Package 'ioanalysis'

October 12, 2018

Title Input Output Analysis

Version 0.1

Author John Wade [aut, cre],
Ignacio Sarmiento-Barbieri [aut]

Maintainer John Wade <jjpwade2@illinois.edu>

Description Calculates fundamental IO matrices (Leontief, Wassily W. (1951) <doi:10.1038/scientificamerican1051-15>, Ghosh, A. (1958) <doi:10.2307/2550694>); within period analysis via various rankings and coefficients (Sonis and Hewings (2006) <doi:10.1080/09535319200000013>, Blair and Miller (2009) <ISBN:978-0-521-73902-3>, Antras et al (2012) <doi:10.3386/w17819>, Hummels, Ishii, and Yi (2001) <doi:10.1016/S0022-1996(00)00093-3>); across period analysis with impact analysis (Dietzenbacher, van der Linden, and Steenge (2006) <doi:10.1080/09535319300000017>, Sonis, Hewings, and Guo (2006) <doi:10.1080/09535319600000002>); and a variety of table operators.

Depends R (>= 3.4.0)

License GPL (>= 2)

URL http://www.real.illinois.edu

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-12 13:20:06 UTC

R topics documented:

`agg.region` *Aggregate Regions*

Description

`agg.sector` takes specified regions and creates a "new" joint region. This produces a new `InputOutput` object. Note the Leontief Inverse and Ghoshian Inverse are elements. All regions must have exactly the same sectors. See `locate.mismatch`.

Caution: Inverting large matrices will take a long time. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
agg.region(io, regions, newname = "newname")
```

Arguments

`io`	An `InputOutput` class object from `as.inputoutput`
`regions`	Character. Specific regions to be aggregated. Can either be a character that exactly matches the name of the region in `RS_label` or the number of the region in the order it appears in `RS_label`.
`newname`	Character. The name to give to the new aggregated region.

Details

Creates an aggregation matrix similar to that of agg.sector. See Blair and Miller 2009 for more details.

Value

A new InputOutput object is created. See as.inputoutput.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

as.inputoutput, locate.mismatch, agg.region

Examples

```
data(toy.IO)
class(toy.IO)
agg.region(toy.IO, regions = c(1,2), newname = "Magic")
```

---

agg.sector *Aggregate Sectors*

Description

agg.sector takes specified sectors and creates a "new" joint sector. This produces a new InputOutput object. Note the Leontief Inverse and Ghoshian Inverse are elements. There is deliberately no warning if the sector does not occur in all regions. See locate.mismatch.

Caution: Inverting large matrices will take a long time. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
agg.sector(io, sectors, newname = "newname")
```

Arguments

io	An InputOutput class object from as.inputoutput
sectors	Character. Specific sectors to be aggregated. Can either be a character that exactly matches the name of the sector in RS_label or the number of the sector in the order it appears in RS_label.
newname	Character. The name to give to the new aggregated sector.

Details

Creates the aggregation matrix to pre (and/or post when appropriate) to aggregate the matrices in the InputOutput object. Say you have 1 region with n sectors and you wish to aggregate sectors i and i+1. A diagonal matrix is converted into a n-1xn matrix where rows i and i+1 are additively combined together. This matrix is then used to create new aggregated tables. The "new" sector is then stored in location i. See Blair and Miller 2009 for more details.

Value

A new InputOutput object is created. See as.inputoutput.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

as.inputoutput, locate.mismatch, agg.region

Examples

```
data(toy.IO)
class(toy.IO)
newIO <- agg.sector(toy.IO, sectors = c(1,2), newname = "Party.Supplies")
```

`as.inputoutput` *Creating an Input-Output Object*

Description

Creates a list of class `InputOutput` for easier use of the other functions within `ioanalysis`. The Leontief inverse and Ghoshian inverse are calculated. A little work now to save a bunch of work in the future. For most functions in the package, this is a prerequisite. At a minimum, `Z`, `X`, and `RS_label` must be provided. See `Usage` for details.

Caution: Inverting large matrices will take a long time. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
as.inputoutput(Z, RS_label, f, f_label, E, E_label, X, V, V_label, M, M_label,
               fV, fV_label, A, B, L, G)
```

Arguments

	Let n = #sectors*#regions, l = # of labels, m = arbitrary length, r = #regions
	Required. A nxn matrix of intermediate transactions between sectors and regions. It should be in units of currency, kg, etc.
`RS_label`	Required. A nx2 "column" matrix of the regions in column 1 and sector in column 2. Other functions use those locations to correctly identify elements in the matices. If there is only one region, it still needs to be specified in column 1.
`f`	Not required. A nxm matrix of final demand. Exports SHOULD NOT be included in this matrix. Instead, put exports in the `E` matrix. However, net exports should stay.
`f_label`	Not required. A 2xn "row" matrix of the region and accounts to help identify the elements of `f`. The first row should be regions and the second should be regional account labels.
`E`	Not required. A nxr matrix of exports. Multiple columns per region is accepted.
`E_label`	Not required. A 2xn "row" matrix of the region and type of export to help identify the elements of `E`.
`X`	Required. A 1xn vector of total production for each sector across all regions. `RS_label` identifies the objects
`V`	Not required. A nxm matrix of value added. Imports SHOULD NOT be included in this matrix. Instead, put exports in the `M` matrix.
`V_label`	Not required. A mx1 "column" matrix where the only column is the type of value added. This helps identify the rows of value added. `RS_label` identifies the columns.
`M`	Not required. A mxn matrix of import. Multiple types of imports is accepted.

M_label	Not required. A mx1 "column" matrix to identify the rows of imports. RS_label identifies the columns.
fV	Not Required. The matrix of final demand's value added
fV_label	Not Required. Column matrix to identify the row elements of fV
A	Not required. A nxn matrix of technical input coefficients. If not provided, A is calculated for you.
B	Not required. A nxn matrix of technical output coefficients. If not provided, B is calculated for you.
L	Not required. The Leontief inverse. If not provided, L is calculated for you.
G	Not required. The Ghoshian inverse. If not provided, G is calculated for you.

Details

If the A matrix is not provided, it is calculated as follows:

$$a_{ij} = z_{ij}/x_j$$

If the B matrix is not provided, it is calculated as follows:

$$b_{ij} = z_{ij}/x_i$$

If the L matrix is not provided, it is calculated as follows:

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

If the G matrix is not provided, it is calculated as follows:

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

Value

as.inputouput retuns an object of class "InputOutput". Once created, it is sufficient to provide this object in all further functions in the ioanalysis package.

Z	Intermediate Transactions Matrix
RS_label	Column matrix of labels for the region and sectors used to identify elements in A, Z, X, L, ...
f	Final Demand
f_label	Row matrix of labels for accounts for f
E	Exports
E_label	Row matrix of labels for exports by sector and region for E
X	Total Production
V	Value added
V_label	Column matrix of labels for types of value added for V
M	Imports
M_label	Colum matrix of labels for type of imports for M

fV	The matrix of final demand's value added
fV_label	Column matrix to identify the row elements of fV
A	Technical Input Coefficients
B	Technical Input Coefficients
L	Leontief inverse
G	Ghoshian inverse

Note

Currently, there is no use for an intermediate transaction matrix in physical units (P). If you wish to carry this with the matrix then you can create the InputOutput object and add to it by using io$P <- P.

Author(s)

John J. P. wade

References

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. *PyIO. Input-Output Analysis with Python*. REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

Examples

