# Package 'exvatools'

June 16, 2023

**Type** Package

**Title** Value Added in Exports and Other Input-Output Table Analysis Tools

**Version** 0.4.0

**Description** Analysis of trade in value added with international input-output tables. Includes commands for easy data extraction, matrix manipulation, decomposition of value added in gross exports and calculation of value added indicators, with full geographical and sector customization. Decomposition methods include Borin and Mancini (2023) <doi:10.1080/09535314.2022.2153221>, Miroudot and Ye (2021) <doi:10.1080/09535314.2020.1730308>, Wang et al. (2013) <https://econpapers.repec.org/paper/nbrnberwo/19677.htm> and Koopman et al. (2014) <doi:10.1257/aer.104.2.459>.

**License** GPL-3

**Depends** R (>= 3.4.0)

**Imports** cli, data.table, methods, openxlsx, reshape2, utils

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Enrique Feas [aut, cre] (<https://orcid.org/0000-0002-9431-6051>)

**Maintainer** Enrique Feas <efeas@runbox.com>

**Repository** CRAN

**Date/Publication** 2023-06-15 22:30:09 UTC

# R topics documented:

## bkd *Get block diagonal matrix*

### Description

Produces a block diagonal matrix version of block matrix, i.e., a matrix in which the diagonal blocks are non-zero and the off-diagonal blocks are zero

### Usage

```
bkd(df)
```

### Arguments

| | |
|---|---|
| df | A block matrix with named rows and columns. Names of countries and sectors are automatically identified. |

### Value

Block diagonal version of the original matrix.

### Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
# Block diagonal version of Y (coincides with Yd)
bkd(wio$Y)
```

## bkdiag *Diagonalize blocks of a block matrix*

### Description

Diagonalize each block of a block matrix, so sectors of origin become also sectors of destination. Blocks of dimension NxN will remain NxN, but diagonalized, and blocks of dimensions Nx1 will be expanded to NxN and then diagonalized.

### Usage

```
bkdiag(df)
```

### Arguments

| | |
|---|---|
| df | A block matrix with named rows and columns. |

### Value

Matrix df with blocks of dimension NxN diagonalized.

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
# Normal version of matrix Y
wio$Y
# Diagonal version (show first columns only)
bkdiag(wio$Y)[, 1:6]
```

---

bkoffd *Get block off-diagonal matrix*

---

## Description

Produces a block off-diagonal matrix version of block matrix, i.e., a matrix in which the diagonal blocks are zero and the off-diagonal blocks are non-zero.

## Usage

```
bkoffd(df)
```

## Arguments

| | |
|---|---|
| df | A block matrix with named rows and columns. Names of countries and sectors are automatically identified. |

## Value

Block off-diagonal version of the original matrix.

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
# Block off-diagonal version of Y (coincides with Ym)
bkoffd(wio$Y)
```

---

bkt *Block transpose matrix*

---

## Description

Transpose a matrix by blocks, so `block(s,r)` becomes `block(r,s)`, but elements within each block are not transposed.

## Usage

```
bkt(df)
```

### Arguments

| | |
|---|---|
| df | A block matrix with named rows and columns. Names of countries and sectors are automatically identified. |

### Details

bkt() takes a matrix of c1 x c2 blocks where each block has a dimension s1 x s2 and transposes its blocks. Block B21 becomes B12, B31 becomes B13, etc., but blocks are not altered internally. For instance, a matrix with rows 5 exporting countries of 4 sectors each and columns with 3 importing countries with 2 aggregated sectors, i.e., a (5 x 4) x (3 x 2), matrix will become a (3 x 4) x (5 x 2) matrix. The rows will now show the importing countries and the sectors they import from, and the columns will show the the exporting countries and the sectors they export from.

### Value

Block transposed version of df.

### See Also

bktt().

### Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
# Matrix Ym (exports of final products)
wio$Ym
# Block transposed version of Ym (imports of final products)
bkt(wio$Ym)
```

---

## bktt *Block transpose matrix with transposed blocks*

---

### Description

Block transpose matrix and then transpose each block. block(s,r) is transformed into block(r,s) and then internally transposed. This is not equivalent to directly transpose the matrix.

### Usage

```
bktt(df)
```

### Arguments

| | |
|---|---|
| df | A square block matrix with named rows and columns. Names of countries and sectors are automatically identified. Unlike bkt(), bktt() can only be used with square block matrices with NxN blocks (with row and column names in the form AUS_01T02, AUS_05, etc.) |

## Value

Block transposed version of `df` with elements transposed.

## See Also

`bkt()`.

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
# Block-transpose Z and transpose blocks (show first elements only)
bktt(wio$Z)[1:6, 1:6]
# Note that directly transposing Z produces a different result:
t(wio$Z)[1:6, 1:6]
```

---

`csums` *Sum matrix columns and assign name to resulting row*

---

## Description

Improved version of `colSums()` for matrix output. The sum of columns is kept as a row vector with column names and the resulting row can be named in the same command.

## Usage

```
csums(df, row_name = NULL)
```

## Arguments

| | |
|---|---|
| `df` | A matrix with named rows and columns. |
| `row_name` | String, name to assign to resulting row. |

## Value

A row matrix (with rows and column names)

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
csums(wio$Y, "TOTAL_Y")
```

## diagcs *Diagonalize the sums of columns of a matrix*

### Description

Makes a diagonal matrix with the sums of columns of a matrix.

