Iotables

Daniel Antal, CFA

October 23, 2017

Motivation

I would like to creat an open-source R package that makes working with Eurostat and OECD symmetric input-output tables very easy. The aim is to make various comparisons, for example in backward linkages or employment multipliers accross regions easy to compute. My 0.2 version works with 7 types of Eurostat input-output tables from 2010. I will soon add Hungarian, Croatian and OECD standard types. At the moment the bulk data can be organized into tidy SIOTs and some multipliers (value added, tax, employment, if data permits) and coefficients can be calculated.

I would like to release the code soon on CRAN and publish the documentation in the Journal of Statistical Software. I am looking for cooperation to use the software or test ie, i.e. to make comparable input-output economics research. I am mainly interested in creative and cultural industries, but the software works with any industry. It would be particularly interesting to re-run already published cases and complement them with international comparison.

Please get in touch on www.linkedin.com/in/antaldaniel/ or in email.

iotables

The symmetric input-output tables (SIOTs) are complex statistical products that present inter-related statistics in a predefined structure. They are often found in spreadsheets that follow this structure, or in the case Eurostat in a data repository. In both cases they in reproducible research must be downloaded and restructured to programmatically accessible form. Often these highly structured statistics need to be analyzed together with other data, for example, when employment effects and multipliers are calculated. In this case processing the employment data to SIOT conforming format is a significant preprocessing challenge.

The iotables are exactly designed for these tasks. Currently the package downloads and processes standardized European SIOTs conforming to the latest statistical regulations, i.e. SIOTs starting from the year 2010. The analytical functions work with any SIOTs but only standardized SIOTs have reproducible input functions. It is intended to later include European SIOTs from 1995 and OECD standardized SIOTs in later versions.

The aim of this introduction is not to introduce input-output economics, or SIOTs in detail. The Eurostat manual on input-output tables and the Eurostat tematic page should be consulted for further information about the data and the metadata.

In order to test the analytical functions of the package and to have a manageable sized example data set, we use the real-life data from the Eurostat manual. The `germany_1990` dataset is a simplified 6x6 sized SIOT taken from the Eurostat SIOT manual. The package function examples can be checked against Jörg Beutel's published results.

To show how to work with other IO tables and other data, we include the latest SIOT and employment statistics from Croatia with the package. Croatia joined the EU after 2010 and the country's SIOTs are following the Eurostat guidelines but are not available on Eurostat yet. Because they are almost identical to a Eurostat SIOT, they give a good opportunity to review the real-life processing challenges when you cannot rely on the Eurostat downloader. The Croatian data are found in several small datasets to show the filtering and joining of real-life data.

The Croatia vignette is not working with iotables 0.2 will be updated soon.

Installation

You can install iotools from github with:

```r
# install.packages("devtools")
devtools::install_github("antaldaniel/iotables")

#with vignettes:
#devtools::install_github("antaldaniel/iotables", build_vignettes = TRUE)
```

The package is not yet ready for CRAN publication, but it builds without errors and warnings on Windows, Mac OSX and Ubuntu / Linux, and will be released after sufficient testing is made with real-life examples. YOu can follow changes on the NEWS.md file.

You can download the CRAN-style documentation in a single PDF here.

Acquiring data

Because the input-output tables are highly structured statistics, their conversion to tidy data and making them available for bulk downloading is not straightforward. The package currently depends on the excellent `eurostat` package to download the latest Eurostat SIOT data. The Eurostat R package is a part of the rOpenGov Algorithms for Computational Social Science and Digital Humanities initiative.

Eurostat's data can be downloaded in several tidy, long-form, files, and a lot of filtering is needeed to start working with it. In case you need to work with a SIOT given in Excel, for example, Eurostat's pre-2010 files, which cannot be downloaded with the eurostat package, please consult the Croatia vignette. What you want to do is to arrive to a similar format to Eurostat's long-form.

The challenge is to filter out only the necessary rows, and then to spread out the long-form information in the exactly right order. Because in input-output economics we are working equations solved on symmetric matrix, the ordering of the rows and columns is very strict. If we mix up column number 63 with 64 we will get an erroneous result.

Currently the following Eurostat SIOTs can be used: "naio_10_cp1700", "naio_10_cp1750", "naio_10_pyp1700", "naio_10_pyp1750", "naio_cp17_r2", "naio_17_agg_60_r2", "naio_17_agg_10_r2". The tables follow a similar structure, but they are differing in two important aspects. The first quadrant may be derived from goods and services sales data based on COICOP codes, or from sales data of enterprises based on their NACE Rev 2. Industry codes. Eurostat uses different labelling for product x product, product x industry and industry x industry cases. You can preserve this labelling with choosing the labelling = short option.