```
# In toy,FullIOTable it is a full matrix of characters: a pseudo worst case scenario
data(toy.FullIOTable)
Z <- matrix(as.numeric(toy.FullIOTable[3:12, 3:12]), ncol = 10)
f <- matrix(as.numeric(toy.FullIOTable[3:12, c(13:15, 17:19)]), nrow = dim(Z)[1])
E <- matrix(as.numeric(toy.FullIOTable[3:12, c(16, 20)]), nrow = 10)
X <- matrix(as.numeric(toy.FullIOTable[3:12, 21]), ncol = 1)
V <- matrix(as.numeric(toy.FullIOTable[13:15, 3:12]), ncol = 10)
M <- as.numeric(toy.FullIOTable[16, 3:12])
fV <- matrix(as.numeric(toy.FullIOTable[15:16, c(13:15,17:19)]), nrow = 2)

# Note toy.FullIOTable is a matrix of characters: non-numeric
toy.IO <- as.inputoutput(Z = Z, RS_label = toy.FullIOTable[3:12, 1:2],
                         f = f, f_label = toy.FullIOTable[1:2, c(13:15, 17:19)],
                         E = E, E_label = toy.FullIOTable[1:2, c(16, 20)],
                         X = X,
                         V = V, V_label = toy.FullIOTable[13:15, 2],
                         M = M, M_label = toy.FullIOTable[16,2],
                         fV = fV, fV_label = toy.FullIOTable[15:16, 2])

# Notice we do not need to supply the matrix of technical coefficients (A)
```

check.RS *Do all regions have the same sectors?*

Description

Produces a logical answer to the question do all regions have the same sectors.

Usage

```
check.RS(io)
```

Arguments

io	An InputOutput class object from as.inputoutput

Details

Uses the RS_label to determine if all regions have the same sectors

Value

Produces either TRUE or FALSE

Author(s)

John J. P. Wade

See Also

locate.mismatch

Examples

```
data(toy.IO)
class(toy.IO)
check.RS(toy.IO)
```

easy.select *Region and Sector Selection Interface*

Description

This is a user interface, answering prompts to significantly simplify choosing sectors and regions in large models. You can either search through the regions and sectors using keywords, partial phrases, or partial words. There is alternatively an option to select across the comprehensive list of all regions and then sectors. Once selections are made, you can view and edit the list once selections are made. Outputs a matrix to be input into other functions to help identify desired region-sector combinations.

Usage

```
easy.select(io)
```

Arguments

io	An InputOutput object. See as.inputoutput.

Details

easy.select calls upon the RS_label object in io to sort through regions and sectors. The regions should be in the first column and sectors should be in the second.

Value

EasySelect	A numeric vector of class EasySelect that can be used to identify desired elements for future functions.

Author(s)

John J. P. Wade

See Also

as.inputoutput

export.coef *Calculates the Matrix of Trade Coefficients*

Description

Uses the matrix of technical input coefficients (A) to calculate either the matrix of import coefficients or the matrix of export coefficients. It does require that all regions have the same sectors. This can be verified using check.RS

This function is intended to be a helper function for vs

Usage

```
export.coef(io, region)
```

Arguments

io	An InputOutput class object from as.inputoutput
region	Integer. Specific region to be used. The number of the region in the order it appears in RS_label. You can only do one region at a time.

Details

Adds appropriate blocks of the matrix of technical input coefficients to calculate the matrix of import/export coefficients. If there is an export matrix or an import matrix as a part of the InputOutput object, the results in the generated matrix may be biased.

Value

Produces a nxn matrix, where n is the number of sectors.

Author(s)

John J. P. Wade

See Also

check.RS, locate.mismatch, upstream, vs

Examples

```
data(toy.IO)
class(toy.IO)
import.coef(toy.IO, 1)
```

export.total *Calculates Total Exports for InputOutput Objects*

Description

Uses values of the intermediate transaction matrix (Z) and when applicable final demand (f), and either exports (E) or imports (M) to calculate the total exports or imports for each region sector combination.

This function is intended to be a helper function for upstream and vs.

Usage

```
export.total(io)
import.total(io)
```

Arguments

io	An InputOutput class object from as.inputoutput

Value

Produces a nameless vector of total exports.

Author(s)

John J. P. Wade

See Also

export.coef

Examples

```
data(toy.IO)
class(toy.IO)
export.total(toy.IO)
import.total(toy.IO)
```

`extraction` *Hypothetical Extraction*

Description

Computes the hypothetical extraction as outlined in Dietzenbacher et al. (1993) and as outlined in Blar and Miller (2009).

Caution: Inverting large matrices will take a long time. Each individual hypothetical extraction requires the inversion of a matrix. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
extraction(io, ES = NULL, regions = 1, sectors = 1, type = "backward.total",
           aggregate = FALSE, simultaneous = FALSE, normalize = FALSE)
```

Arguments

`io`	An `InputOutput` class object from `as.inputoutput`
`ES`	An `EasySelect` class object from `easy.select` to specify which region and sector combinations to use.
`regions`	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in `RS_label` or the number of the region in the order it appears in `RS_label`.
`sectors`	Character or Integer. Specific sectors to be used. Can either be a character that exactly matches the name of the sector in `RS_label` or the number of the sector in the order it `RS_label`.
`type`	Character. Any combination of `"backward"`, `"forward"`, `"backward.total"`, and/or `"forward.total"`. See details.
`aggregate`	`TRUE` or `FALSE`. If `TRUE` produces the value of the impact over all sectors. If `FALSE` produces the impact for each sector.
`simultaneous`	`TRUE` or `FALSE`. Determines whether to extract all specified regions sequentially or simultaneously.
`normalize`	`TRUE` or `FALSE`. Whether or not to divide each linkage by total production.