Diagonalizes the sums of each column of a matrix.

### Usage

```
diagcs(df)

diagcs(df)
```

### Arguments

| | |
|---|---|
| df | A matrix with named rows and columns. |

### Value

A diagonal matrix with the sums of columns in the diagonal.

A diagonal matrix with the sums of columns in the diagonal.

### Examples

```
wio <- make_wio("wiodtest")
diagcs(wio$W %*% wio$Bd)
wio <- make_wio("wiodtest")
diagcs(wio$W %*% wio$Bd)
```

## dmult *Multiply a diagonal matrix by another matrix*

### Description

Fast multiplication of a diagonal matrix by another matrix, taking taking advantage of the properties of diagonal matrices.

### Usage

```
dmult(matrix1, matrix2)
```

### Arguments

| | |
|---|---|
| matrix1 | A diagonal matrix. |
| matrix2 | An ordinary matrix. |

## Details

`dmult()` will turn `matrix1` into a vector and multiply it horizontally by every rows in `matrix2`. This saves precious computing time.\ The number of rows and columns of the diagonal `matrix1` must be equal to the number of rows of `matrix1`.

## Value

Product of matrix1 and matrix2.

## See Also

`multd()`.

## Examples

```
wio <- make_wio("wiodtest")
dmult(wio$W, wio$Bd)
```

---

# get_data *Get data from different exvatools objects*

---

## Description

Extracts exporting country and sector and destination data from a specific variable in an `exvatools` object.

## Usage

```
get_data(
  exvatools_object,
  var,
  exporter,
  sector = "TOTAL",
  importer = "WLD",
  demand_comp = "TOTAL",
  custom = FALSE
)
```

## Arguments

| | |
|---|---|
| `exvatools_object` | An `exvatools` object (`wio`, `exvadec` or `exvadir`). If it is an ICIO `wio`, it will be previously melded (i.e., China and Mexico will be grouped). |
| `var` | String for the selected variable included in the `exvatools` object: `"VA"`, `"X"`, `"EXGR"`, `"VAX"`, `"DC"`, `"DVA"`, etc. |

exporter — String vector with codes of the exporting countries.\ If the exvadec object includes only one country or country group, exporter is not required (data can only be extracted for that country).\ If exporter is not specified and it is an exvadir object, the exporter will be considered the world ("WLD"), as by definition exporters in exvadir objects are the countries of origin of value added. \ To include a vector with several exporters (e.g., c("ESP", "FRA")) the exvadec object must have been created with the option exporter = "all" in the command make_exvadec(). get_data() will then produce matrices horizontally bound.

sector — A character vector with sector codes, e.g. TOTAL, AGF, MANUF, c("TOTAL", "AGF", "MANUF", "SERVS"). Available codes can be checked with info_sec().

importer — String vector with importing country or country group codes, e.g. "WLD", "ESP", "EU27", c("WLD", "EU27", "NONEU27"). Available codes can be checked with info_geo().\ Please note that country groups will not show the strict values of "DVA", "VAX" etc. but an average value of the countries included in that group. To obtain the specific "DVA", "VAX", etc. for a group, an exvadec object must be specifically created for that country group.\ Of course, variables that do not require to exclude double-counting, like "EXGR", "DC" or"FC" will be the same in both cases, so no specific exvadec object will be required.

demand_comp — A character vector of demand components, e.g., "HFCE", c("HFCE", "GCFC"). Only valid for wio objects.

custom — Boolean specifying whether custom-made groups of countries or sectors are present in the environment to be used. For instance, a custom HITECH custom variable including high-tech sectors or a LDC variable with list of least-developed countries. Note that custom variables should be referred to as strings in get_data(), i.e. as "HITECH" and "LDC".

## Value

A two-dimensional matrix with sector and geographical data of a variable.

## Examples

```
wio <- make_wio("wiodtest")
get_data(wio, "EXGR", exp = "ESP", sec = "MANUF")
get_data(wio, "EXGR", exp = "ESP", sec = c("TOTAL", "MANUF", "SRVWC"),
         imp = c("USA", "FRA"))
```

---

# get_exvadec_bkdown *Get a summary decomposition of value added in exports*

---

## Description

Detail from an exvadec decomposition of a country by sector and by destination

### Usage

```
get_exvadec_bkdown(
  exvadec_object,
  exporter = "WLD",
  sector = "TOTAL",
  importer = "WLD"
)
```

### Arguments

| | |
|---|---|
| exvadec_object | An exvadec object created by make_exvadec(). |
| exporter | A character string with the code for the exporting country |
| sector | Character code of sector |
| importer | Character code of importer |

### Value

Print result to console

### Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
exvadec <- make_exvadec(wio, quiet = TRUE)
get_exvadec_bkdown(exvadec, "ESP", "MANUF")
```

---

## get_geo_codes *Get the ISO3 codes of standard country groups*

---

### Description

Gets the ISO3 codes of standard country groups available for the different input-output tables. The resulting format can be used to extract elements of a matrix using grep.

### Usage

