^{-1}$$

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](#) pp 622–639.

For the similar inverse created from input coefficients, see the [leontief\\_inverse\\_create](#) function.

### See Also

Other analytic object functions: [input\\_flow\\_get\(\)](#), [leontief\\_inverse\\_create\(\)](#), [leontief\\_matrix\\_create\(\)](#)

### Examples

```
om <- output_coefficient_matrix_create(
  data_table = iotable_get()
)

ghosh_inverse_create( output_coefficients_matrix = om )
```

---

household\_column\_find *Return the position of final household expenditure*

---

**Description**

Return the position of final household expenditure

**Usage**

```
household_column_find(data_table)
```

**Arguments**

data\_table      A symmetric input output table, a use table or a supply table.

**Value**

An integer value with the final household expenditure. Returns NULL if not found.

**Examples**

```
household_column_find( iotable_get ( source = 'germany_1995' ) )
```

---

household\_column\_get *Return Final Household Expenditure*

---

**Description**

Return Final Household Expenditure

**Usage**

```
household_column_get(data_table)
```

**Arguments**

data\_table      A symmetric input output table, a use table or a supply table.

**Value**

The column containing final household expenditure. If not found NULL is returned.

**See Also**

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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'))
```

---

indirect\_effects\_create

*Create indirect effects*

---

### Description

The function creates the indirect effects vector.

### Usage

```
indirect_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 indirect effects and the necessary metadata to sort them or join them with other matrixes.

### Examples

```
n1 <- netherlands_2006

input_coeff_n1 <- input_coefficient_matrix_create(
  data_table = netherlands_2006,
  households = FALSE)

compensation_indicator <- input_indicator_create(netherlands_2006, 'compensation_employees')

I_n1 <- leontief_inverse_create(input_coeff_n1)

indirect_effects_create(input_requirements = compensation_indicator,
  inverse = I_n1)
```

---

input\_coefficient\_matrix\_create

*Create an input coefficient matrix*

---

### Description

Create an input coefficient matrix from the input flow matrix and the output vector. The two input vectors must have consistent labelling, i.e the same column names must be found in the use table (input flow) and the output vector.

**Usage**

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

**Arguments**

data_table	A symmetric input-output table, a use table, a margins or tax table retrieved by the <a href="#">iotable_get</a> function.
households	Defaults to NULL. Household column can be added with TRUE.
digits	An integer showing the precision of the technology matrix in digits. Default is NULL when no rounding is applied.

**Details**

The input coefficients of production activities may be interpreted as the corresponding cost shares for products and primary inputs in total output. Our terminology follows the [Eurostat Manual of Supply, Use and Input-Output Tables](#). Input-Output Multipliers Specification Sheet and Supporting Material, Spicosa Project Report, which cannot be linked due to a malformed url, but can be found with a search engine. this matrix is called 'technological coefficients'. The results of the function are tested on both sources. This is a wrapper function around [coefficient\\_matrix\\_create](#).

**Value**

A data frame that contains the matrix of first quadrant of the use table as input\_flow divided by output supported by a key column of product or industries, with a key column. Optionally the results are rounded to given digits.

An input coefficient matrix of data.frame class. The column names are ordered, and the row names are in the first, auxiliary metadata column.

**Examples**

```
input_coefficient_matrix_create (
  iotable_get(),
  digits = 4 )

#This is a wrapper function and equivalent to

coefficient_matrix_create( iotable_get(),
  total = "total",
  return = "products")
```

---

input_flow_get	<i>Create an inter-industry or input flow matrix</i>
----------------	--

---

**Description**

Select the first quadrant of the symmetric input-output table.

**Usage**

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

**Arguments**

data_table	A symmetric input-output table or use table retrieved by the <a href="#">iotable_get</a> function.
empty_remove	Defaults to TRUE. If you want to keep empty primary input rows, choose FALSE. Empty product/industry rows are always removed to avoid division by zero error in the analytic functions.
households	Defaults to FALSE. If TRUE, the final household expenditure is added to the input flow table.

**Details**

The first quadrant is called the input flow matrix, or the input requirements matrix, or the inter-industry matrix in different contexts.

**Value**

A data flow matrix (a symmetric use table) with a key column.

**See Also**

Other analytic object functions: [ghosh\\_inverse\\_create\(\)](#), [leontief\\_inverse\\_create\(\)](#), [leontief\\_matrix\\_create\(\)](#)

**Examples**

```
input_flow <- input_flow_get(data_table = iotable_get(),
                             empty_remove = FALSE,
                             households = TRUE)
```

---

input\_indicator\_create

*Create input indicator(s)*

---

**Description**

The function creates the input indicators from the inputs and the outputs.

