Package 'fio'

August 21, 2024

Title Friendly Input-Output Analysis

Version 0.1.1

Description Simplifies the process of importing and managing input-output matrices from 'Microsoft Excel' into R, and provides a suite of functions for analysis. It leverages the 'R6' class for clean, memory-efficient object-oriented programming. Furthermore, all linear algebra computations are implemented in 'Rust' to achieve highly optimized performance.

URL https://albersonmiranda.github.io/fio/, https://github.com/albersonmiranda/fio

BugReports https://github.com/albersonmiranda/fio/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.0)

SystemRequirements Cargo (Rust's package manager), rustc >= 1.67.1, xz

LazyData true

Imports cli, clipr, emoji, fs, miniUI, readxl, rlang, shiny, Rdpack, R6

License MIT + file LICENSE

Config/rextendr/version 0.3.1.9000

Suggests knitr, rmarkdown, spelling, microbenchmark, leontief, ggplot2, writexl, callr, testthat (>= 3.0.0)

VignetteBuilder knitr

Language en-US

Config/testthat/edition 3

RdMacros Rdpack

NeedsCompilation yes

Author Alberson da Silva Miranda [aut, cre, cph] (<https://orcid.org/0000-0001-9252-4175>), Celso Bissoli Sessa [dtc] (<https://orcid.org/0000-0001-7616-0244>)

Maintainer Alberson da Silva Miranda <albersonmiranda@hotmail.com>

Repository CRAN

Date/Publication 2024-08-21 12:40:02 UTC

Contents

`br_2020` *Brazil input-output matrix, year 2020, 51 sectors*

Description

This dataset contains the Brazilian input-output matrix for the year 2020, with 51 sectors. The data is based on the Brazilian Institute of Geography and Statistics (IBGE) and the Brazilian Institute of Applied Economic Research (IPEA).

Usage

```
br_2020
```

Format

`br_2020`**:**

A R6 class containing a set of matrices:

`id` Identifier of the new instance

`intermediate_transactions` Intermediate transactions matrix.

`total_production` Total production matrix.

`final_demand` Final demand matrix.

`exports` Exports matrix.

`imports` Imports matrix.

`taxes` Taxes matrix.

`value_added` Value added matrix.

fio_addin *Conveniently import data from an Excel file*

Description

`fio_addin()` opens an RStudio gadget and addin that allows you to say where the data source is (either clipboard or Excel file) and import the data into the global environment. Appears as "Import input-output data" in the RStudio Addins menu.

Usage

```
fio_addin()
```

References

This function is based on the reprex package.

import_element *Import IOM data*

Description

Import data from a input-output matrix (IOM) from Excel format.

Usage

```
import_element(file, sheet, range, col_names = FALSE, row_names = FALSE)
```

Arguments

`file`	Path to the Excel file.
`sheet`	Name of the sheet in the Excel file.
`range`	Range of cells in the Excel file.
`col_names`	Range of cells with column names.
`row_names`	Range of cells with row names.

Value

A (`matrix`).

Examples

```
# Excel file with IOM data
path_to_xlsx <- system.file("extdata", "iom/br/2020.xlsx", package = "fio")
# Import IOM data
intermediate_transactions = import_element(
  file = path_to_xlsx,
  sheet = "MIP",
  range = "D6:BB56",
  col_names = "D4:BB4",
  row_names = "B6:B56"
)
# Show the first 6 rows and 6 columns
intermediate_transactions[1:6, 1:6]
```

`iom` *R6 class for input-output matrix*

Description

R6 class for input-output matrix.

Value

A new instance of the `iom` class.

Public fields

`id` (`character`)
: Identifier of the new instance.

`intermediate_transactions` (`matrix`)
: Intermediate transactions matrix.

`total_production` (`matrix`)
: Total production vector.

`household_consumption` (`matrix`)
: Household consumption vector.

`government_consumption` (`matrix`)
: Government consumption vector.

`exports` (`matrix`)
: Exports vector.

`final_demand_others` (`matrix`)
: Other vectors of final demand that doesn't have dedicated slots.

`final_demand_matrix` (`matrix`)
: Aggregates final demand vectors into a matrix.

`imports` (`matrix`)
: Imports vector.

`taxes` (`matrix`)
: Taxes vector.

`wages` (`matrix`)
: Wages vector.

`operating_income` (`matrix`)
: Operating income vector.

`value_added_others` (`matrix`)
: Other vectors of value-added that doesn't have dedicated slots.