Details

`type`

(1) `backward` - Calculates the impact of hypothetically extracting the jth region/sector using the formula

$$X - (I - A_c)^{-1} f$$

where A_c is the matrix of technical input coefficients with the jth column replaced by zeros

(2)`forward` - Calculates the impact of hypothetically extracting the jth region/sector using the formula

$$X - V(I - B_r)^{-1}$$

where B_r is the matrix of technical output coefficients with the jth row replaced by zeros

(3) `backward.total` - Calculates the impact of hypothetically extracting the jth region/sector using the formula

$$X - (I - A_{cr})^{-1} f$$

where A_{cr} is the matrix of technical input coefficients with the jth column and jth row replaced by zeros except for the diagonal element.

(4) `forward.total` - Calculates the impact of hypothetically extracting the jth region/sector using the formula

$$X - V(I - B_{cr})^{-1}$$

where B_{cr} is the matrix of technical output coefficients with the jth column and jth row replaced by zeros except for the diagonal element.

`aggregate`

If `TRUE` multiplies the impact vector by a vector of ones to received the summed value of the impact from hypothetical extraction.

`normalize`

If `TRUE` each component in the impact vector is divided by the total output of that sector/region combination.

Value

Produces a list over regions of a list over type of extraction. If there is only one region and one type, then a matrix is returned. For example, items can be called by using `extraction$region$type`.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Dietzenbacher Erik & van der Linden Jan A. & Steenge Alben E. (1993). The Regional Extraction Method: EC Input-Output Comparisons. Economic Systems Research. Vol. 5, Iss. 2, 1993

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

`as.inputoutput`, `easy.select`, `linkages`, `key.sector`

Examples

```
data(toy.IO)
class(toy.IO)
E1 <- extraction(toy.IO)

# Using an EasySelect object
data(toy.IO)
class(toy.IO)
E2 <- extraction(toy.IO, toy.ES)
E2$Hogwarts

# Using more options
E3 <- extraction(toy.IO, regions = c(1,2), sectors = c("Wii", "Minions"),
                 type = c("backward", "backward.total"), aggregate = TRUE)
E3$Hogwarts$backward.total

# Multiple regions and types
E4 <- extraction(toy.IO, type = c("forward","forward.total"), normalize = TRUE)
E4$Hogwarts$forward.total
```

`f.influence` *Field of Influence*

Description

Calculates the field of influence. Can handle first to nth order field of influence. Uses the method as Sonis & Hewings 1992. This is a recursive technique, so computation time depends on the size of the data and order of field of influence.

WARNING: Since this is a recursive function, each recursion saves the object for each recursion. This would make it impossible to run for large datasets, even with only a small order of field of influence. Therefore, it is WILDLY more memory efficient to call the object from the global environment (workspace). Thus, for this function you only input the name of the `InputOutput` object, not the object itself.

This function is primarily intended as a helper function for `inverse.important`

Usage

```
f.influence(ioname, i , j)
```

Arguments

`ioname`	Character. The name verbatim of your `InputOutput` from `as.inputoutput` as saved in your workspace.
`i`	Numeric. The row component(s) of the coefficient(s) of interest
`j`	Numeric. The column component(S) of the coefficient(s) of interest

Details

First Order Field of Influence - This is simply the product of the jth column of the Leontief inverse multiplied by the ith row of the Leontief inverse. In matrix notation:

$$F_1[i,j] = L_{.j}L_{i.}$$

where F denotes the field of influence, and i and j are scalars

Nth Order Field of Influence - This is a recursive function used to calculate higher order fields of influence. The order cannot exceed the size of the Intermediate Transaction Matrix (Z). I.e. if Z is 20x20, you can only calculate up to the 19th order. The formula is as follows:

$$F_k[(i_1,...,i_k),(j_1,...,j_k)] = \frac{1}{k-1}\sum_{s=1}^{k}\sum_{r=1}^{k}(-1)^{s+r+1}l_{i_s,j_r}F_{k-1}[i_{-s},j_{-r}]$$

where F is the field of influence, k is order of influence, l_ij is the ith row and jth column element of the Leontief Inverse and -s indicates the sth element has been removed.

Value

Returns a matrix of the Field of Influence

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Sonis, Michael & Hewings, Geoffry J.D. (1992), "Coefficient Chang in Input-Output Models: Theory and Applications," Economic Systems Research, 4:2, 143-158 (https://doi.org/10.1080/09535319200000013)

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

inverse.important

Examples

```
data(toy.IO)
class(toy.IO)
# First order field of influence on L[3,2]
i <- 3
j <- 2
f.influence("toy.IO", i, j)
```

```
# Second order field of influence on L[3,2], L[4,5], L[6, 3], and L[1,10]
i <- c(3, 4, 6,  1)
j <- c(2, 5, 3, 10)
f.influence("toy.IO", i, j)
```

---

ghosh.inv *Ghoshian Inverse*

Description

Computes the Ghoshian (ouput) inverse. `ghosh.inv` has inputs to invert a subset of all regions if desired. If not using an `InputOutput` object from `as.inputoutput`, the functionality is limited. See example for more details.