**Usage**

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

**Arguments**

data_table	A symmetric input-output table, a use table, a margins or tax table retrieved by the <a href="#">iotable_get</a> function.
input_row	The name of input(s) for which you want to create the indicator(s). Must be present in the data_table.
digits	Rounding digits, if omitted, no rounding takes place.
households	If the households column should be added, defaults to FALSE.
indicator_names	The names of new indicators. Defaults to NULL when the names in the key column of input_matrix will be used to create the indicator names.

**Value**

A tibble (data frame) containing the input\_matrix divided by the output\_vector with a key column for products or industries.

**See Also**

Other indicator functions: [coefficient\\_matrix\\_create\(\)](#), [direct\\_effects\\_create\(\)](#)

**Examples**

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

---

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 <a href="#">input_indicator_create</a>
Im	A Leontief-inverse created by <a href="#">leontief_inverse_create</a> .
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: [multiplier\\_create\(\)](#)

**Examples**

```
n1 <- netherlands_2006

input_coeff_n1 <- input_coefficient_matrix_create(
  data_table = netherlands_2006,
  households = FALSE)

compensation_indicator <- input_indicator_create(netherlands_2006, 'compensation_employees')

I_n1 <- leontief_inverse_create(input_coeff_n1)

input_multipliers_create(input_requirements = compensation_indicator,
  Im = I_n1)
```

---

iotables\_download      *Download input-output tables*

---

**Description**

This function downloads standard input-output table files. Currently only Eurostat files are supported. You are not likely to use this function, because [iotable\\_get](#) will call this function if necessary and properly filter out an input-output table.

**Usage**

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



**Arguments**

- `source` See the available list of sources above in the Description.
- `data_directory` Defaults to NULL when the files will be temporarily stored in the path retrieved by `tempdir`. If it is a different valid directory, it will try to save the pre-processed data file here with labelling.
- `force_download` Defaults to FALSE which will use the existing downloaded file in the `data_directory` or the temporary directory, if it exists. TRUE will try to download the file from the Eurostat warehouse.

**Details**

The data is downloaded in the `tempdir()` under the name the statistical product as an rds file. (For example: `naio_10_cp1750.rds`) The temporary directory is emptied at every normal R session exit.

To save the file for further use (which is necessary in analytical work because download times are long) set the `download_directory` [see parameters]. The function will make a copy of the rds file in this directory.

- `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` column. The data is saved into the actual `tempdir()`, too.

**See Also**

Other import functions: `airpol_get()`, `employment_get()`, `iotables_metadata_get()`, `iotables_read_tempdir()`

**Examples**

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

---

iotables\_metadata\_get *Get Metadata from Nested iotables File*

---

### Description

Remove the data column and return only the metadata information of input-output (or related tables) from a source. If `dat` is not inputted as a nested data frame created by `iotables_download`, validate the source input parameter and try to load the table from the current sessions' temporary directory.

`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

### Usage

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

### Arguments

<code>dat</code>	A nested data file created by <code>iotables_download</code> . Defaults to <code>NULL</code> in which case an attempt is made to find and read in the nested data from the current R sessions' temporary directory.
<code>source</code>	See the available list of sources above in the Description.

### Value

A data frame, which contains the metadata of all available input-output tables from a specific source.

### See Also

Other import functions: `airpol_get()`, `employment_get()`, `iotables_download()`, `iotables_read_tempdir()`

**Examples**

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

# Now you can get the metadata:
iotables_metadata_get(source = "naio_10_pyp1750")
```

---

`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**

`source` See the available list of sources above in the Description. Defaults to `source = "naio_10_cp1700"`.

**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` column.

**See Also**

Other import functions: [airpol\\_get\(\)](#), [employment\\_get\(\)](#), [iotables\\_download\(\)](#), [iotables\\_metadata\\_get\(\)](#)

**Examples**

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

iotables_read_tempdir (source = "naio_10_pyp1750")
```

---

iotable\_get

*Get An Input-Output Table From Bulk File*


---

**Description**

This function is used to filter out a single input-output table from a database, for example a raw file downloaded from the Eurostat website. It provides some functionality to avoid some pitfalls.

**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**

labelled\_io\_data

If you have downloaded a bulk data file with [iotables\\_download](#), it is faster to work with the data in the memory. Defaults to NULL when the data will be retrieved from the hard disk or from the Eurostat website invoking the same function.

source

A data source, for example `naio_10_cp1700`.