```
get_geo_codes(geo_id, wiotype = "icio2021", icio_extend = FALSE)
```

### Arguments

| | |
|---|---|
| geo_id | String, country group id. Available geo_ids for a specific input-output table can be obtained with the command info_geo(). |
| wiotype | String, type of input-output table. |
| icio_extend | Boolean. If TRUE and the input-output table is of type icio, codes will also include the extended elements for China (CN1, CN2) and Mexico (MX1, MX2). |

### Value

Codes of country/countries ready to grep, e.g. AUS|ARG|BEL

### Examples

```
# Get the codes of EU27 countries
get_geo_codes("EU27", "icio2021")
# Gets the codes for NAFTA and extends MEX to MX1|MX2
get_geo_codes("NAFTA", "icio2021", icio_extend = TRUE)
```

---

## get_sec_codes *Get the ISO3 codes of standard sector groups*

---

### Description

Gets the ISO3 codes of standard sector groups available for the different input-output tables. The resulting format can be used to extract elements of a matrix using grep.

### Usage

```
get_sec_codes(sector_id, wiotype = "icio2021", remove_letter = FALSE)
```

### Arguments

| | |
|---|---|
| sector_id | String, sector or sector group code. Available sector_ids can be obtained with the command info_sec(). |
| wiotype | String, type of input-output database. |
| remove_letter | Boolean. If TRUE, the initial letter from the sector code will be removed: D20 or C20 will become _20. This is needed to grep rows and columns, as country-sector naming follows the pattern AUS_01T02, i.e., without the initial letter D or C. |

### Value

Codes of sector ready to grep, e.g. _01|_02|_03.

### Examples

```
# Get sector codes for manufactures in the icio2021 database.
get_sec_codes("MANUF", "icio2021")
# Get sector codes for services (including construction)
get_sec_codes("SRVWC", "icio2021")
# Get sector codes for manufacturing, removing the first letter so
# the result can be used with `grep` to select specific sectors from
# a matrix
get_sec_codes("MANUF", "icio2021", remove_letter = TRUE)
```

get_va_exgr *Detailed origin and destination of value added in gross exports*

## Description

Origin of value added in gross exports. It combines a `make_exvadir()` command and a `get_data()` command to obtain a result equivalent to the OECD's Origin of Value added in Gross Exports `EXGR_BSCI`, but with much more flexible geographical and sector options.

## Usage

```
get_va_exgr(
  wio_object,
  va_type = "FC",
  geo_orig = "WLD",
  sec_orig = "TOTAL",
  geo_export,
  sec_export = "TOTAL",
  as_numeric = TRUE
)
```

## Arguments

| | |
|---|---|
| `wio_object` | An object of class `wio`. |
| `va_type` | Character string specifying the output as domestic content (`"DC"`), foreign content (`"FC"`) or total content (`"TC"`) from the perspective of the exporter. As origin of value added is specified, this is normally redundant, but in the case of exporter `"WLD"`, the domestic and foreign content is considered as the sum of domestic/foreign contents of all individual countries. For groups (such as `"EU27"`) domestic/foreign means value added from within/outside the group. |
| `geo_orig` | Character string with code of country or country group of origin of value added |
| `sec_orig` | Character string with code of sector or sector group of origin of value added. Combinations (with `"\|"`) and exceptions (with `"x"`) are allowed. |
| `geo_export` | Character string with code of exporting country or country group. |
| `sec_export` | Character string with code of exporting sector or sector group. Combinations (with `"\|"`) and exceptions (with `"x"`) are allowed. |
| `as_numeric` | Boolean specifying whether to return a numeric value or matrix (`TRUE`, default) or a data frame (default for `get_data()`). |

## Value

A matrix, vector or data frame with export value added data.

## Examples

```
wio <- make_wio("iciotest")
# Exports of manufactures of Spain using foreign VA from France
get_va_exgr(wio, "FC", "FRA", "TOTAL", "ESP", "MANUF")
```

---

`get_va_exgry` *Detailed origin and final absorption of value added in gross exports*

---

## Description

Get exports in terms of final absorption by origin of value added and final destination. It combines a `make_exvadir()` command and a `get_data()` command to obtain a result equivalent to that of the OECD's Gross Exports by Origin of Value Added and Final destination (`FD_EXGR_VA`, `FD_EXGRFNL_VA` and `FD_EXGRINT_VA`), but with much more flexible geographical and sector options.

## Usage

```
get_va_exgry(
  wio_object,
  va_type = "TC",
  flow_type = "EXGRY",
  geo_orig = "WLD",
  geo_export,
  sec_export = "TOTAL",
  geo_fd = "WLD",
  as_numeric = TRUE
)
```

## Arguments

| | |
|---|---|
| `wio_object` | An object of class `wio`. |
| `va_type` | String for domestic content (`"DC"`), foreign content (`"FC"`) or total content (`"TC"`) from the perspective of the exporter. As origin of value added is specified, this is normally redundant, but in the case of exporter `"WLD"`, the domestic and foreign content is considered as the sum of domestic/foreign contents of all individual countries. For groups (such as `"EU27"`) domestic/foreign means value added from within/outside the group. |
| `flow_type` | String specifying the type of flow in terms of absorption. It can be total gross exports (`"EXGRY"`), exports of final products (`"EXGRY_FIN"`) or exports of intermediates (`"EXGRY_INT"`). |
| `geo_orig` | Character string with code of country or country group of origin of value added. |
| `geo_export` | Character string with code of exporting country or country group. |
| `sec_export` | Character string with code of exporting sector or sector group. Combinations (with `"|"`) and exceptions (with `"x"`) are allowed. |

| | |
|---|---|
| geo_fd | String character with code of country or country group of final destination of exports |
| as_numeric | Boolean. If TRUE (default), returns a numeric value, vector or matrix instead of a data frame (default for get_data()). |

## Value

A matrix, vector or data frame with data of exports

## Examples