Caution: Inverting large matrices will take a long time. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
ghosh.inv(Z = NULL, X, B, RS_label, regions)
```

Arguments

`Z`	Either an object class of `InputOutput` calculated from `as.inputoutput` or the intermediate transaction matrix. Do NOT use matrix of technical coefficients.
`X`	Vector. Total production vector. Not required if Z is an object with `InputOutput` class.
`B`	Matrix. Matrix of technical output coefficients.
`RS_label`	Matrix. A nx2 column matrix of labels for regions and sectors. The first column must be regions and the second column must be sectors. This is used to match with the intermediate transaction matrix.
`regions`	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in `RS_label` or the number of the region in the order it appears in `RS_label`.

Details

The Ghoshian inverse is derived from the input-output table A=[a_ij] where

$$b_i j = z_i j / X_i$$

where z_ij is the input from i required in the production of j. X_i is the corresponding input in each row. The Leontief inverse is then computed as

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

Observe we result with the following system

$$X' = V'G$$

Therefore, the element g_{ij} is interpreted as the ratio of sector i's value added contributing to the total production of sector j.

Value

Returns a matrix with the Ghoshian Inverse

Author(s)

Ignacio Sarmiento-Barbieri, John J. P. Wade

References

Ghosh, A. (1958). "Input-output Approach in an Allocation System," *Econometrica*, New Series, Vol. 25, No. 97 (Feb., 1958), pp. 58-64.

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. *PyIO. Input-Output Analysis with Python*. REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

Examples

```
# Using an "InputOutput" object
data(toy.IO)
class(toy.IO)

G1 <- ghosh.inv(toy.IO, region = "Narnia")

# Otherwise
Z <- toy.IO$Z
X <- toy.IO$X
G3 <- ghosh.inv(Z, X)
```

---

inverse.important *Inverse.Important Coefficients*

Description

Calculates the inverse-important coefficients as in Blair and Miller (2009)

Usage

```
inverse.important(io, i, j, delta.aij)
```

Arguments

io	An InputOutput class object from as.inputoutput
i	Integer. The row component of the change in the matrix of technical input coefficients
j	Integer. The column component of the change in the matrix of technical input coefficients
delta.aij	Integer. By how much aij should change by

Details

The inverse-important coefficients is the change in the Leontief matrix due to a specified change in one element of the matrix of technical input coefficients (A). This uses the formula:

$$\Delta L = \frac{\Delta a_{ij}}{1 - l_{ji}\Delta a_{ij}} F_1(i,j)$$

where F_1(X,Y) is the first order field of influence.

Value

Returns the change in the Leontief matrix due the change in one element of the matrix of technical input coefficients. To find the new Leontief inverse induced by this change, use io$L + inverse.important().

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Examples

```
data(toy.IO)
class(toy.IO)
i <- 3
j <- 4
delta.aij <- 0.5
II <- inverse.important(toy.IO, i, j, delta.aij)
```

key.sector *Impact Analysis via Backward and Forward Linkages*

Description

Uses backward and forward linkages to identify key sectors in the system. Can calculate total and direct linkages. If the data is multiregional, intraregional and interregional linkages can be calculated. Can also be used on a specified subset of all regions.

Usage

```
key.sector(io, ES = NULL, crit = 1, regions = "all", sectors = "all",
           type = c("direct"), intra.inter = FALSE)
```

Arguments

io	An object of class InputOutput calculated from as.inputoutput.
ES	An object of class EasySelect from easy.select
crit	Integer. The value to compare linkages above or below to classify sectors. Default is 1.
regions	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in RS_label or the number of the region in the order it appears in RS_label.
sectors	Character or Integer. Specific sectors to be used. Can either be a character that exactly matches the name of the sector in RS_label or the number of the sector in the order it RS_label.
type	Character. Identifying the type of backward and forward linkages to be calculated. Options are "total" and "direct".
intra.inter	Logical. Only applies to multiregional systems. Determines whether or not to calculate intraregional and interregional backward and forward linkages in addition to aggregate linkages.

Details

Uses the (various) specified backward and forward linkages to calculate a key to identify dependence using the specified critical value.

I BL < crit, FL < crit - Generally independent

II BL < crit, FL > crit - Dependent on interindustry demand

III BL > crit, FL > crit - Generally dependent

IV BL > crit, FL < crit - Dependent on interindustry supply

Value

If there is only one region, key sector binds to the output from linkages to make a table. Otherwise, it produces a list of key sector codes for each country using the names of regions provided. See Examples for more details.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

linkages, as.inputoutput

Examples

```
data(toy.IO)
class(toy.IO)
key1 <- key.sector(toy.IO)
key1$Narnia

data(toy.ES)
class(toy.ES)
key2 <- key.sector(toy.IO, toy.ES)
key2

# A more detailed example
# Using critical value of 2 because this is randomly generated data and better
# illustrates functionality
key3 <- key.sector(toy.IO, intra.inter = TRUE, type = c("direct"), crit = 2)
key3

key4 <- key.sector(toy.IO, regions = c(1:2), sectors = c(1:3,5))
key4
```

---

leontief.inv *Leontief Inverse*

Description

Computes the Leontief (input) inverse. leontief.inv has inputs to invert a subset of all regions if desired. If not using an InputOutput object from as.inputoutput, the functionality is limited. See example for more details.

Note: if you have a non InputOutput object and you wish to use only a subset of all regions, you must supply the intermediate transaction matrix (Z) and total production matrix (X). Otherwise use L <- Z %*% diag(c(1/X))

Caution: Inverting large matrices will take a long time. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
leontief.inv(Z = NULL, X, A, RS_label, regions)
```

Arguments

Z	Either an object class of InputOutput calculated from as.inputoutput or the intermediate transaction matrix. Do NOT use matrix of technical coefficients.
X	vector. Total production vector. Not required if Z is an object with InputOutput class.
A	Matrix. Technical Matrix of Input Coefficients. If provided and the data is large, the computations will be noticeably sped up.
RS_label	Matrix. A nx2 column matrix of labels for regions and sectors. The first column must be regions and the second column must be sectors. This is used to match with the intermediate transaction matrix.
regions	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in RS_label or the number of the region in the order it appears in RS_label.

Details

The Leontief inverse is derived from the input-output table A=[a_ij] where

$$a_ij = z_ij/X_j$$

where z_ij is the input from i required in the production of j. X_j is the corresponding input in each column. The Leontief inverse is then computed as

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

Observe we result with the following system

$$X = Lf$$

Therefore, element l_{ij} is interpreted as the ratio of final demand for sector j contributing to the total production in sector i.

Value

Returns a matrix with the Leontief Inverse.

Author(s)

Ignacio Sarmiento-Barbieri, John J. P. Wade

References

Leontief, Wassily W. (1951). "Input-Output Economics." *Scientific American*, Vol. 185, No. 4 (October 1951), pp. 15-21.

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. *PyIO. Input-Output Analysis with Python*. REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

Examples

```
# Using an "InputOutput" object
data(toy.IO)
class(toy.IO)

L1 <- leontief.inv(toy.IO, region = "Narnia")

# Otherwise
Z <- toy.IO$Z
X <- toy.IO$X
L2 <- leontief.inv(Z, X)
```

---

linkages *Backward and Forward Linkages*

Description

Calculates backward and forward linkages with an option to normalize values. Can calculate total and direct linkages. If the data is multiregional, intraregional and interregional linkages can be calculated. Can also be used on a specified subset of all regions.