`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

For further information consult the [Eurostat Symmetric Input-Output Tables](#) page.

geo A country code or a country name. For example, SK or as Slovakia.

year A numeric variable containing the year. Defaults to 2010, because this year has the most data.

unit A character string containing the currency unit, defaults to MIO\_NAC (million national currency unit). The alternative is MIO\_EUR.

stk\_flow Defaults to DOM as domestic output, alternative IMP for imports and TOTAL for total output. For source = 'naio\_10\_cp1620' and trade and transport margins and source = 'naio\_10\_cp1630' taxes less subsidies only TOTAL is not used.

labelling Defaults to iotables which gives standard row and column names regardless of the source of the table, or if it is a product x product, industry x industry or product x industry table. The alternative is short or eurostat which is the original short row or column code of Eurostat or OECD.

data\_directory Defaults to NULL, if a valid directory, it will try to save the pre-processed data file here with labelling.

force\_download Defaults to TRUE. If FALSE it will use the existing downloaded file in the data\_directory or the temporary directory, if it exists. Will force download only in a new session.

## Details

Unless you want to work with bulk data files, you should not invoke `iotables_download` directly, rather via this function, if and when it is necessary.

## Value

A wide format data.frame with a well-ordered input-output table. The bulk data files on the Eurostat website are in a long form and they are not correctly ordered for further matrix equations.

## Examples

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

---

iotable\_year\_get      *Get the available years from bulk downloaded input-output tables*

---

### Description

The function selects the available tables by year or time as a date for a specific country and currency unit in the Eurostat bulk file.

### 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

labelled_io_data	If you have downloaded a bulk data file with <a href="#">iotables_download</a> , it is faster to work with the data in the memory. Defaults to NULL when the data will be retrieved from the hard disk or from the Eurostat website invoking the same function.
source	A data source, for example naio_10_cp1700. Symmetric input-output table at basic prices (product by product) (naio_10_cp1700) Symmetric input-output table at basic prices (industry by industry) (naio_10_cp1750) Symmetric input-output table at basic prices (product by product) (previous years prices) (naio_10_pyp1700) Symmetric input-output table at basic prices (industry by industry) (previous years prices) (naio_10_pyp1750) Table of trade and transport margins at basic prices (naio_10_cp1620) and at previous' years prices (naio_10_pyp1620) Table of taxes less subsidies on products at basic prices (naio_10_cp1630) and at previous' years prices (naio_10_pyp1630) For further information consult the <a href="#">Eurostat Symmetric Input-Output Tables</a> page.
geo	A country code or a country name. For example, SK or as Slovakia.
unit	A character string containing the currency unit, defaults to MIO_NAC (million national currency unit). The alternative is MIO_EUR.
time_unit	Defaults to 'year' and years are returned as numbers. Alternative is to return 'time' as vector of dates.
stk_flow	Defaults to DOM as domestic output, alternative IMP for imports and TOTAL for total output. For source = 'naio_10_cp1620' and trade and transport margins and source = 'naio_10_cp1630' taxes less subsidies only TOTAL is not used.
data_directory	Defaults to NULL. Use if it you used a data_directory parameter with <a href="#">iotable_get</a> or <a href="#">iotables_download</a> .

`force_download` Defaults to TRUE. If FALSE it will use the existing downloaded file in the `data_directory` or the temporary directory, if it exists. Will force download only in a new session.

### Details

Unless you want to work with bulk data files, you should not invoke `iotables_download` directly, rather via this function, if and when it is necessary.

### Value

A vector with the years that have available input-output tables.

### See Also

Other iotables processing functions: `conforming_vector_create()`, `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" )
```

---

<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

---

key_column_create	<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\(\)](#), [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	<i>Create the inverse of a Leontief-matrix</i>
-------------------------	--

---

**Description**

Create the Leontief inverse from the technology coefficient matrix.

**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 the [input\\_coefficient\\_matrix\\_create](#).
- `digits`  
An integer showing the precision of the technology matrix in digits. Default is NULL when no rounding is applied.

**Details**

The Leontief-inverse is

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

where B is the input coefficient matrix created by [input\\_coefficient\\_matrix\\_create](#). For the similar inverse created from output coefficients, see the [ghosh\\_inverse\\_create](#) function.

**See Also**

Other analytic object functions: [ghosh\\_inverse\\_create\(\)](#), [input\\_flow\\_get\(\)](#), [leontief\\_matrix\\_create\(\)](#)

**Examples**