`value_added_matrix` (`matrix`)
: Aggregates value-added vectors into a matrix.

`occupation` (`matrix`)
: Occupation vector.

`technical_coefficients_matrix` (`matrix`)
: Technical coefficients matrix.

`leontief_inverse_matrix` (`matrix`)
: Leontief inverse matrix.

`multiplier_output` (`data.frame`)
: Output multiplier dataframe.

`multiplier_employment` (`data.frame`)
: Employment multiplier dataframe.

`multiplier_taxes` (`data.frame`)
: Taxes multiplier dataframe.

`multiplier_wages` (`data.frame`)
: Wages multiplier dataframe.

`field_influence` (`matrix`)
: Influence field matrix.

`key_sectors` (`data.frame`)
: Key sectors dataframe.

`allocation_coefficients_matrix` (`matrix`)
: Allocation coefficients matrix.

`ghosh_inverse_matrix` (`matrix`)
: Ghosh inverse matrix.

`hypothetical_extraction` (`matrix`)
: Absolute and relative backward and forward differences in total output after a hypothetical extraction

`threads` (`integer`)
: Number of threads available for Rust to run in parallel

Methods

Public methods:

- `iom$new()`
- `iom$add()`

- iom$remove()
- iom$update_final_demand_matrix()
- iom$update_value_added_matrix()
- iom$compute_tech_coeff()
- iom$compute_leontief_inverse()
- iom$compute_multiplier_output()
- iom$compute_multiplier_employment()
- iom$compute_multiplier_wages()
- iom$compute_multiplier_taxes()
- iom$compute_field_influence()
- iom$compute_key_sectors()
- iom$compute_allocation_coeff()
- iom$compute_ghosh_inverse()
- iom$compute_hypothetical_extraction()
- iom$set_max_threads()
- iom$clone()

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
iom$new(
  id,
  intermediate_transactions,
  total_production,
  household_consumption = NULL,
  government_consumption = NULL,
  exports = NULL,
  final_demand_others = NULL,
  imports = NULL,
  taxes = NULL,
  wages = NULL,
  operating_income = NULL,
  value_added_others = NULL,
  occupation = NULL,
  threads = 0
)
```

Arguments:

`id` (`character`)
: Identifier for the input-output matrix.

`intermediate_transactions` (`matrix`)
: Intermediate transactions matrix.

`total_production` (`matrix`)
: Total production vector.

`household_consumption` (`matrix`)
: Household consumption vector.

`government_consumption` (`matrix`)
: Government consumption vector.

`exports` (`matrix`)
: Exports vector.

`final_demand_others` (`matrix`)
: Other vectors of final demand that doesn't have dedicated slots. Setting column names is advised for better readability.

`imports` (`matrix`)
: Imports vector.

`taxes` (`matrix`)
: Taxes vector.

`wages` (`matrix`)
: Wages vector.

`operating_income` (`matrix`)
: Operating income vector.

`value_added_others` (`matrix`)
: Other vectors of value-added that doesn't have dedicated slots. Setting row names is advised for better readability.

`occupation` (`matrix`)
: Occupation matrix.

`threads` (`integer`)
: Number of threads available for Rust to run in parallel.

Method `add()`**:** Adds a `matrix` to the `iom` object.

Usage:

```
iom$add(matrix_name, matrix)
```

Arguments:

`matrix_name` (`character`)
: One of household_consumption, government_consumption, exports, final_demand_others, imports, taxes, wages, operating income, value_added_others or occupation matrix to be added.

`matrix` (`matrix`)
: Matrix object to be added.

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- iom$new("mock", intermediate_transactions, total_production)
# Create a dummy matrix
exports_data <- matrix(as.numeric(1:3), 3, 1)
# Add the matrix
my_iom$add("exports", exports_data)
```

Method `remove()`**:** Removes a `matrix` from the `iom` object.

Usage:

```
iom$remove(matrix_name)
```

Arguments:

`matrix_name (character)`
: One of household_consumption, government_consumption, exports, final_demand_others, imports, taxes, wages, operating_income, value_added_others or occupation matrix to be removed.