Usage

```
linkages(io, ES = NULL, regions = "all", sectors = "all", type = c("total"),
         normalize = FALSE, intra.inter = FALSE)
```

Arguments

io	An object of class InputOutput calculated from as.inputoutput.
ES	An object of class EasySelect from easy.select
regions	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in RS_label or the number of the region in the order it appears in RS_label.
sectors	Character or Integer. Specific sectors to be used. Can either be a character that exactly matches the name of the sector in RS_label or the number of the sector in the order it RS_label.
type	Character. Identifying the type of backward and forward linkages to be calculated. Options are "total" and "direct".

`normalize` Logical. Identifying whether or not to calculate normalized or raw linkages. Default is `TRUE`

`intra.inter` Logical. Only applies to multiregional systems. Determines whether or not to calculate intraregional and interregional backward and forward linkages in addition to aggregate linkages.

Details

There are arguments for `type` of linkages, `normalized` linkages, and `intra.inter` linkages. Let (r) denote the dimension of the block in the transaction matrix of the region of interest and (s) denote the dimension of the rest. If there are (n) sectors and (m) regions then r = n and s = (m - 1)*s

`type`: For the following types, if `normalize = TRUE` then the calculation takes the specified form below. Otherwise if `normalize = FALSE` then the denominator is removed:

`"total"` caclculates the total backward and forward linkages. For backward linkages, this is the column sum of the Leontief inverse.

$$BL_j = \frac{\sum_{i=1}^{n} l_{ij}}{\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{n} l_{ij}}$$

For forward linkages, this is the row sum of the Goshian inverse.

$$FL_i = \frac{\frac{1}{n}\sum_{j=1}^{n} g_{ij}}{\frac{1}{n^2}\sum_{j=1}^{n}\sum_{i=1}^{n} g_{ij}}$$

`"direct"` calculates the direct backward and forward linkages. For backward linkages, this is the column sum of the input matrix of technical coefficients (A):

$$BL_j = \frac{\sum_{i=1}^{n} a_{ij}}{\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}}$$

For forward linkages, this is the row sum of the output matrix of technical coefficients (B):

$$FL_i = \frac{\frac{1}{n}\sum_{j=1}^{n} b_{ij}}{\frac{1}{n^2}\sum_{j=1}^{n}\sum_{i=1}^{n} b_{ij}}$$

`intra.inter`: This calculates the intraregional, interregional and aggregate backward and forward linkages. If `intra.inter = FALSE`, then only calculates the aggregate. If `normalize = FALSE` then the aggregate linkage is equivalent to the sum of the intraregional and interregional linkages. If `normalize = TRUE`, then this is not the case. Note that normalizing adds the denominator to the following equations. Using matrix notation we have

$$BL.intra = \frac{1_r' J_{rr}}{\frac{1}{n*m} 1_r' J_{rr} 1_r}$$

$$FL.intra = \frac{J_{rr} 1_r}{\frac{1}{n*m} 1_r' J_{rr} 1_r}$$

$$BL.inter = \frac{1_s' J_{sr}}{\frac{1}{n*m} 1_s J_{sr} 1_r}$$

$$FL.inter = \frac{J_{rs}1_s}{\frac{1}{n*m}1_r J_{rs} 1_s}$$

$$BL.agg = \frac{1J_{.r}}{\frac{1}{n*m}1J_{.r}1_r}$$

$$FL.agg = \frac{J_{r.}1}{\frac{1}{n*m}1_r J_{r.}}1$$

Value

Returns a `data.frame`. The following are assigned to the column names to help identify which column is belongs to which. The first element of the column label is the region of interest, grabbed from `RS_label`.

`.BL`	Backward linkages
`.FL`	Forward linkages
`.intra`	Intraregional linkages
`.inter`	Interregional linkages
`.agg`	Aggregate linkages
`.tot`	Total linkages
`.dir`	Direct linkages

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

`leontief.inv`, `ghosh.inv`, `key.sector`

Examples

```
data(toy.IO)
class(toy.IO)
link1 <- linkages(toy.IO)
link1$Hogwarts

data(toy.ES)
class(toy.ES)
link2 <- linkages(toy.IO, toy.ES)
```

```
link2

# More detailed
link3 <- linkages(toy.IO, regions = "Narnia", sectors = c("Wii","Pizza"),
                  type = c("total", "direct"), normalize = FALSE, intra.inter = TRUE)
link3

link4 <- linkages(toy.IO, regions = 1:2, sectors = c(1:3,5))
link4
```

`locate.mismatch` *Identify Sectors not in All Regions*

Description

`locate.mismatch` finds which sectors are not found in all regions. If a sector is not in all regions a report is generated to indicate which regions have that sector, which regions don't have that sector, and where this sector is in the repository.

Usage

```
locate.mismatch(io)
```

Arguments

`io`	An object of class `InputOutput` created from `as.inputoutput`.

Details

`locate.mismatch` begins by identifying all sectors. Then if a sector is not in every region, the function identifies which regions have the sector, which regions don't have the sector, and where this sector is located. If it is important to have all regions having the same sectors, the location output can be used in `agg.sector`. For a full list of sectors, use `easy.select`.

Value

Produces a list of sectors. Each sector has a list of `location`, `regionswith`, and `regionswithout`. For example to find the regions that have a mismatched sector, use

`(mismatch.object)$sector$regionswith`

Author(s)

John J. P. Wade

See Also

`as.inputoutput`, `agg.sector`, `easy.select`

Examples

```
data(toy.IO)
class(toy.IO)
# No mismatches
MM1 <- locate.mismatch(toy.IO)

# Making toy.IO have mismatches
toy.IO$RS_label <- rbind(toy.IO$RS_label,
                         c("Valhalla", "Wii"),
                         c("Valhalla", "Pizza"),
                         c("Valhalla", "Pizza"),
                         c("Valhalla", "Minions"))
MM2 <- locate.mismatch(toy.IO)
MM2$Lightsabers
```

---

lq *Simple Location Quotient Updating*

Description

Uses simple linear quotient technique to update the matrix of technical input coefficients (A)

Usage

```
lq(io)
```

Arguments

io	An InputOutput class object from as.inputoutput

Details

Uses the simple linear quotient technique as follows:

$$lq_i = \frac{X_i^r / X^r}{X_i^n / X^n}$$

where X^n is the total production, X^r is the total production for region r, X_i^r is the production for region r sector i, and X_i^n is the total production for the ith sector.