```
tm <- input_flow_get (
  data_table = iotable_get(),
  households = FALSE)
I <- leontief_inverse_create( technology_coefficients_matrix = tm )
```

---

leontief\_matrix\_create

*Create a Leontief matrix*

---

**Description**

Create a Leontief matrix from technology matrix after some basic error handling. Most likely you will need this function as a step to invoke the function to create its inverse: [leontief\\_inverse\\_create](#).

**Usage**

```
leontief_matrix_create(technology_coefficients_matrix)

leontieff_matrix_create(technology_coefficients_matrix)
```

**Arguments**

- `technology_coefficients_matrix`  
A technology coefficient matrix created by the [input\\_coefficient\\_matrix\\_create](#) or [output\\_coefficient\\_matrix\\_create](#).

**Value**

A Leontief matrix of data.frame class. The column names are ordered, and the row names are in the first, auxiliary metadata column.

**See Also**

Other analytic object functions: [ghosh\\_inverse\\_create\(\)](#), [input\\_flow\\_get\(\)](#), [leontief\\_inverse\\_create\(\)](#)

**Examples**

```
tm <- input_flow_get (
  data_table = iotable_get(),
  households = FALSE)
L <- leontief_matrix_create( technology_coefficients_matrix = tm )
```

---

matrix\_round

*Round all matrix values to required number of digits.*

---

**Description**

For comparison with results created with other software or published with rounding, systematically round the values of an input-output table, a use, supply, tax or margins table.

**Usage**

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

**Arguments**

`data_table` A symmetric input output table, a use, supply, tax or margins table.  
`digits` An integer number, defaults to 0.

**Value**

The matrix, with the intact key column and the numeric columns rounded.

**See Also**

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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\(\)](#)

---

metadata

*Metadata*

---

**Description**

An arrangement of the Eurostat national accounts vocabulary, used to correctly order wide format rows and columns from bulk long-form tables.

**Usage**

```
data(metadata)
```

**Format**

A data frame with 8 variables.

**variable** Eurostat vocabulary source, i.e. t\_rows, t\_cols, prod\_na, induce

**group** Informal labelling for macroeconomic groups

**code** Eurostat labels

**label** Eurostat label descriptions

**quadrant** Where to place the data from a long-form raw data file

**account\_group** Different from Eurostat tables, in thousand national currency units.

**numeric\_label** ordering from quadrant, account\_group, digit\_1, digit\_2

**iotables\_label** Custom, machine\_readable snake format variable names

**See Also**

Other Metadata datasets: [employment\\_metadata](#), [metadata\\_uk\\_2010](#)

---

metadata_uk_2010	<i>Multipliers and effects (product) for testing from the United Kingdom Input-Output Analytical Tables, 2010</i>
------------------	---

---

**Description**

The Excel-imported UK data.

**Usage**

```
data(uk_2010_data)
```

**Format**

A data frame with 10 variables.

**variable** Constant for the iotable\_get function.

**uk\_row** The UK row identifier. Dots and '&' converted to '-'.

**uk\_col** The UK row identifier. Dots and '&' converted to '-'.

**uk\_row\_label** The original UK row labels.

**uk\_col\_label** The original UK column labels.

**eu\_prod\_na** The Eurostat vocabulary equivalent of uk\_row

**row\_order** Ordering variable for rows.

**col\_order** Ordering variable for columns.

**prod\_na** The Eurostat-like key values for rows.

**induse** The Eurostat-like column names

**See Also**

Other Metadata datasets: [employment\\_metadata](#), [metadata](#)

---

multiplier\_create      *Create multipliers*

---

### Description

This function is in fact a wrapper around the [equation\\_solve](#) function, adding a key column with the name to the multiplier to maintain structural consistency.

### Usage

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

### Arguments

input_vector	An input matrix or vector created by the <a href="#">input_indicator_create</a> function.
Im	The Leontief inverse as a named object created by the <a href="#">leontief_inverse_create</a> function.
multiplier_name	A variable name to be given to the returned multipliers. Defaults to multiplier.
digits	Rounding digits, if omitted, no rounding takes place.

### Details

As opposed to direct effects, multipliers are expressed per input of product/industry.

### Value

A data frame with the vector of multipliers and the an auxiliary metadata column (for joining with other matrixes.)

### See Also

Other multiplier functions: [input\\_multipliers\\_create\(\)](#)

### Examples

```
data_table <- iotable_get()

coeff_de <- input_coefficient_matrix_create( data_table )

de_gva_indicator <- input_indicator_create (
  data_table = data_table,
  input = 'gva') #this is a correct input

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\_2006

*Simple input-output table for the Netherlands, 2006.*


---

### Description

This simplified SIOT is taken from the Science Policy Integration for Coastal Systems Assessment project's input-output multiplier specification sheet. It is used as a simple example SIOT for controlled analytical results. The column names were slightly altered to resemble more the current Eurostat conventions and the main example dataset [germany\\_1995](#).