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
 exports_data <- matrix(as.numeric(1:3), 3, 1)
# instantiate iom object
my_iom <- iom$new("mock", intermediate_transactions, total_production, exports = exports_data)
# Remove the matrix
my_iom$remove("exports")
```

Method `update_final_demand_matrix()`**:** Aggregates final demand vectors into the `final_demand_matrix` field.

Usage:

```
iom$update_final_demand_matrix()
```

Details: Some methods, as `$compute_hypothetical_extraction()`, require the final demand and value-added vectors to be aggregated into a matrix. This method does this aggregation, binding the vectors into `$final_demand_matrix`.

Returns: This functions doesn't returns a value.

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
exports_data <- matrix(c(10, 20, 30), 3, 1)
households <- matrix(as.numeric(4:6), 3, 1)
# instantiate iom object
my_iom <- iom$new(
 "mock",
 intermediate_transactions,
 total_production,
 exports = exports_data,
 household_consumption = households
)
# aggregate all final demand vectors
my_iom$update_final_demand_matrix()
# check final demand matrix
my_iom$final_demand_matrix
```

Method `update_value_added_matrix()`**:** Aggregates value-added vectors into the `value_added_matrix` field.

Usage:

```
iom$update_value_added_matrix()
```

Details: Some methods, as `$compute_hypothetical_extraction()`, require the final demand and value-added vectors to be aggregated into a matrix. This method does this aggregation, binding the vectors into `$value_added_matrix`.

Returns: This functions doesn't returns a value.

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
imports_data <- matrix(c(5, 10, 15), 1, 3)
taxes_data <- matrix(c(2, 5, 10), 1, 3)
# instantiate iom object
my_iom <- iom$new(
"mock",
intermediate_transactions,
total_production,
imports = imports_data,
taxes = taxes_data
)
# aggregate all value-added vectors
my_iom$update_value_added_matrix()
# check value-added matrix
my_iom$value_added_matrix
```

Method `compute_tech_coeff()`**:** Computes the technical coefficients matrix and populate the `technical_coefficients_matrix` field with the resulting (`matrix`).

Usage:

```
iom$compute_tech_coeff()
```

Details: It computes the technical coefficients matrix, a nxn matrix known as `A` matrix which is the column-wise ratio of intermediate transactions to total production (Leontief 1983).

Returns: Self (invisibly).

Examples:

```
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- iom$new("test", intermediate_transactions, total_production)
# Calculate the technical coefficients
my_iom$compute_tech_coeff()
# show the technical coefficients
my_iom$technical_coefficients_matrix
```

Method `compute_leontief_inverse()`**:** Computes the Leontief inverse matrix and populate the `leontief_inverse_matrix` field with the resulting (`matrix`).

Usage:

```
iom$compute_leontief_inverse()
```

Details: It computes the Leontief inverse matrix (Leontief 1983), which is the inverse of the Leontief matrix, defined as:

$$L = I - A$$

where I is the identity matrix and A is the technical coefficients matrix. The Leontief inverse matrix is calculated by solving the following equation:

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

Since the Leontief matrix is a square matrix and the subtraction of the technical coefficients matrix from the identity matrix guarantees that the Leontief matrix is invertible, underlined Rust function uses LU decomposition to solve the equation.

Returns: Self (invisibly).

Examples:

```
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# show the Leontief inverse
my_iom$leontief_inverse_matrix
```

Method `compute_multiplier_output()`**:** Computes the output multiplier and populate the `multiplier_output` field with the resulting (`data.frame`).

Usage:

```
iom$compute_multiplier_output()
```

Details: An output multiplier for sector *j* is defined as the total value of production in all sectors of the economy that is necessary in order to satisfy a monetary unit (e.g., a dollar) worth of final demand for sector *j*'s output (Miller and Blair 2009).
This method computes the simple output multiplier, defined as the column sums of the Leontief inverse matrix, the direct and indirect output multipliers, which are the column sums of the technical coefficients matrix and the difference between total and direct output multipliers, respectively (Vale and Perobelli 2020).

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
```

```
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the output multiplier
my_iom$compute_multiplier_output()
# show the output multiplier
my_iom$multiplier_output
```

Method `compute_multiplier_employment()`**:** Computes the employment multiplier and populate the `multiplier_employment` field with the resulting (`data.frame`).

Usage:

```
iom$compute_multiplier_employment()
```

Details: The employment multiplier for sector *j* relates the jobs created in each sector in response to a initial exogenous shock (Miller and Blair 2009).
Current implementation follows (Vale and Perobelli 2020).