Another major difference is the treatment of the use of imported inputs. These can be added to uses, or separately shown. The number of import rows is different in the A and B type SIOTs, and generally, correspondingly, also the export column in quadrant 2. In some SIOTs some rows and columns are present while others are not.

The different data sources and estimates applied will yield to slightly different SIOTs. However, once you acknowledge these differences, the further calculation is the same all the time. The analytical functions will always use the labelling = iotables option. This solution may be preferred in programmatic comparisons when the different Eurostat labelling would create a lot of exception handling. Eventually the differences in the original labelling of the Eurostat data do not apply to the content of the data, but they provide information on how they were estimated. So, for example, the data labelled A01 or CPA_A01 has the same economic meaning, but it is derived with a different estimation technique.

Currently only some Eurostat products works. This means that the sysdata.rda metadata file of this package understands their labelling, orders them correctly and spreads them out as they should be spread out. You can work with other IO tables, but in this case you have to make sure that you are ordering the symmetric matrix correctly, for example, in your spreadsheet software such as Excel.

There is currently no downloader function for the OECD STAN SIOT databases, which contain the SIOT's of some OECD countries, often overlapping with the EU countries who create their SIOTs, but this is planned in the near future. The code below gets the 2011 Australian SIOT in USD terms in long-form. The next pre-processing steps that are already available for Eurostat will be included in the 0.3 version of the package.

The Eurostat bulk downloader will give you a very long file: all country data in a long-form, with different currencies, and for different years. The `iotable_get` function will filter out a table for you.

- The default labelling = 'iotables' parameter will change all row and column names to a standard snake_var_name format naming. The alternative is to keep the 'short' Eurostat codes, but they vary accross statistical products, reflecting the COICOP, NACE or mixed source of the data, and sometimes the aggregation level, too.

The code above depends on the `get_eurostat()` function of the Eurostat package. This function call will create a temporary .rds file during your session to avoid too large data volumes. The `iotable_get()` filters out in the example of the Slovak IO table for the year 2010 in million Euros. Given that the Eurostat bulk file has all year and two currency units (MIO_EUR and MIO_NAC, which is equivalent in the eurozone member Slovakia) it is a rather large file.

You can access the temporary file name from the function message: `A temporary file is is saved as C:\Users\...AppData\Local\Temp\RtmpcPsWmy\naio_cp17_r2_SK_2010_MIO_EUR.rds.`

or create it yourself

```r
require (dplyr) ; require (iotables)

#Retrieve the bulk file from the temporary directory
retrieve_from_temp_bulk <- readRDS(paste0(tempdir(), "\\eurostat/naio_cp17_r2_date_code_TF.rds" ))

source <- "naio_cp17_r2"
#Get the temporary file name of any already existing temporary file with:
tmp_rds <- paste0(tempdir(), "\\", source,
          "_", geo, "_", year , "_", unit, ".rds")

#Retrieve the filtered file from the temporary directory
retrieve_from_temp <- readRDS(tmp_rds)
```

If you want to make comparisons among countries, over time or different price concepts, you will likely use a lot of data. The temp files are speeding up this process. If you make a reference to any of the tables in your R session the function will always first look for a temporary local file before downloading again the large bulk files (1.8-9 MB in size.)

The next version of the package will have a downloader for the OECD database that has input-output tabels for 61 countries. If you need to work with Vietnamese or Australian tables, you can give it a go with the following code based on Attilio Mattiocco's RJSDMX package. See alternatives:

A wrapper function with proper data wrangling will be provided later.

Metadata and preprocessing

The SIOTs are in fact four inter-related tables, or so-called "quadrants". The first quadrant has two forms, the product x product or industry x industry form. This are the inter-industry flows, or shortly input flows, or the first quadrant of the SIOT. Currently the package relies on the `dplyr` data wrangling package, but due to the size of the SIOTs, dplyr filters out the technology data slowly. Later this function should be re-written with `data.table`.

In the ESA 2010, the product-by-product input-output table is the most important symmetric input-output table and this table is described here. However, it should be noted that a few countries in the EU prefer to compile industry-by-industry tables. Eurostat uses CPA codes describing products and NACE codes describing industries.

The common parameter formats are given in the description of the functions dealing with the inter-industry flows, or first quadrant of the SIOT.