### Usage

```
data(netherlands_2006)
```

### Format

A data frame with 14 observations and 13 variables.

A data frame of 13 observations in 14 variables.

**prod\_na** Product name, simplified, following the Eurostat conventions

**agriculture\_group** Simple aggregated agricultural products

**mining\_group** Simple aggregated mining products

**manufacturing\_group** Simple aggregated manufacturing products

**construction\_group** Construction

**utilities\_group** Simple aggregated utilities products/services

**services\_group** Simple aggregated services products

**TOTAL** Column / row sums, simple summary, not included in the original source

**final\_consumption\_private** Simple aggregated final private use

**final\_consumption\_households** Simple aggregated final household consumption

**final\_consumption\_government** Simple aggregated final government consumption

**gross\_fixed\_capital\_formation** Gross fixed capital formation 'GFCF'

**exports** Simple aggregated exports

**total\_use** Simple aggregated total use

### Source

Source: Input-Output Multipliers Specification Sheet and Supporting Material in the Spicosa Project Report

### See Also

Other Validation datasets: [germany\\_1995](#), [germany\\_airpol](#), [uk\\_2010\\_data](#), [uk\\_test\\_results](#)

---

output\_coefficient\_matrix\_create

*Create an output coefficient matrix*

---

### Description

Create an output coefficient matrix from the input flow matrix or a symmetric input-output table.

### Usage

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

### Arguments

data_table	A symmetric input-output table, a use table, a margins or tax table retrieved by the <code>iotable_get</code> . In case you use <code>type="tfu"</code> you need to input a full iotable, create by the <code>iotable_get</code> , because the final demand column is in the second quadrant of the IOT.
total	The <code>output='total'</code> (or <code>CPA_TOTAL</code> , depending on the names in your table, default) returns the output coefficients for products (intermediates) while the <code>final_demand</code> returns output coefficients for final demand. See <a href="#">Eurostat Manual of Supply, Use and Input-Output Tables</a> p495 and p507.
digits	An integer showing the precision of the technology matrix in digits. Default is <code>NULL</code> when no rounding is applied.

### Details

The output coefficients may be interpreted as the market shares of products in total output. If there are zero values in present, they will be changed to 0.000001 and you will get a warning. Some analytical equations cannot be solved with zero elements. You either have faulty input data, or you have to use some sort of data modification to carry on your analysis.

### Value

An output coefficient matrix of `data.frame` class. The column names are ordered, and the row names are in the first, auxiliary metadata column.

### Examples

```
data_table <- iotable_get()

output_coefficient_matrix_create (data_table = data_table,
                                 total = 'tfu',
                                 digits = 4)
```

---

output_get	<i>Get an output vector</i>
------------	-----------------------------

---

**Description**

This is a wrapper function around the [primary\\_input\\_get](#) function.

**Usage**

```
output_get(data_table)
```

**Arguments**

data_table	A symmetric input-output table or use table retrieved by the <a href="#">iotable_get</a> function.
------------	--

**Value**

A data frame with the vector of multipliers and the an auxiliary metadata column (for joining with other matrixes.)

**See Also**

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [household\\_column\\_get\(\)](#), [iotable\\_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_get ( data_table = iotable_get () )
```

---

output_multiplier_create	<i>Create output multipliers</i>
--------------------------	----------------------------------

---

**Description**

Create a data frame of output multipliers.

**Usage**

```
output_multiplier_create(input_coefficient_matrix)
```

**Arguments**

input_coefficient_matrix	A Leontief inverse matrix created by the <a href="#">input_coefficient_matrix_create</a> function.
--------------------------	--

### Details

Output multipliers as defined by the Eurostat Manual of Supply, Use and Input-Output Tables on p500.

### Value

A data frame with a key column and the output multipliers of the industries.

### Examples

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

---

primary\_inputs

*Primary input abbreviations*

---

### 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](#)



---

primary_input_get	<i>Get primary inputs</i>
-------------------	---------------------------

---

**Description**

This function will retrieve any primary input from the input-output table.

**Usage**

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

**Arguments**

`data_table` A symmetric input-output table, a use table, or a supply table retrieved by the [iotable\\_get](#) function.

`primary_input` The primary input to be returned from the table.

**Value**

A data frame with the vector of multipliers and the an auxiliary metadata column (for joining with other matrixes.)

**See Also**

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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**