```
# What part of French value added exported as US final intermediate
# manufactures ends up absorbed by Spain?
wio <- make_wio("iciotest")
get_va_exgry(wio, flow_type = "EXGRY_INT", geo_orig = "FRA",
             geo_export = "USA", sec_export = "MANUF", geo_fd = "ESP")
```

---

# get_va_fd *Value added induced by final demand*

---

## Description

Details of both geographical and sector origin of the VA incorporated in exports induced by final demand. Equivalent to the OECD's Origin of Value added in Final Demand (FDVA_BSCI), but with much more flexible geographical and sector options.

## Usage

```
get_va_fd(
  wio_object,
  va_type = "TOTAL",
  geo_orig = "WLD",
  sec_orig = "TOTAL",
  geo_fd = "WLD",
  sec_fd = "TOTAL",
  intra = FALSE
)
```

## Arguments

| | |
|---|---|
| wio_object | A wio object |
| va_type | String character with the type of VA induced (VA domestically absorbed "VAD" or exported "VAX") or the equivalent inducing VA (domestic final demand "DFD" or foreign final demand "FFD"). That is, "VAD" and "DFD" will produce the same result, and so will "VAX" and "FFD". Default is both, i.e. "TOTAL" VA o total demand. |

| | |
|---|---|
| geo_orig | String character with code of country or country group generating value added, i.e., exporter. Default is "all") |
| sec_orig | String character with code of sector or sector group generating value added. Default: "all") |
| geo_fd | String character with code of country (or country group) of final demand (inducing the generation of VA) |
| sec_fd | String character with code of sector (or sector group) of final demand (inducing the generation of VA) |
| intra | Boolean for inclusion of intra-regional exports (default: FALSE) |

## Value

Matrix with source and destination of value added.

## Examples

```
wio <- make_wio("iciotest")
# Get USA's total VA in services induced by China's manufacturing
get_va_fd(wio, geo_orig = "USA", sec_orig = "SRVWC",
         geo_fd = "CHN", sec_fd = "MANUF")
# Get world VA exported (VAX), i.e., world VA induced by the rest of
# the world not domestically absorbed
get_va_fd(wio, "VAX", "WLD", "TOTAL", "WLD", "TOTAL")
```

---

# get_xmatrix *Get extraction matrix.*

---

## Description

Creates a global extraction matrix Anots of an exporter and its inverse Bnots.

## Usage

```
get_xmatrix(
  wio,
  exporter,
  perim = "country",
  partner = "WLD",
  sector = "TOTAL",
  inverse = TRUE
)
```

### Arguments

| | |
|---|---|
| wio | A class wio object |
| exporter | String, code of country or country group |
| perim | String: "country" for country perspective and "WLD" for world perspective. |
| partner | String: code of country or country group for bilateral perspectives (only with country). |
| sector | Character string: code of sector or sector group for sector perspectives (only with country). |
| inverse | Boolean, if TRUE returns the global inverse extraction matrix Bnots, if FALSE just the global extraction matrix Anots. |

### Value

The global (inverse) extraction matrix of the specified exporter.

---

## hmult *Hadamard product of matrices*

---

### Description

Hadamard product, i.e., element by element product of matrix df1 and matrix df2 (by blocks). Both matrices must be block matrices, and the number and dimension of blocks in matrix df1 and df2 must be compatible.

### Usage

```
hmult(df1, df2)
```

### Arguments

| | |
|---|---|
| df1 | A block matrix with named rows and columns (country/sector) |
| df2 | A block matrix with named rows and columns (country/sector) |

### Details

In a Hadamard product, matrices are multiplied block by block, i.e., block (s,r) %*% block(s,r).

### Value

Hadamard product of the two matrices.

## iciotest_data *ICIO-type input-output table example data*

### Description

An example of an ICIO-type input-output table, with rows for MEX and CHN disaggregated into MX1 and MX2 and CN1 and CN2, respectively.

### Usage

```
iciotest_data
```

### Format

A matrix of 30 by 42, composed of two sub-matrices:

- Sub-matrix Z, intermediate inputs: 30 by 30 (10 countries with 3 sectors each). 4 of those 10 are the extensions of CHN and MEX).
- Sub-matrix Yfd, final demand: 30 by 12 (10 countries with 3 sectors each in rows, 6 countries by 2 demand components each in columns).

### Source

Data were randomly generated with an uniform distribution.

## info_geo *Show available countries and country groups in a specific Input-Output table*

### Description

Show available countries and country groups in a specific Input-Output table

### Usage

```
info_geo(wiotype = "icio2021", lang = "en")
```

### Arguments

| | |
|---|---|
| wiotype | Character string specifying the world input-output database. |
| lang | Character string for the language of the descriptive text: "eng" for English (default) and "es" for Spanish. |

### Value

Prints country codes and descriptive text in the console.

## Examples