First quadrant or inter-industry flows

However, for several reasons, the SIOT's have a special aggregation scheme that requires the use of special coding. For some economic reasons, some industries and products are aggregated. This requires the use of special coding for the unusually aggregated industries or products. For example, the transport sectors of the economy are not directly addressed in the input flow.

The `use_table_get ()` function extracts only the input flow matrix from the table. The input-flow matrix contains the structural relations within an economy in a symmetrical matrix. For example, Eurostat's standard national input flow matrices have a dimension of 63 x 63, and this "field" is filtered out and pre-processed to a matrix. It is "pre" processing in the economic sense, because the user will most likely further manipulate the input flow matrix before writing actual equations.

Any further work usually requires aggregation, disaggregation following these technologies, which, from a data pre-processing point of view requires a lot filtering, joining. It is useful to start the analysis with getting the technology information and storing it in the global environment. The technology data can be added explicitly in each function call. This can save much computing time. If the technology data is not stored in the global or parent environment, each pre-processing function call will try to invoke this function.

The SIOT's do not follow a tidy format, because they are formatted in a way to facilitate matrix algebraic manipulations with the data. However, the standard fields of the SIOT's can be brought to tidy format. Pre-processing in this case means that the long-format bulk data is wrangled to the standard fields of the SIOT.

- The `unit` parameter is necessary because Eurostat files contain the same data in national currency units and euros.
- The `year` should be given as a numeric.
- The `geo` can be inputed with 2 letter country codes or country names. Beware, that Eurostat UK for the United Kingdom instead of GB and EL for Greece instead of GR.

Final demand or Second quadrant

The second quadrant contains the data of macroeconomic demand. Currently only the most often used vector, the final demand / output is processed. However, if you have some specific analytical need, you can filter out any details from the second quadrant following the logic of working with the first quadrant.

In all cases, the `households = TRUE` parameter will add the household consumption expenditure column, and in the case of the use table the wages (or compensation) row.

```
##   iotables_label_r agriculture_group manufacturing_group construcion_group
## 1        output_bp             43910             1079446            245606
##   trade_group business_services_group other_services_group
## 1      540063                  692487               508918
##   consumption_expenditure_household
## 1                           1001060
```

Third quadrant or primary inputs

The following function gets the primary input that you want to analyze. The `households` paramter select or deselects the household consumption expenditure column. In case it is selected, it set to default 0.

```
## # A tibble: 1 x 68
##   iotables_label_r agriculture forestry fishing mining
## *           <fctr>       <dbl>    <dbl>   <dbl>  <dbl>
## 1   wages_salaries      263.27    99.02    4.21 107.44
## # ... with 63 more variables: food_beverages_tobacco <dbl>,
## #   textiles_apparel <dbl>, wood_products <dbl>, paper_products <dbl>,
## #   printing_recording <dbl>, coke_refined_petrol <dbl>, chemical <dbl>,
## #   basic_pharmaceutical <dbl>, rubber_plasic <dbl>,
## #   mineral_products <dbl>, basic_metals <dbl>, fabricated_metal <dbl>,
## #   computer_electronic_optical <dbl>, electrical_equipment <dbl>,
## #   machinery <dbl>, motor_vechicles <dbl>,
## #   other_transport_equipment <dbl>, furniture <dbl>,
## #   repair_machinery <dbl>, eletricity_gas_steam <dbl>,
## #   water_services <dbl>, sewage <dbl>, construcion <dbl>,
## #   trade_motor_vechicles <dbl>, wholesale_trade <dbl>,
## #   retail_trade <dbl>, land_transport <dbl>, water_transport <dbl>,
## #   air_transport <dbl>, warehousing <dbl>, post_courier <dbl>,
## #   accommodation_food <dbl>, publishing <dbl>, audiovisual <dbl>,
## #   telecommunications <dbl>, computer_programing_consulting <dbl>,
## #   financial_services <dbl>, insurance <dbl>,
## #   auxiliary_financial_services <dbl>, real_estate_imputed_a <dbl>,
## #   `real_estate _services_b` <dbl>, legal_accounting_consulting <dbl>,
## #   architectural_engineering <dbl>, research_development <dbl>,
## #   advertising_marketing <dbl>, other_professional_services <dbl>,
## #   rental_leasing <dbl>, employment_services <dbl>,
## #   travel_agency_services <dbl>, security_investigation <dbl>,
## #   public_administration <dbl>, education <dbl>, human_health <dbl>,
## #   residential_care <dbl>, creative_industries <dbl>,
## #   sport_recreation <dbl>, membership_organizations <dbl>,
## #   repair_computer_durables <dbl>, other_personal_services <dbl>,
## #   household_services <dbl>, extraterriorial_organizations <dbl>,
## #   total <dbl>, consumption_expenditure_household <dbl>
```

Fourth quadrant

The fourth quadrant (primary inputs to final demand) is not used in version 0.2.