```
comp_employees_de <- primary_input_get(
  data_table = iotable_get(),
  primary_input = "compensation_employees")
```

---

rows_add	<i>Add conforming row(s)</i>
----------	------------------------------

---

**Description**

Add a conforming row, or elements of a conforming row to a names matrix.

**Usage**

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

**Arguments**

<code>data_table</code>	A symmetric input-output table, a use table, a margins or tax table retrieved by the <code>iotable_get</code> function.
<code>rows_to_add</code>	A data frame or a named numeric vector.
<code>row_names</code>	An optional name or vector of names for the key column. Defaults to NULL.
<code>empty_fill</code>	What should happen with missing column values? Defaults to 0. If you want to avoid division by zero, you may consider a very small value such as 0.000001.

**Details**

If you want to add a single row manually, you can input a named numeric vector or a data frame with a single row. For multiple rows, input them as wide form data frame (see examples.)

**Value**

An extended `data_table` with the new row(s) binded.

**See Also**

Other iotables processing functions: `conforming_vector_create()`, `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" )
```

---

`supplementary_add`      *Add supplementary data*

---

**Description**

Add supplementary data to a SIOT, a use, supply or margins table.

**Usage**

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

**Arguments**

- `data_table` A SIOT, a use table, a supply table, or a margins table.
- `supplementary_data` Supplementary data to be added. It must be a data.frame or tibble with a key column containing the indicator's name, and the column names must match with the `data_table`. Can be a vector or a data frame of several rows.
- `supplementary_names` Optional names for the new supplementary rows. Defaults to NULL.

**Details**

This function is a wrapper around the more general [rows\\_add](#) function.

**Value**

An extended `data_table` with the new row(s) binded.

A symmetric input-output table with supplementary data, of data.frame class. The column names are ordered, and the row names are in the first, auxiliary metadata column.

**See Also**

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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>Summarize and add tax data</i>
---------------	-----------------------------------

---

### Description

Create and add a total tax row, if there are multiple tax rows present in the `data_table`.

### Usage

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

### Arguments

`data_table` A SIOT, a use table, a supply table, or a margins table that has product and production tax rows in among the primary inputs.

`tax_names` Defaults to `( "d21x31", "d29x39" )`, which are the Eurostat names for taxes. The parameter is not case sensitive.

`total_tax_name` Defaults to `'TOTAL_TAX'`. The name of the summarized row. It is case sensitive.

### Value

A data frame with the vector of multipliers and the an auxiliary metadata column (for joining with other matrixes.)

### See Also

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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 (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

Replication data exported from the Office of National Statistics.

### Usage

```
data(uk_2010_data)
```

### Format

A data frame with 10 variables.

**uk\_row** The UK row identifier. Dots and '&' converted to '-'.

**uk\_row\_lab** The original UK row labels.

**uk\_col** The UK row identifier. Dots and '&' converted to '-'.

**uk\_col\_lab** The original UK column labels.

**geo** Eurostat-style geocode, i.e. UK

**geo\_lab** United Kingdom

**indicator** The name of the indicator, i.e. Excel sheet.

**unit** Eurostat label equivalents units, i.e. MIO\_NAC.

**unit\_lab** Eurostat label equivalents, i.e. millions of national currency unit.

**values** The numeric values of the variable

**year** Constant = 2010.

### Details

You can retrieve the data with [iotable\\_get](#), setting the source parameter as follows:

uk\_2010\_siot Input-Output table (domestic use, basic prices, product by product)

uk\_2010\_use Domestic use table at basic prices (product by industry)

uk\_2010\_imports Imports use table at basic prices (product by product)

uk\_2010\_coeff Matrix of coefficients (product by product)

uk\_2010\_inverse Leontief Inverse (product by product)

### Source

[United Kingdom Input-Output Analytical Tables 2010](#)

### See Also

Other Validation datasets: [germany\\_1995](#), [germany\\_airpol](#), [netherlands\\_2006](#), [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             *Multipliers and effects (product) for testing from the United Kingdom Input-Output Analytical Tables, 2010*

---

### Description

The Excel-imported UK data.

### Usage

```
data(uk_test_results)
```

### Format

A data frame with 12 variables.