Then lq is converted such that if $lq_i > 1$, then $lq_i = 1$. Then lq is converted into a diagonal matrix of values less than or equal to 1, which gives us our final results

$$\hat{A} = Alq$$

Value

Produces the forecast of the matrix of technical input coefficients (A) using the Slq technique.

Author(s)

John J. P. Wade

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

ras

Examples

```
data(toy.IO)
class(toy.IO)

Anew <- lq(toy.IO)
```

---

mpm *Multiplier Product Matrix*

Description

mpm calculates the multiplier product matrix using an InputOutput object calculated from as.inputoutput. The method is described below.

Usage

```
mpm(io)
```

Arguments

io	An InputOutput class object from as.inputoutput

Details

Let L be the Leontief inverse. Then the multiplier product matrix M is calculated as follows:

$$M = 1/vL_cL_r$$

where $v = t(1)L1$ such that 1 is a column matrix of ones, $L_c = L1$ is a column matrix of row sums, and $L_r = t(1)L$ is a row matrix of column sums.

Value

M	Multiplier Product Matrix

Author(s)

John J. P. Wade

References

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

Examples

```
data(toy.IO)
class(toy.IO)
M <- mpm(toy.IO)
```

---

multipliers *Multiplier Analysis*

Description

multipliers is currently able to calculate four different multipliers: output, input, income, and employment. See details for formulas.

Usage

```
multipliers(io, ES, regions = "all", sectors = "all", multipliers, wage.row,
            employ.closed.row, employ.physical.row)
```

Arguments

io	An InputOutput class object from as.inputoutput
ES	An EasySelect class object from easy.select to specify which region and sector combinations to use.
regions	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in RS_label or the number of the region in the order it appears in RS_label.
sectors	Character or Integer. Specific sectors to be used. Can either be a character that exactly matches the name of the sector in RS_label or the number of the sector in the order it RS_label.
multipliers	Character. Any combination of the following: output, input, income, and/or employment

`wage.row` Integer. The row(s) in Value Added where wages is stored. See `io$V_label` if you do not know. This is not to be confused with the labor located in the intermediate transaction matrix (`Z`)

`employ.closed.row` Integer. The row(s) in the intermediate transaction matrix (`Z`) where labor is stored. This is not to be confused with "wages" or "employee compensation" etc.

`employ.physical.row` character or Integer. The row(s) in the phtsical matrix (`P`) where labor is stored. This is not to be confused with "wages" or "employee compensation" etc.

Details

There are four different multipliers able to be calculated:

(1) `output` - Output multipliers are calculated as the row sums of the Leontief matrix:

$$O_j = \sum_{i=1}^{n} l_{ij}$$

where l_{ij} is the ith row and jth column element of the Leontief matrix.

(2)`input` - Input multipliers are calculated as the row sums of the Ghoshian matrix:

$$I_j = \sum_{i=1}^{n} g_{ij}$$

where $g_i j$ is the ith row and jth column element of the Ghoshian matrix

(3) `income` - Income multipliers are calculated using value add due to employee compensation or wages. Multiple types of wages are supported. Wages are standardized and multiplied by the Leontief matrix:

$$W_j = \sum_{i=1}^{n} \omega_i l_{ij}$$

where $\omega_i = w_i / X_i$ is the wage divided by the total production for that region-sector combination, and l_{ij} is the ith row and jth column element of the Leontief matrix.

(4) `employment` - Employment multipliers are calculated using the employment row in the matrix of technical input coefficients (`A`):

$$E_j = \sum_{i=1}^{n} \epsilon_{ei} l_{ij}$$

where ϵ_{ei} is the row(s) corresponding to labor at the ith column, and l_{ij} is the ith row and jth column element of the Leontief matrix.

Value

Produces a list over regions of multilpliers.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

as.inputoutput, key.sector, linkages, output.decomposition

Examples

```
data(toy.IO)
class(toy.IO)
M1 <- multipliers(toy.IO, multipliers = "income", wage.row = 1)
M2 <- multipliers(toy.IO, multipliers = "employment.closed", employ.closed.row = "Minions")

data(toy.ES)
class(toy.ES)
M3 <- multipliers(toy.IO, toy.ES, multipliers = c("input", "output"))
```

---

output.decomposition *Decomposition of Output Changes*

Description

Performs decomposition of output changes given two periods of data. You can decompose by origin over internal, external, or total and you can additionally decompose by changes due to final demand, technical change, or total. This follows the technique of Sonis et al (1996).

Usage

```
output.decomposition(io1, io2, origin = "all", cause = "all")
```

Arguments

io1	The first period InputOutput class object from as.inputoutput
io2	An InputOutput class object from as.inputoutput
origin	Character. Choosing to decompose changes to the sectors due to internal changes, external changes, and/or total
cause	Character. Choosing to decompose changes to the sectors due to changes in fianldemand (f), technical changes leontief (L), or total changes

Details

A superscript of f indicates changes due to final demand, l indicates changes due to the Leontief inverse, and no superscript indicates total. A subscript of s indicates changes in output originating internally of the sectors, n indicates externally, and no subscript indicates total. L is the Leontief inverse and f is aggregated final demand. Analysis is over changes from period 1 to period 2. The values are calculated as follows:

Originating: Total

$$\Delta X^f = L_1 \Delta f$$

$$\Delta X^l = \Delta L f_1$$

$$\Delta X = \Delta L \Delta f$$

Originating: Internal

$$\Delta X_s^f = diag(L_1)\Delta f$$

$$\Delta X_s^l = diag(\Delta L) f_1$$

$$\Delta X_s = diag(\Delta L)\Delta f$$

Originating: External

$$\Delta X_n^f = \Delta X^f - \Delta X_s^f$$

$$\Delta X_n^l = \Delta X^l - \Delta X_s^l$$

$$\Delta X_n = \Delta X - \Delta x_s$$

Value

The function always outputs a named row of some variant of `delta.X`. A prefix indicates the changes origin where total is blank. A suffix indicates the cause of the change where total is also blank.

`int`	A prefix for internal
`ext`	A prefix for external
`f`	A suffix for final demand
`L`	A suffix for technical or Leontief

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. *PyIO. Input-Output Analysis with Python*. REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

Sonis, Michael & Geoffrey JD Hewings, & Jiemin Guo. *Sources of structural change in input-output systems: a field of influence approach*. Economic Systems Research 8, no. 1 (1996): 15-32.