Combining the data for economics

Practitioners can easily combine the tidy data or the standard matrices returned by the pre-processing functions. The most frequently used, basic combinations are included in the package, following the function guidelines of rOpenSci. However, using the named versions of the returned objects, any further combination can be easily programmed with careful joining, and performing rowwise or columnwise arithmetic operations.

The biggest difficulty of working with SIOTs is usually the problem of (pre-)processing important axillary elements, for example the creattion of satellite accounts that strictly conform the SIOT's vectors or matrixes. For example, when used in matrix algebraic equations with the a 64x64 input-flow table, the employment data must be aggregated (or disaggregated) into a numerical vector that has exactly 64 elements without missing values. Currently the exception handling is designed to throw at least meaningful errors in the process.

The Eurostat SIOT's contain only the basic, but already very structured tables, which are in fact three conforming matrices. The futher axillary tables that are often used, such as employment or CO2 inputs are often found in in different Eurostat statistics, or they cannot be found altogether in Eurostat. The SIOTs have a unique formatting and naming conventions, but unless an EU or OECD member state creates the satellite account together with the SIOT, there are not formatting guidelines to help furher programming. In this case the aim of this package (with later additions) could be to provide even better error handling and meaningful helper functions, but the creation of the satellite accounts will in almost any scenario require further data processing or modelling not covered by the `iotables` package.

Currently only the `input_indicator_create ()` function is available which can carefully create the indicators by pairing the input vector with the output vector. [It is under reprogramming, will be corrected very soon. Not working in this form.]

```r
de_output <- output_get ( source = "germany_1990", geo = "DE",
                          year = 1990, unit = "MIO_EUR",
                          households = TRUE, labelling = "iotables")

de_wages <- primary_input_get ( input = "d1",
                          source = "germany_1990", geo = "DE",
                          year = 1990, unit = "MIO_EUR",
                          households = TRUE, labelling = "iotables")

#Wage_indicator <- iotables::input_indicator_create (de_wages, de_output, 4)

#print (employment_indicator)
```

In this case the employment indicator can be found on p499 in the Table 15.14. The combination of the data is not always straightforward.

Analytical functions

The most frequently used input-output equations are programmed into convenient R functions. In almost all statistics and data scientific context, data pre-processing is usually the most time consuming effort, and this is probably even more true in the case of SIOT's, because of their complexity. If all elements are in place, working with the elements is easy with R's matrix algebraic operators such as `%*%`. (Needless to say that in the othervise userful vectorization nature of R's main operators and functions is avoidable.) However, because of the high complexity of the data, any violation of matrix algebraic rules causes a very hard-to-detect problem. If a 63x63 matrix has a single faulty element, the mathematical error message of the matrix operation is not very helpful on localizing and the correcting the element in question. For this reason we create a few helper functions that cover the most used scenarios.

The `leontieff_matrix_create()` function creates the Leontieff-matrix from the appropriate SIOT fields. The `leontieff_inverse_create(L)` creates the inverse of the Leontieff-matrix which is used in all basic equations.

```r
de_use <- use_table_get ( source = "germany_1990", geo = "DE",
                          year = 1990, unit = "MIO_EUR",
                          households = FALSE, labelling = "iotables")

de_output <- output_get ( source = "germany_1990", geo = "DE",
                          year = 1990, unit = "MIO_EUR",
                          households = FALSE, labelling = "iotables")
```

```r
de_coeff <- input_coefficient_matrix_create( de_use, de_output, digits = 4)

L <- iotables::leontieff_matrix_create( technology_coefficients_matrix =
                                  de_coeff )

print (L)
```

```
##          iotables_label_r agriculture_group manufacturing_group
## 1       agriculture_group            0.9742             -0.0236
## 2     manufacturing_group           -0.1806              0.7178
## 3       construcion_group           -0.0097             -0.0068
## 4             trade_group           -0.0811             -0.0674
## 5 business_services_group           -0.0828             -0.0890
## 6    other_services_group           -0.0353             -0.0139
##   construcion_group trade_group business_services_group
## 1            0.0000     -0.0011                 -0.0010
## 2           -0.2613     -0.0761                 -0.0173
## 3            0.9842     -0.0098                 -0.0339
## 4           -0.0578      0.8622                 -0.0156
## 5           -0.1263     -0.1218                  0.7210
## 6           -0.0071     -0.0208                 -0.0217
##   other_services_group
## 1              -0.0015
## 2              -0.0597
## 3              -0.0180
## 4              -0.0413
## 5              -0.0672
## 6               0.9566
```