**uk\_row\_label** The UK row label

**Output multiplier** The imported Output multipliers

**output\_multiplier\_rank** The imported ranking of output multipliers

**Employment cost multiplier** The imported Employment cost multipliers.

**employment\_cost\_multiplier** The imported ranking of Employment cost multipliers.  
**Employment cost effects** The imported Employment cost multipliers.  
**employment\_cost\_effects\_rank** The imported ranking of employment cost multipliers.  
**GVA effects** The imported GVA effects.  
**gva\_effects\_rank** The imported ranking GVA effects.  
**gva\_multiplier\_rank** The imported ranking GVA multipliers.  
**GVA multiplier** The imported GVA multipliers.  
**indicator** Indicator names.

### See Also

Other Validation datasets: [germany\\_1995](#), [germany\\_airpol](#), [netherlands\\_2006](#), [uk\\_2010\\_data](#)

---

vector\_transpose\_longer

*Transpose a vector to a long form*

---

### Description

Many vectors (indicators, multipliers) are create in the wide form to conform matrixes in analytical functions. For printing it is more useful to have them in long form.

### 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 matrix or vector that normally has a key column.
<code>names_to</code>	Defaults to 'nace_r2'.
<code>values_to</code>	Defaults to 'value'.
<code>key_column_name</code>	The name of the first column. Defaults to NULL when it is not changed. It should usually match the key column of the matrix or vector you would like to join the new vector created with <code>vector_transpose_longer</code> .
<code>.keep</code>	Keep the indicator identifier column? Defaults to FALSE.

**Details**

This is a wrapper around [pivot\\_longer](#) so you do not necessarily need to import or load the entire *tidyr* package.

**Value**

A long form vector with a key column, and optionally the identifier of the indicator in the first column.