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
jobs_data <- matrix(c(10, 12, 15), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production, occupation = jobs_data)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the employment multiplier
my_iom$compute_multiplier_employment()
# show the employment multiplier
my_iom$multiplier_employment
```

Method `compute_multiplier_wages()`**:** Computes the wages multiplier dataframe and populate the `multiplier_wages` field with the resulting (`data.frame`).

Usage:

```
iom$compute_multiplier_wages()
```

Details: The wages multiplier for sector *j* relates increases in wages for each sector in response to a initial exogenous shock (Miller and Blair 2009).
Current implementation follows (Vale and Perobelli 2020).

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
```

```
wages_data <- matrix(c(10, 12, 15), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production, wages = wages_data)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the wages multiplier
my_iom$compute_multiplier_wages()
# show the wages multiplier
my_iom$multiplier_wages
```

Method `compute_multiplier_taxes()`**:** Computes the taxes multiplier and populate the `multiplier_taxes` field with the resulting (`data.frame`).

Usage:

```
iom$compute_multiplier_taxes()
```

Details: The taxes multiplier for sector *j* relates the increases on tax revenue from each sector in response to a initial exogenous shock (Miller and Blair 2009).
Current implementation follows (Vale and Perobelli 2020).

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
tax_data <- matrix(c(10, 12, 15), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production, taxes = tax_data)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the tax multiplier
my_iom$compute_multiplier_taxes()
# show the taxes multiplier
my_iom$multiplier_taxes
```

Method `compute_field_influence()`**:** Computes the field of influence for all sectors and populate the `field_influence` field with the resulting (`matrix`).

Usage:

```
iom$compute_field_influence(epsilon)
```

Arguments:

`epsilon (numeric)`
: Epsilon value. A technical change in the input-output matrix, caused by a variation of size `epsilon` into each element of technical coefficients matrix.

Details: The field of influence shows how changes in direct coefficients are distributed throughout the entire economic system, allowing for the determination of which relationships between sectors are most important within the production process.

It determines which sectors have the greatest influence over others, specifically, which coefficients, when altered, would have the greatest impact on the system as a whole (Vale and Perobelli 2020).

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate field of influence
my_iom$compute_field_influence(epsilon = 0.01)
# show the field of influence
my_iom$field_influence
```

Method `compute_key_sectors()`**:** Computes the key sectors dataframe, based on it's power and sensitivity of dispersion, and populate the `key_sectors` field with the resulting (`data.frame`).

Usage:

```
iom$compute_key_sectors()
```

Details: Increased production from a sector *j* means that the sector *j* will need to purchase more goods from other sectors. At the same time, it means that more goods from sector *j* will be available for other sectors to purchase. Sectors that are above average in the demand sense (stronger backward linkage) have power of dispersion indices greater than 1. Sectors that are above average in the supply sense (stronger forward linkage) have sensitivity of dispersion indices greater than 1 (Miller and Blair 2009).
As both power and sensitivity of dispersion are related to average values on the economy, coefficients of variation are also calculated for both indices. The lesser the coefficient of variation, greater the number of sectors on the demand or supply structure of that sector (Vale and Perobelli 2020).

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate key sectors
my_iom$compute_key_sectors()
```

```
# show the key sectors
my_iom$key_sectors
```

Method `compute_allocation_coeff()`**:** Computes the allocation coefficients matrix and populate the `allocation_coefficients_matrix` field with the resulting (`matrix`).

Usage:

```
iom$compute_allocation_coeff()
```

Details: It computes the allocation coefficients matrix, a nxn matrix known as `B` matrix which is the row-wise ratio of intermediate transactions to total production (Miller and Blair 2009).

Returns: Self (invisibly).

Examples:

```
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# Calculate the allocation coefficients
my_iom$compute_allocation_coeff()
# show the allocation coefficients
my_iom$allocation_coefficients_matrix
```

Method `compute_ghosh_inverse()`**:** Computes the Ghosh inverse matrix and populate the `ghosh_inverse_matrix` field with the resulting (`matrix`).

Usage:

```
iom$compute_ghosh_inverse()
```

Details: It computes the Ghosh inverse matrix (Miller and Blair 2009), defined as:

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

where I is the identity matrix and B is the allocation coefficients matrix.

Returns: Self (invisibly).

Examples:

```
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# Calculate the allocation coefficients
my_iom$compute_allocation_coeff()
# Calculate the Ghosh inverse
my_iom$compute_ghosh_inverse()
# show the Ghosh inverse
my_iom$ghosh_inverse_matrix
```

Method `compute_hypothetical_extraction()`**:** Computes total impact after extracting a each sector and populate the `hypothetical_extraction` field with the resulting (`data.frame`).

Usage:

```
iom$compute_hypothetical_extraction()
```

Details: Computes impact on demand and supply structures after extracting each sector (Miller and Blair 2009).
The total impact is calculated by the sum of the direct and indirect impacts.

Returns: Self (invisibly).

Examples:

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
exports_data <- matrix(c(5, 10, 15), 3, 1)
holsehold_consumption_data <- matrix(c(20, 25, 30), 3, 1)
operating_income_data <- matrix(c(2, 5, 10), 1, 3)
taxes_data <- matrix(c(1, 2, 3), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new(
 "test",
 intermediate_transactions,
 total_production,
 exports = exports_data,
 household_consumption = holsehold_consumption_data,
 operating_income = operating_income_data,
 taxes = taxes_data
)
# update value-added matrix
my_iom$update_value_added_matrix()
# update final demand matrix
my_iom$update_final_demand_matrix()
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate allocation coefficients
my_iom$compute_allocation_coeff()
# calculate Ghosh inverse
my_iom$compute_ghosh_inverse()
# calculate hypothetical extraction
my_iom$compute_hypothetical_extraction()
# show results
my_iom$hypothetical_extraction
```

Method `set_max_threads()`**:** Sets max number of threads used by fio and populate the `threads` field with the resulting `(integer)`.

Usage:

```
iom$set_max_threads(max_threads)
```

Arguments:

`max_threads (integer)`
: Number of threads enabled for parallel computing. Defaults to 0, meaning all threads available.

Details: Calling this function sets a global limit of threads to Rayon crate, affecting all computations that runs in parallel by default.
Default behavior of Rayon is to use all available threads (including logical). Setting to 1 will result in single threaded (sequential) computations.
Initialization of the global thread pool happens exactly once. Once started, the configuration cannot be changed in the current session. If `$set_max_threads()` is called again in the same session, it'll result in an error.
Methods that deals with linear algebra computations, like `$compute_leontief_inverse()` and `$compute_ghosh_inverse()`, will try to use all available threads by default, so they also initializes global thread pool. In order to choose a maximum number of threads other than default, `$set_max_threads()` must be called before any computation, preferably right after `iom$new()`.

Returns: This function does not return a value.

Examples:

```
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# to run single threaded (sequential)
my_iom$set_max_threads(1L)
my_iom$threads
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
iom$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

There are no references for Rd macro `\insertAllCites` on this help page.

Examples

```
# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
exports <- matrix(c(10, 20, 30), 3, 1)
households <- matrix(as.numeric(4:6), 3, 1)
imports <- matrix(c(5, 10, 15), 1, 3)
jobs <- matrix(c(10, 12, 15), 1, 3)
taxes <- matrix(c(2, 5, 10), 1, 3)
wages <- matrix(c(11, 12, 13), 1, 3)

# a new iom instance can be created by passing just intermediate transactions and total production
my_iom <- iom$new(
 "example_1",
 intermediate_transactions,
 total_production
)

# or by passing optional arguments
my_iom <- iom$new(
"example_2",
intermediate_transactions,
total_production,
household_consumption = households,
exports = exports,
imports = imports,
taxes = taxes,
wages = wages,
occupation = jobs
)


## ------------------------------------------------
## Method `iom$add`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- iom$new("mock", intermediate_transactions, total_production)
# Create a dummy matrix
exports_data <- matrix(as.numeric(1:3), 3, 1)
# Add the matrix
my_iom$add("exports", exports_data)

## ------------------------------------------------
## Method `iom$remove`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
```

```
total_production <- matrix(c(100, 200, 300), 1, 3)
 exports_data <- matrix(as.numeric(1:3), 3, 1)
# instantiate iom object
my_iom <- iom$new("mock", intermediate_transactions, total_production, exports = exports_data)
# Remove the matrix
my_iom$remove("exports")

## ------------------------------------------------
## Method `iom$update_final_demand_matrix`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
exports_data <- matrix(c(10, 20, 30), 3, 1)
households <- matrix(as.numeric(4:6), 3, 1)
# instantiate iom object
my_iom <- iom$new(
 "mock",
 intermediate_transactions,
 total_production,
 exports = exports_data,
 household_consumption = households
)
# aggregate all final demand vectors
my_iom$update_final_demand_matrix()
# check final demand matrix
my_iom$final_demand_matrix

## ------------------------------------------------
## Method `iom$update_value_added_matrix`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
imports_data <- matrix(c(5, 10, 15), 1, 3)
taxes_data <- matrix(c(2, 5, 10), 1, 3)
# instantiate iom object
my_iom <- iom$new(
"mock",
intermediate_transactions,
total_production,
imports = imports_data,
taxes = taxes_data
)
# aggregate all value-added vectors
my_iom$update_value_added_matrix()
# check value-added matrix
my_iom$value_added_matrix