```
info_geo("icio2021")
```

---

`info_sec` *Show available sectors and sector groups included in a specific Input-Output table*

---

## Description

Show available sectors and sector groups included in a specific Input-Output table

## Usage

```
info_sec(wiotype = "icio2021", lang = "en")
```

## Arguments

| | |
|---|---|
| `wiotype` | Character string specifying the world input-output database |
| `lang` | Character string for the language of the descriptive text: `eng` for English (default) and `es` for Spanish. |

## Value

Prints ids, sector codes and descriptive text

## Examples

```
info_sec("icio2021")
```

---

`make_custom_wio` *Make standard world input-output matrices from custom data*

---

## Description

Creates a list object of class `wio` containing the typical international input-output matrices in a standardized format, as well as a list of code names (countries, sectors and demand components) and a list of dimensions (number of countries, sectors and demand components), using custom data.

## Usage

```
make_custom_wio(
  df,
  g_names,
  n_names = NULL,
  fd_names = NULL,
  year = NULL,
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| df | A data frame or matrix containing data for intermediate inputs and final demand. |
| g_names | A string vector with names of countries. |
| n_names | A string vector with names of sectors. If missing, sectors will be S01, S02, etc. If just one sector, it will be named TOTAL. |
| fd_names | A string vector with names of final demand components. If missing, demand components will be FD1, FD2, etc. If just one, it will be named FD. |
| year | Integer. If missing, the current year will be used. |
| quiet | Boolean, if TRUE, the function will produce a silent output. |

### Details

make_custom_wio() creates a wio from custom input-output data provided as a single matrix of dimension GxN x GxFD, i.e., the matrix Z of intermediate inputs (dimension GxN x GxN) bound with the matrix Yfd of final demand (dimension GxN x GxFD). The matrices of total output X and value added VA will be automatically generated, so should not be included. Data must be exclusively numeric.

A string vector with the names of countries is required. Number of countries will be calculated from this vector. Names for sectors and final demand components can be provided or will otherwise be automatically generated. All names must be composed of alphabetic characters (no special characters are allowed).

### Value

A wioobject of wiotype = "custom".

### Examples

```
df <- as.data.frame(matrix(c(19:36), nrow = 3))
wio <- make_custom_wio(df, g_names = c("C01", "C02", "C03"))
```

---

## make_exvadec *Decomposition of value added in exports using different methodologies*

---

### Description

Calculates the decomposition of value added in exports of a country or a group of countries according to different methodologies.

## Usage

```
make_exvadec(
  wio_object,
  exporter = "all",
  method = "bm_src",
  output = "standard",
  quiet = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| wio_object | An object of class wio (standardized world input-output table) obtained using make_wio(). |
| exporter | String with a country or a country group code (e.g., "USA", code"NAFTA", etc.). The default "all" produces the decomposition of value added in exports for all available individual countries. |
| method | A string specifying the export VA decomposition method:<br>• "bm_src": Borin and Mancini, source-based (2023) (default).<br>• "bm_snk": Borin and Mancini, sink-based (2019).<br>• "wwz": Wang et al. (2013).<br>• "kww": Koopman et al. (2014).<br>• "kww": Miroudot and Ye (2021)<br>• "oecd": OECD (not properly a decomposition). |
| output | Type of matrices in output:<br>• "standard" (default): Shows the domestic content (DC), domestic value added (DVA), the domestic double counting (DDC), the foreign content (FC), the foreign value added (FVA) and the foreign double counting (DDC). The value added exported (VAX) is also produced in most cases, and additional indicators in some cases.<br>• "terms": Shows the basic decomposition terms, whose sum gives the value of gross exports. The number and specification of terms follows the standard in the economic literature: 12 in the Borin and Mancini (2019) decompositions (source and sink), 16 in the Wang et al. (2013) decomposition, 9 in the Koopman et al. (2014) decomposition and just 4 in the Miroudot and Ye (2021) decomposition (as the latter does not expand value added in terms of final absorption). \ Additional outputs:<br>• For the "bm_src" (Borin and Mancini, 2023, source-based) method there is an additional "basic" output, without GVC indicators. This output will replace "standard" if targeted perspectives (sector, bilateral or sector-bilateral) are selected.<br>• For the Miroudot and Ye (2021) method there is an additional "terms2" output, when world perspective is selected.<br>• For the "wwz" (Wang et al. ,2013) decomposition there is an additional "terms2" option with an alternative arrangement of the 16 terms. |

- For the "oecd" decomposition there is an additional "tiva" output with several indicators of the OECD TiVA database.

quiet — Boolean, if TRUE, suppresses all status messages. Default is FALSE, i.e., messages are shown.

... — Additional parameters for targeted value added perspectives. These are only available for the "bm_src" (Borin and Mancini, 2023, source-based) and the "my" (Miroudot and Ye, 2021) decomposition. methods. Specific perimeters can be:

- partner String, for bilateral perspective. Default is "WLD", but any country or country group code (e.g. "USA" or "EU27") can be specified. In that case, all flows that cross the bilateral geographic perimeter more than once will be considered as double counting.
- sector String, for sector perspective. Default is "TOTAL", but any sector or sector group code (e.g. "MANUF") can be specified. In that case, all flows that cross the sector perimeter more than once will be considered as double counting. \ The bilateral and sector perspectives can be combined in a bilateral-sector perspective.
- perim Boolean (only for "my", and incompatible with sector or partner specifications). String, for general perimeter of value added. If perim = "WLD" (world) is specified (default is exporting country), then all flows that cross the border of *any* country more than once will be considered as double counting (unlike in the country perspective, where flows are considered as double counting only when they *exit* the border of *the exporting country* more than once). Please note that, when using the world perspective (perim = "WLD") and the terms output (output = "terms", the foreign double counting will be automatically divided into two elements ("terms2").

## Value

A list object of class exvadec with several matrices plus metadata.

## References


- Borin, A., & Mancini, M. (2023). Measuring What Matters in Value-Added Trade. Economic Systems Research, 1-25.
- Koopman, R., Wang, Z., & Wei, S.-J. (2014). Tracing Value-Added and Double Counting in Gross Exports. American Economic Review, 104(2), 459–494.
- Miroudot, S., & Ye, M. (2021). Decomposing Value Added in Gross Exports. Economic Systems Research, 33(1), 67–87.
- Wang, Z., Wei, S.-J., & Zhu, K. (2013). Quantifying International Production Sharing at the Bilateral and Sector Levels (NBER Working Paper No. 19677). National Bureau of Economic Research, Inc.


## Examples