See Also

as.inputoutput

Examples

```
data(toy.IO)
data(toy.IO2)
class(toy.IO)
class(toy.IO) == class(toy.IO2)

OD1 <- output.decomposition(toy.IO, toy.IO2)
OD1$Hogwarts

OD2 <- output.decomposition(toy.IO, toy.IO2, origin = "external",
                            cause = c("finaldemand","leontief"))
OD2
```

---

ras *ras Updating Proejcting*

Description

Uses the ras technique to update the matrix of technical input coefficients A. You must have knowledge of or forecasts for the following three objects: (1) row sums u1 of A, (2) column sums v1 of A, and (3) total production x1.

Usage

```
ras(io, x1, u1, v1, tol, maxiter, type, verbose = FALSE)
```

Arguments

io	An InputOutput class object from as.inputoutput
x1	Vector. The forecast for future total production of each region-sector combination, matching the X object in io
u1	Vector. The forecast for future row sums of the matrix of technical input coefficients in A from io
v1	Vector. The forecast for future column sums of the matrix of technical input coefficients in A from io
tol	Numeric. The tolerance for convergence. Default: 0.001
maxiter	Numeric. The maximum number of iterations to try for convergence. Defualt: 10000
type	Character. The type of norm to use for convergence. See ?norm. Default: "o"
verbose	Logical. If TRUE will print the iteration and norm at each step. This is useful if the dataset is large. Deafult: FALSE

Details

Uses the ras iterative technique for updating the matrix of technical input coefficients. This takes the form:

$$lim_{n \Rightarrow \infty} A^{2n} = lim_{n \Rightarrow \infty} [\hat{R}^n ... \hat{R}^1] A_t [\hat{S}^1 ... \hat{S}^n] = A_{t+1}$$

where $R^1 = diag(u_{t+1}/u_0)$, $u_0 = A_t X$, and $u_{t+1} =$u1. Similarly $S^1 = diag(v_{t+1}/v_0)$, $v_0 = X R^1 A_t$.

Each iteration calculates the full ras object; that is, 2 steps are caluclated per iteration.

See Blair and Miller (2009) for more details.

Value

Produces the forecast of the matrix of technical input coefficients given the forecasted row sums, column sums, and total production.

Author(s)

John J. P. Wade

References

Blair, P.D. and Miller, R.E. (2009). "Input-Output Analysis: Foundations and Extensions". Cambridge University Press

See Also

as.inputoutput, lq

Examples

```
data(toy.IO)
class(toy.IO)

set.seed(117)
growth <- 1 + 0.1 * runif(10)
sort(growth)

X <- toy.IO$X
X1 <- X * growth
U <- rowSums(toy.IO$Z)
U1 <- U * growth
V <- colSums(toy.IO$Z)
V1 <- V * growth

ras <- ras(toy.IO, X1, U1, V1, maxiter = 10, verbose = TRUE)
```

rsp *Regional Supply Percentage Updating*

Description

rsp uses the RSP technique to update the matrix of technical input coefficients A from an InputOutput object created from as.inputoutput. The function calls upon import.total and export.total to calculate the imports and exports.

Usage

```
rsp(io)
```

Arguments

io	An InputOutput class object from as.inputoutput

Details

The new matrix of technical coefficients is calculated as follows:

$$A_{new} = \hat{p}A$$

where $\hat{p}$ is a diagonal matrix with each diagonal componenet calculated as

$$p_i = \frac{X_i - E_i}{X_i - E_i + M_i}$$

Value

Anew	The updated matrix of technical input coefficients

Author(s)

John J. P. Wade

References

Nazara, Suahasil & Guo, Dong & Hewings, Geoffrey J.D., & Dridi, Chokri, 2003. "PyIO. Input-Output Analysis with Python". REAL Discussion Paper 03-t-23. University of Illinois at Urbana-Champaign. (http://www.real.illinois.edu/d-paper/03/03-t-23.pdf)

See Also

import.total, export.total

Examples

```
data(toy.IO)
class(toy.IO)

Anew <- rsp(toy.IO)
```

`toy.ES` *An example dataset of class* `EasySelect`

Description

An object of `EasySelect` class created from `easy.select`.

Usage

```
data("toy.ES")
```

Format

A character matrix with three columns and 5 rows with class `EasySelect`. The first row indicates which rows/columns of `toy.IO` are of interest. The second and third column are the regions and sectors that respectively match the the first column.

Examples

```
data(toy.ES)
class(toy.ES)
```

`toy.FullIOTable` *An example data set to illustrate as.inputoutput*

Description

This data is designed to be a small dimension worst case scenario. The numbers are saved as a string and there are many `NA`s floating around. The data itself was randomly generated.

Usage

```
data("toy.FullIOTable")
```

Format

An input output matrix with two regions, five sectors, four national accounts categories (including exports), four values added (including imports), and total production.

Details

toy,FullIOTable was created using the following code where `toy.FullIOTable` was created using the seed of `117` and `toy.FullIOTable2` was created using the seed `112358`