## ------------------------------------------------
## Method `iom$compute_tech_coeff`
```

```
## ------------------------------------------------

intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- iom$new("test", intermediate_transactions, total_production)
# Calculate the technical coefficients
my_iom$compute_tech_coeff()
# show the technical coefficients
my_iom$technical_coefficients_matrix

## ------------------------------------------------
## Method `iom$compute_leontief_inverse`
## ------------------------------------------------

intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# show the Leontief inverse
my_iom$leontief_inverse_matrix

## ------------------------------------------------
## Method `iom$compute_multiplier_output`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the output multiplier
my_iom$compute_multiplier_output()
# show the output multiplier
my_iom$multiplier_output

## ------------------------------------------------
## Method `iom$compute_multiplier_employment`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
jobs_data <- matrix(c(10, 12, 15), 1, 3)
# instantiate iom object
```

```
my_iom <- fio::iom$new("test", intermediate_transactions, total_production, occupation = jobs_data)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the employment multiplier
my_iom$compute_multiplier_employment()
# show the employment multiplier
my_iom$multiplier_employment

## ------------------------------------------------
## Method `iom$compute_multiplier_wages`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
wages_data <- matrix(c(10, 12, 15), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production, wages = wages_data)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the wages multiplier
my_iom$compute_multiplier_wages()
# show the wages multiplier
my_iom$multiplier_wages

## ------------------------------------------------
## Method `iom$compute_multiplier_taxes`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
tax_data <- matrix(c(10, 12, 15), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production, taxes = tax_data)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate the tax multiplier
my_iom$compute_multiplier_taxes()
# show the taxes multiplier
my_iom$multiplier_taxes

## ------------------------------------------------
## Method `iom$compute_field_influence`
## ------------------------------------------------

# data
```

```
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate field of influence
my_iom$compute_field_influence(epsilon = 0.01)
# show the field of influence
my_iom$field_influence

## ------------------------------------------------
## Method `iom$compute_key_sectors`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate key sectors
my_iom$compute_key_sectors()
# show the key sectors
my_iom$key_sectors

## ------------------------------------------------
## Method `iom$compute_allocation_coeff`
## ------------------------------------------------

intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# Calculate the allocation coefficients
my_iom$compute_allocation_coeff()
# show the allocation coefficients
my_iom$allocation_coefficients_matrix

## ------------------------------------------------
## Method `iom$compute_ghosh_inverse`
## ------------------------------------------------

intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# Calculate the allocation coefficients
```

```
my_iom$compute_allocation_coeff()
# Calculate the Ghosh inverse
my_iom$compute_ghosh_inverse()
# show the Ghosh inverse
my_iom$ghosh_inverse_matrix

## ------------------------------------------------
## Method `iom$compute_hypothetical_extraction`
## ------------------------------------------------

# data
intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
exports_data <- matrix(c(5, 10, 15), 3, 1)
holsehold_consumption_data <- matrix(c(20, 25, 30), 3, 1)
operating_income_data <- matrix(c(2, 5, 10), 1, 3)
taxes_data <- matrix(c(1, 2, 3), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new(
 "test",
 intermediate_transactions,
 total_production,
 exports = exports_data,
 household_consumption = holsehold_consumption_data,
 operating_income = operating_income_data,
 taxes = taxes_data
)
# update value-added matrix
my_iom$update_value_added_matrix()
# update final demand matrix
my_iom$update_final_demand_matrix()
# calculate the technical coefficients
my_iom$compute_tech_coeff()
# calculate the Leontief inverse
my_iom$compute_leontief_inverse()
# calculate allocation coefficients
my_iom$compute_allocation_coeff()
# calculate Ghosh inverse
my_iom$compute_ghosh_inverse()
# calculate hypothetical extraction
my_iom$compute_hypothetical_extraction()
# show results
my_iom$hypothetical_extraction

## ------------------------------------------------
## Method `iom$set_max_threads`
## ------------------------------------------------

intermediate_transactions <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), 3, 3)
total_production <- matrix(c(100, 200, 300), 1, 3)
# instantiate iom object
my_iom <- fio::iom$new("test", intermediate_transactions, total_production)
# to run single threaded (sequential)
```

```
my_iom$set_max_threads(1L)
my_iom$threads
```

Index