```
# Create a test wio
wio <- make_wio("iciotest")
```

```
# Make Borin and Mancini (2023) source decomposition for Spain
exvadec <- make_exvadec(wio, exporter = "ESP", method = "bm_src")
# Make Wang et al. (2013) decomposition for all countries
# expressed in the traditional 16 terms
exvadec <- make_exvadec(wio, method = "wwz", output = "terms")
```

---

`make_exvadir` *Direction (detailed origin and destination) of value added in exports*

---

## Description

Direction of value added in exports, i.e., details of both geographical and sector origin of the VA incorporated in exports and of the final destination (in gross terms or in terms of final absorption).

## Usage

```
make_exvadir(
  wio_object,
  exporter,
  va_type = "TC",
  flow_type = "EXGR",
  orig_geo = "all",
  sec_orig = "all",
  via = "any",
  perspective = "exporter",
  intra = FALSE
)
```

## Arguments

| | |
|---|---|
| `wio_object` | A `wio` object |
| `exporter` | Country code (or country group code) of exporting country |
| `va_type` | VA total content (`"TC"`), domestic (`"DC"`) or foreign content (`"FC"`) or VA content excluding double counting (`"TVA"`, `"DVA"`, `"FVA"`) |
| `flow_type` | Gross exports (`"EXGR"`) or in terms of final demand: `"EXGRY"`, `"EXGRY_FIN"`, `"EXGRY_INT"`. |
| `orig_geo` | Geographical origin of value added (default: `"all"`) |
| `sec_orig` | Code of sector of origin of value added (default: `"all"`) |
| `via` | Code of intermediate importing country (default: `"any"`) |
| `perspective` | Sector perspective, `"origin"` or `"exporter"`. |
| `intra` | Boolean for inclusion of intra-regional exports (default: `FALSE`) |

## Value

Matrix with source and destination of value added in exports

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
# Foreign services content of value added incorporated in exports of Spain,
# by country of origin and final destination, expressed in gross terms
# (equivalent to OECD TiVA's indicator EXGR_SERV_FVA).
exvadir <- make_exvadir(wio, va = "FC", flow="EXGR", exp="ESP",
                        sec_orig="SRVWC")
summary(exvadir)
```

---

`make_wio` *Make standard world input-output matrices from source files*

---

## Description

Creates a list object of class `wio` containing the typical international input-output matrices in a standardized format, as well as a list of code names (countries, sectors and demand components) and a list of dimensions (number of countries, sectors and demand components). It can use source files from well-known databases or internal data (test data).

## Usage

```
make_wio(wiotype = "icio2021", year = NULL, src_dir = NULL, quiet = FALSE)
```

## Arguments

| | |
|---|---|
| `wiotype` | String specifying the name and edition of the input-output tables to be used: * `"icio2021"` for the 2021 edition of the OECD ICIO tables (1995-2018). Deprecated editions `"icio2018"` (1995-2011) and `"icio2016"`(2005-2015) remain available for literature replication purposes. * `"wiod2016"` for the 2016 edition of the WIOD tables (2000-2014). The deprecated edition `"wiod2013"` (1995-2011) remains available for literature replication purposes. * `"lrwiod2022"` for the 2022 edition of the long-run WIOD (1965-2000), useful for historical analysis. * `"figaro2022i` for the 2022 edition of the FIGARO EU Input-Output Tables (EU IC-SUIOTs), industry-by-industry (2010-2020), and `"figaro2022p` for the product-by-product version of the same database. * `"iciotest"` for an example of an ICIO-type international input-output table (disaggregated for `MEX` into `MX1` and `MX2` and for `CHN` into `CN1` and `CN2`) and `"wiodtest"` for an example of a WIOD-type international input-output table (not disaggregated). Data for these tables is not real, but these small input-output tables are useful for didactic purposes and to check the functionality of the program. |
| `year` | Integer specifying reference year. If `NULL` (default), the last available year of the specified database will be used. |
| `src_dir` | String specifying the source directory where the source file of the international input-output tables is saved, normally as a zip file (containing `.csv` files, `.RData` or `.xlsx` files, see Details). In order for `make_wio()` to work, these zip files should not be renamed. If `src_dir` is not specified, `make_wio()` will look in the working directory. |

quiet — Boolean, if TRUE suppress all status messages. Default is FALSE, i.e., messages are shown.