See Also

See also `toy.IO`, `as.inputoutput`

Examples

```
set.seed(117)
# Creating the T (transaction) matrix

T11 <- matrix(sample(1:100, 25), ncol = 5, nrow = 5)
T12 <- matrix(sample(1:100, 25), ncol = 5, nrow = 5)
T21 <- matrix(sample(1:100, 25), ncol = 5, nrow = 5)
T22 <- matrix(sample(1:100, 25), ncol = 5, nrow = 5)
Trd <- rbind(cbind(T11,T12),cbind(T21,T22))
# Creating Labels
region <- c(rep("Hogwarts",5),rep("Narnia",5))
sector <- c("Pizza","Wii","Spaceships","Lightsabers","Minions")
sector <- c(sector,sector)
id <- rbind(region,sector)
blank <- matrix(NA, ncol = 2, nrow = 2)
Trd <- rbind( cbind(blank, id), cbind(t(id), Trd))
# Creating value added matrix
V <- matrix(sample(100:300, 30), ncol = 10, nrow = 3)
label <- matrix(c("Employee Compensation", "Proprietor Income", "Indirect Business Tax"),
                ncol = 1)
blank <- matrix(NA, ncol = 1, nrow = 3)
V <- cbind(blank, label, V)
# Creating final demand matrix
f <- matrix(sample(1:300, 80), ncol = 8, nrow = 10)
label <- c("Household", "Government", "Investment", " Exports")
label <- matrix(c(label, label), nrow = 1)
id <- rbind(region[c(1:4,6:9)], label)
f <- rbind(id, f)
# Creating total production
one.10 <- matrix(rep(1, 10), ncol = 1)
one.8 <- matrix(rep(1, 8), ncol = 1)
X <- matrix(as.numeric(Trd[3:12, 3:12]), nrow = 10)%*%one.10 +
     matrix(as.numeric(f[3:12,]), nrow = 10)%*%one.8
label <- matrix(c(NA,"Total"))
X <- rbind(label, X)
# Creating imports (in this case it is a residual)
M <- matrix(NA, nrow = 1, ncol = 12)
one.3 <- matrix(rep(1, 3), ncol = 1)
M[1, 3:12] <- t(one.10)%*%matrix(as.numeric(Trd[3:12, 3:12]), nrow = 10) +
              t(one.3)%*%matrix(as.numeric(V[,3:12]), nrow = 3)
M[1, 2] <- "Imports"
# Putting this beast together
blank <- matrix(NA, nrow=5, ncol = 9)
```

```
holder <- cbind(f, X)
holder <- rbind(holder, blank)
hold <- rbind(Trd, V, M, t(X))
toy.FullIOTable <- cbind(hold, holder)
# Creating an FV matrix
a <- matrix(round(80*runif(12)), nrow = 2, ncol = 6)
toy.FullIOTable[15:16, c(13:15, 17:19)] <- a
```

`toy.IO` *An example dataset of class* `InputOutput`

Description

An object of `InputOutput` class created from `toy.FullIOTable` using `as.inputoutput`.

Usage

```
data("toy.IO")
```

Format

toy.IO is a list with 14 elements: 7 matrices and 7 labels.

Value

`Z`	Intermediate Transactions
`RS_label`	Column matrix of labels for region and sector
`f`	Final Demand
`f_label`	Row matrix of labels for accounts for `f`
`E`	Exports
`E_label`	Row matrix of labels for exports by sector and region for `E`
`X`	Total Production
`V`	Value added
`V_label`	Column matrix of labels for types of value added for `V`
`M`	Imports
`M_label`	Colum matrix of labels for type of imports for `M`
`A`	Technical Coefficients
`L`	Leontief Inverse

Examples

```
data(toy.IO)
class(toy.IO)
```

upstream *Upstreamness - Average Distance from Final Use*

Description

Measures upstreamness as in Antras et al. (2012), equation (9) page 5. The value is weakly bounded below by one, where a value close to one indicates it is near its final use on average and a higher value indicates it is further away from final use on average.

Usage

```
upstream(io, ES, regions = "all", sectors = "all")
```

Arguments

io	An InputOutput class object from as.inputoutput
ES	An EasySelect class object from easy.select to specify which region and sector combinations to use.
regions	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in RS_label or the number of the region in the order it appears in RS_label.
sectors	Character or Integer. Specific sectors to be used. Can either be a character that exactly matches the name of the sector in RS_label or the number of the sector in the order it RS_label.

Details

The upstreamness is calculated as follows, where, A is the matrix of technical input coefficients, X is total production, E is exports, and M is imports.

$$d_{ij} = a_{ij} \frac{x_i}{x_i + e_{ij} - m_{ij}}$$

$$U = (I - D)^{-1}$$

$$u_i = \sum_{j=1}^{n} U_{ij}$$

Value

Produces a list over regions of each region's sectors upstreamness measure.

Note

If the import (M) and/or export (E) is a matrix (i.e. not a nx1 vector) they are summed across region-sector combinations.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Pol Antras & Davin Chor & Thibault Fally & Russell Hillberry, 2012. *Measuring the Upstreamness of Production and Trade Flows*. NBER Working Papers 17819, National Bureau of Economic Research, Inc.

See Also

`as.inputoutput`

Examples

```
data(toy.IO)
class(toy.IO)
u1 <- upstream(toy.IO)
u1$Hogwarts
```

---

vs *Vertical Specialization*

Description

Calculates the vertical specialization share of total exports of each sector as described by Hummels et al. (2001), equation 3. Creates a value between zero and one to indicate relative specialization. For each region, a Leontief inverse is calculated. You need a multi-region input-output dataset for `vs` to be relevant.

Caution: Inverting large matrices will take a long time. Each individual hypothetical extraction requires the inversion of a matrix. R does a computation roughly every 8e-10 second. The number of computations per matrix inversion is n^3 where n is the dimension of the square matrix. For n = 5000 it should take 100 seconds. I trust you know how cubic functions grow.

Usage

```
vs(io, ES, regions = "all", sectors = "all")
```

Arguments

`io`	An `InputOutput` class object from `as.inputoutput`
`ES`	An `EasySelect` class object from `easy.select` to specify which region and sector combinations to use.
`regions`	Character or Integer. Specific regions to be used. Can either be a character that exactly matches the name of the region in `RS_label` or the number of the region in the order it appears in `RS_label`.

`sectors` Character or Integer. Specific sectors to be used. Can either be a character that exactly matches the name of the sector in `RS_label` or the number of the sector in the order it `RS_label`.

Details

The vertical specialization share of total exports is calculated as follows:

$$\frac{vs_r}{X_r^{total}} = \frac{1}{X_r^{total}} A_r^M L_r X_r$$

where X_r^{total} is the total exports for region r, A_r^M is the matrix of technical import coefficients, L_r is the domestic Leontief inverse calculated from the domestic matrix of technical coefficients i.e. A_{rr} not the full A matrix, and X_r is the vector of total exports.

Value

Creates a region list of vs share of total exports.

Author(s)

John J. P. Wade, Ignacio Sarmiento-Barbieri

References

Hummels, David & Ishii, Jun & Yi, Kei-Mu, 2001. *The nature and growth of vertical specialization in world trade*. Journal of International Economics, Elsevier, vol. 54(1), pages 75-96, June.

See Also

`import.coef`, `export.total`, `check.RS`, `leontief.inv`

Examples

```
data(toy.IO)
class(toy.IO)
(vs1 <- vs(toy.IO, regions = "all"))
vs1$Hogwarts
sum(vs1$Hogwarts)

data(toy.ES)
class(toy.ES)
vs2 <- vs(toy.IO, toy.ES)
vs2
```

Index