**See Also**

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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_inidicator",
             agriculture = 0.0123,
             manufacturing = 0.1436,
             trade = 0.0921)
)
```

---

vector\_transpose\_wider

*Transpose a vector to wider format*

---

**Description**

Many vectors (indicators, multipliers) are create in the wide form to conform matrixes in analytical functions. For binding it is more useful to have them in wide format.

**Usage**

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

**Arguments**

**data\_table** A matrix or vector that normally has a key column. If the key column must be created or replaced, used `key_column_name` and `key_column_values`.

**names\_from, values\_from**

A pair of arguments describing which column (or columns) to get the name of the output column ('names\_from'), and which column (or columns) to get the cell values from ('values\_from').



key\_column\_name

The name of the key column.

key\_column\_values

You can explicitly supply key column values. Defaults to NULL when the key column values will be created from the long data.

### Details

This is a wrapper around [pivot\\_wider](#) so you do not necessarily need to import or load the entire *tidyr* package.

### See Also

Other iotables processing functions: [conforming\\_vector\\_create\(\)](#), [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

- \* **Croatia 2010 datasets**
    - croatia\_2010\_1700, 7
    - croatia\_2010\_1800, 8
    - croatia\_2010\_1900, 9
    - croatia\_employment\_2013, 10
    - croatia\_employment\_aggregation, 10
    - primary\_inputs, 40
  - \* **Metadata datasets**
    - employment\_metadata, 13
    - metadata, 34
    - metadata\_uk\_2010, 35
  - \* **Validation datasets**
    - germany\_1995, 16
    - germany\_airpol, 17
    - netherlands\_2006, 37
    - uk\_2010\_data, 45
    - uk\_test\_results, 46
  - \* **analytic object functions**
    - ghosh\_inverse\_create, 18
    - input\_flow\_get, 21
    - leontief\_inverse\_create, 32
    - leontief\_matrix\_create, 33
  - \* **datasets**
    - croatia\_2010\_1700, 7
    - croatia\_2010\_1800, 8
    - croatia\_2010\_1900, 9
    - croatia\_employment\_2013, 10
    - croatia\_employment\_aggregation, 10
    - employment\_metadata, 13
    - germany\_1995, 16
    - germany\_airpol, 17
    - metadata, 34
    - metadata\_uk\_2010, 35
    - netherlands\_2006, 37
    - primary\_inputs, 40
    - uk\_2010\_data, 45
    - uk\_test\_results, 46
  - \* **import functions**
    - airpol\_get, 3
    - employment\_get, 12
    - iotables\_download, 24
    - iotables\_metadata\_get, 26
    - iotables\_read\_tempdir, 27
  - \* **indicator functions**
    - coefficient\_matrix\_create, 5
    - direct\_effects\_create, 11
    - input\_indicator\_create, 22
  - \* **iotables import functions**
    - iotable\_get, 28
  - \* **iotables processing functions**
    - conforming\_vector\_create, 6
    - household\_column\_get, 19
    - iotable\_year\_get, 30
    - key\_column\_create, 32
    - matrix\_round, 34
    - output\_get, 39
    - primary\_input\_get, 41
    - rows\_add, 41
    - supplementary\_add, 42
    - total\_tax\_add, 44
    - vector\_transpose\_longer, 47
    - vector\_transpose\_wider, 48
  - \* **linkage functions**
    - backward\_linkages, 4
    - forward\_linkages, 15
  - \* **multiplier functions**
    - input\_multipliers\_create, 23
    - multiplier\_create, 36
- airpol\_get, 3, 13, 25, 26, 28
- backward\_linkages, 4, 15
- coefficient\_matrix\_create, 5, 11, 21, 23
- conforming\_vector\_create, 6, 19, 31, 32, 34, 39, 41–44, 48, 49
- croatia\_2010\_1700, 7, 8–10, 40
- croatia\_2010\_1800, 7, 8, 9, 10, 40
- croatia\_2010\_1900, 7, 8, 9, 10, 40
- croatia\_employment\_2013, 7–10, 10, 40
- croatia\_employment\_aggregation, 7–10, 10, 40
- direct\_effects\_create, 6, 11, 23
- employment\_get, 4, 12, 25, 26, 28
- employment\_metadata, 13, 35
- empty\_remove, 14

- equation\_solve, 14, 36
- forward\_linkages, 5, 15
- germany\_1995, 16, 17, 37, 45, 47
- germany\_airpol, 3, 17, 17, 37, 45, 47
- ghosh\_inverse\_create, 18, 22, 33, 34
- household\_column\_find, 19
- household\_column\_get, 7, 19, 31, 32, 34, 39, 41–44, 48, 49
- indirect\_effects\_create, 20
- input\_coefficient\_matrix\_create, 20, 33, 39
- input\_flow\_get, 18, 21, 33, 34
- input\_indicator\_create, 6, 11, 20, 22, 24, 36
- input\_multipliers\_create, 23, 36
- iotable\_get, 5, 21–24, 28, 30, 38, 39, 41, 42, 45
- iotable\_year\_get, 7, 19, 30, 32, 34, 39, 41–44, 48, 49
- iotables\_download, 4, 13, 24, 26, 28–31
- iotables\_metadata\_get, 4, 13, 25, 26, 28
- iotables\_read\_tempdir, 4, 13, 25, 26, 27
- is\_html\_output, 31
- is\_latex\_output, 31
- key\_column\_create, 7, 19, 31, 32, 34, 39, 41–44, 48, 49
- leontief\_inverse\_create, 4, 11, 18, 20, 22, 24, 32, 33, 34, 36
- leontief\_matrix\_create, 18, 22, 33, 33
- leontieff\_inverse\_create  
(leontief\_inverse\_create), 32
- leontieff\_matrix\_create  
(leontief\_matrix\_create), 33
- matrix\_round, 7, 19, 31, 32, 34, 39, 41–44, 48, 49
- metadata, 13, 34, 35
- metadata\_uk\_2010, 13, 35, 35
- multiplier\_create, 14, 24, 36
- netherlands\_2006, 17, 37, 45, 47
- output\_coefficient\_matrix\_create, 15, 18, 33, 38
- output\_get, 7, 19, 31, 32, 34, 39, 41–44, 48, 49
- output\_multiplier\_create, 39
- pivot\_longer, 48
- pivot\_wider, 49
- primary\_input\_get, 7, 19, 31, 32, 34, 39, 41, 42–44, 48, 49
- primary\_inputs, 7–10, 40
- rows\_add, 7, 19, 31, 32, 34, 39, 41, 41, 43, 44, 48, 49
- supplementary\_add, 7, 19, 31, 32, 34, 39, 41, 42, 42, 44, 48, 49
- tempdir, 25
- total\_tax\_add, 7, 19, 31, 32, 34, 39, 41–43, 44, 48, 49
- uk\_2010\_data, 17, 37, 45, 47
- uk\_2010\_results\_get, 46
- uk\_test\_results, 17, 37, 45, 46
- vector\_transpose  
(vector\_transpose\_longer), 47
- vector\_transpose\_longer, 7, 19, 31, 32, 34, 39, 41–44, 47, 49
- vector\_transpose\_wider, 7, 19, 31, 32, 34, 39, 41–44, 48, 48