## Details

make_wio() directly unzips and processes the original source files for the different international input-output tables and returns a list with the traditional matrices, including the coefficient matrix A, the Leontief global inverse matrix B, the Leontief matrix of local inverse matrices Ld and others.

Original source files can be obtained in the OECD's ICIO web page or in the University of Groningen's WIOD web page or in the Eurostat web page

If source files are used, they must be previously downloaded and placed in an accessible folder in disk, without renaming them. The following name pattern is expected:

- ICIO_XXXX-XXXX.zip for "icio2021" (.csv files)
- ICIO2018_XXXX.zip for "icio2018" (.csv files)
- ICIO2016_XXXX.zip for "icio2016" (.csv files)
- WIOTS_in_R.zip for "wiod2016" (.RData files)
- WIOTS_in_EXCEL.zip for "wiod2013" (.xlsx files). Requires package openxlsx..
- lr_wiod_wiot_final_filled.csv for "lrwiod2022". Requires packages data.table and reshape2.
- matrix_eu-ic-io_ind-by-ind_XXXX.csv for "figaro2022i and matrix_eu-ic-io_prod-by-prod_XXXX.csv for "figaro2022p (.csv files) The input-output framework follows the traditional demand model of Leontief (1936), which makes assumptions about the stability of inputs (and therefore value-added) as a proportion of production. This allows production and value-added to be expressed as the result of variations in final demand.\ Details about the content of the world input-output object (wio) produced by make_wio() can be obtained with the command summary(wio_object).

## Value

A list object of class wio including input-output matrices, dimensions, and names.

## See Also

make_custom_wio()

## Examples

```
wio <- make_wio("iciotest")
summary(wio)
## Not run:
# The following examples require the previous download of the source
# files in the working directory or in a directory specified by `src_dir`.
wio <- make_wio("icio2021", 2018)
wio <- make_wio("wiod2016", 2014)
wio <- make_wio("wiod2021", 2018, src_dir = "C:/Users/John/R/")

## End(Not run)
```

## meld — *Meld ICIO-type matrix (consolidating China and Mexico sub-components)*

### Description

Meld `ICIO` matrix with extended countries. Melds countries `CHN` and `MEX` from their extended versions e.g., `CN1` and `CN2` are melded into `CHN`.

### Usage

```
meld(df, meld_rows = TRUE, meld_cols = TRUE)
```

### Arguments

| | |
|---|---|
| `df` | A block matrix. |
| `meld_rows` | Boolean, true to meld rows. |
| `meld_cols` | Boolean, true to meld cols. |

### Value

Melded version of `ICIO` matrix.

## multd — *Multiply a matrix by a diagonal matrix*

### Description

Fast multiplication of a matrix by a diagonal matrix, taking advantage of the properties of diagonal matrices.

### Usage

```
multd(matrix1, matrix2)
```

### Arguments

| | |
|---|---|
| `matrix1` | An ordinary matrix. |
| `matrix2` | A diagonal matrix. |

### Details

`multd()` will turn `matrix2` into a vector and multiply it horizontally by every row in `matrix1`. This saves precious computing time.\ The number of columns of `matrix1` must be equal to the rows and columns of diagonal `matrix2`.

## Value

The product of matrix1 and matrix2.

## See Also

dmult().

## Examples

```
wio <- make_wio("wiodtest")
multd(wio$B, wio$E)
```

---

# print.exvadec *Print method for* exvadec *class*

---

## Description

Print method for exvadec class

## Usage

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

## Arguments

| | |
|---|---|
| x | An object of class exvadec |
| ... | Additional arguments |

## Value

Printout to console

---

# print.exvadir *Print method for* exvadir *class*

---

## Description

Print method for exvadir class

## Usage

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

### Arguments

| | |
|---|---|
| x | An object of class exvadir |
| ... | Additional arguments |

### Value

Printout to console

---

## print.wio *Print method for* wio *class*

---

### Description

Print method for wio class

### Usage

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

### Arguments

| | |
|---|---|
| x | An object of class wio |
| ... | Additional arguments |

### Value

Printout to console

---

## rsums *Sum matrix rows and assign name to resulting column*

---

### Description

Improved version of rowSums() for matrix output. The sum of rows is kept as a column vector with rows names and the resulting column can be named in the same command.