The Leontieff matrix shown here can be checked against the Eurostat manuals table 15.9 on page 487. (Beware, the ordering of the industries is different.)

```r
I <- leontieff_inverse_create(L)
print (I)
```

```
##          iotables_label_r agriculture_group manufacturing_group
## 1       agriculture_group        1.03390950          0.03501839
## 2     manufacturing_group        0.28968590          1.42923465
## 3       construcion_group        0.02070433          0.01910180
## 4             trade_group        0.12698600          0.12146217
## 5 business_services_group        0.18418167          0.20706399
## 6    other_services_group        0.04945504          0.02953987
##   construcion_group trade_group business_services_group
## 1        0.01000918 0.005052355             0.002987288
## 2        0.39622326 0.142036672             0.059631573
## 3        1.02897000 0.021085502             0.050076278
## 4        0.10646629 1.178454208             0.035494905
## 5        0.25032070 0.223971579             1.412619060
## 6        0.02175724 0.033111432             0.034164749
##   other_services_group
## 1          0.004422993
## 2          0.107427569
## 3          0.025014479
## 4          0.063154485
## 5          0.126826042
## 6          1.051529078
```

The Leontieff inverse can be checked against the Eurostat manual's table 15.10 on page 488. (Beware, the ordering of the industries is different.)

Finding an error in a 63x63 matrix can be a painstaking excersize. Basic violations of matrix algebra throw meaningful errors, however, processing errors often lead to credible, but wrong results. Because the matrix equations result in large solutions that are highly abstract in nature, spotting an error in an otherwise numerically credible result matrix is rather difficult. This is why the wrappers try to avoid mismatching of products or industry, and to provide more meaningful error handling. An important future development is to make these warning and error notes better. Adding further meaningful warnings would require testing from practitioners in real-life scenarios.

Using the analytical functions we can calculate the employment multipliers for the German economy in the year 1990.

```
de_emp <- primary_input_get ( input = "emp",
                        source = "germany_1990", geo = "DE",
                        year = 1990, unit = "MIO_EUR",
                        households = FALSE, labelling = "iotables")

de_emp_indicator <- input_indicator_create ( de_emp, de_output)

employment_multipliers <- multiplier_create ( input_vector  = de_emp_indicator,
                                              Im = I, digits = 4 )

#Identical to equation_solve (employment_indicator, I )

print (employment_multipliers)
```

```
##               iotables_label_r agriculture_group manufacturing_group
## 1 employment_total_multiplier            0.0326              0.0162
##   construcion_group trade_group business_services_group
## 1            0.0207      0.0237                  0.0112
##   other_services_group
## 1               0.0242
```

Checking against the Eurostat manual p502, indeed the agriculture's employment multiplier was the highest in Germany in the year 1990 with a value of 0.0326.

The package's `testthat` functions in fact are checking with the same rounding the functions against the Eurostat manuals's select values.

Testing and documentation

Testing the package has two important difficulties. One is related to the size of the tables: working with large SIOT's is resource intensive so some functionality cannot be usefully checked on CRAN.

For easier automated and human testing the example input-output table is imported from the Eurostat manual. The "CPA_" preffix makes the formatting similar to the Eurostat raw files. The "t_rows2" and "t_cols2" variable names (and their labelled pairs, i.e. "t_row2_lab" and "t_cols2_lab") follows the naming conventions of Eurostat, too.

```
require (tidyr) ; require (dplyr)
head ( germany_1990, 2)
```

```
##   t_rows2       t_rows2_lab           t_cols2 values       t_cols2_lab
## 1   cpa_a Agriculture group   agriculture_group   1131   Agriculture group
## 2   cpa_a Agriculture group manufacturing_group  25480 Manufacturing group
##   geo geo_lab       time    unit     unit_lab
## 1  DE Germany 1990-01-01 MIO_EUR Million euro
## 2  DE Germany 1990-01-01 MIO_EUR Million euro
```

The other problem is that the input-output economics problems are very complex and do not necessarily have existing solutions. A failure of a function may be due to a bug to be fixed in the code, or maybe the logical consequence of an input failure. For this reason the test functions are ran with well-documented examples from the literature. The SIOT's used in these examples and in testing are small, simple or simplified data which are included with the package. The expected test results are the published and rounded examples from the literature. Rounding is necessary because the underlying equations often use iterations that may yield slightly different results on different platforms.