### Usage

```
rsums(df, col_name = NULL)
```

### Arguments

| | |
|---|---|
| df | A matrix with named rows and columns. |
| col_name | String, name to assign to resulting column. |

## Value

A column matrix (with rows and column names)

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
rsums(wio$Y, "Y")
```

---

`set_zero` *Set to zero specific rows and columns of a matrix*

---

## Description

Sets to zero specific rows and columns of a matrix, to include and exclude specific geographical and sector effects.

## Usage

```
set_zero(df, orig = NULL, dest = NULL, wiotype = NULL, invert = FALSE)
```

## Arguments

| | |
|---|---|
| `df` | A matrix with named rows and columns. |
| `orig` | A vector of integers with position of rows or a list of strings with codes of country and sector of origin. |
| `dest` | A vector of integers with position of columns or a list of strings with codes of country and sector of destination. |
| `wiotype` | String, type of `wio`. Required if origin or destination is specified with lists of codes. |
| `invert` | Boolean: FALSE (default) to set to zero the specified countries and sectors, or TRUE to set to zero the non-specified countries and sectors. |

## Value

The same matrix with specific rows and columns set to zero.

## Examples

```
wio <- make_wio("wiodtest")
# Set to zero Spanish exports of intermediates of manufacturing to
# non EU27 countries (for any sector of destination) in the coefficient
# matrix A
set_zero(wio$A, list("ESP", "MANUF"), list("NONEU27", "TOTAL"), "wiodtest")
# Set to zero Spanish exports of intermediates (extraction matrix of Spain)
set_zero(wio$A, list("ESP", "TOTAL"), list("WLDxESP", "TOTAL"), "wiodtest")
```

## sumgcols — *Sum groups of columns of a matrix and name the resulting columns*

### Description

Groups a matrix by columns, by summing blocks of columns of size n each. Matrix columns should be multiple of n.

### Usage

```
sumgcols(df, n, col_names = NULL)
```

### Arguments

| | |
|---|---|
| df | A matrix with named rows and columns. |
| n | Integer, specifying the size of each group. |
| col_names | String vector of length n, with names to assign to the resulting columns. |

### Value

A matrix where each column is the sum of groups of n columns of the original matrix.

### Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
sumgcols(wio$Yfd, wio$dims$FD, wio$names$g_names)
```

## sumgrows — *Sum groups of rows of a matrix and name the resulting rows*

### Description

Groups a matrix by rows, summing blocks of rows of size n each. Matrix rows should be multiple of n.

### Usage

```
sumgrows(df, n, row_names = NULL)
```

### Arguments

| | |
|---|---|
| df | A matrix with named rows and columns. |
| n | Integer, specifying the size of each group. |
| row_names | String vector of length n, with names to assign to the resulting rows. |

## Value

A matrix where each row is the sum of groups of n rows of the original matrix.

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
sumgrows(wio$Y, wio$dims$N, wio$names$g_names)
```

---

# summary.exvadec *Summary method for* exvadec *class*

---

## Description

Summary method for exvadec class

## Usage

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

## Arguments

| | |
|---|---|
| object | An object of class exvadec. |
| ... | Additional arguments. |

## Value

Printout to console

---

# summary.exvadir *Summary method for* exvadir *class*

---

## Description

Summary method for exvadir class

## Usage

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

## Arguments

| | |
|---|---|
| object | An object of class exvadir. |
| ... | Additional arguments. |

### Value

Printout to console

---

`summary.wio` *Summary method for* `wio` *class*

---

### Description

Summary method for `wio` class

### Usage

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

### Arguments

| | |
|---|---|
| `object` | An object of class `wio` |
| `...` | Additional arguments |

### Value

Printout to console

---

`sumncol` *Sum every nth column of a matrix and name the resulting columns*

---

### Description

Groups a matrix by columns, summing every Nth column. Matrix should be multiple of N.

### Usage

```
sumncol(df, N, col_names = NULL)
```

### Arguments

| | |
|---|---|
| `df` | A matrix with named rows and columns. |
| `N` | Integer, specifying the resulting number of columns. |
| `col_names` | String vector of length N, with names to assign to the resulting columns. |

### Value

A matrix with N columns, where each columns is the sum of every Nth column of the original matrix.

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
sumncol(wio$Yfd, wio$dims$FD, paste0("WLD", "_", wio$names$fd_names))
```

---

`sumnrow` *Sum every nth row of a matrix and name the resulting rows*

---

## Description

Groups a matrix by rows, summing every `Nth` row. Matrix should be multiple of `N`.

## Usage

```
sumnrow(df, N, row_names = NULL)
```

## Arguments

| | |
|---|---|
| `df` | A matrix with named rows and columns. |
| `N` | Integer, specifying the resulting number or rows. |
| `row_names` | String vector of length `N`, with names to assign to the resulting rows. |

## Value

A matrix with `N` rows, where each row is the sum of every `Nth` row of the original matrix.

## Examples

```
wio <- make_wio("wiodtest", quiet = TRUE)
sumnrow(wio$Y, wio$dims$N, paste0("WLD", "_",
        gsub("^D", "", wio$names$n_names)))
```

---

`wiodtest_data` *WIOD-type input-output table example data*

---

## Description

An example of a WIOD-type input-output table.

## Usage

```
wiodtest_data
```

## Format

A matrix of 18 by 30, composed of two sub-matrices:

- Sub-matrix `Z`, intermediate inputs: 18 by 18 (10 countries with 3 sectors each).
- Sub-matrix `Yfd`, final demand: 18 by 12 (6 countries with 3 sectors each in rows, 6 countries by 2 demand components each in columns).

## Source

Data were randomly generated with an uniform distribution.

# Index