Guide into the WIOD R-package

Sybren Deuzeman

1 Introduction

2 Installing the WIOD R-package

3 Load International Input Output Tables

The WIOD R-package comes with an easy way to easily import international input-output tables into our R environment. The data can be loaded in such a way that the different versions of international input-output tables can be used right away without the need to get the data into a workable format yourself. The data is, however, not delivered with the package as this would mean that the package becomes very large. Instead, the data is stored separately on the internet and can be downloaded with some pre-build functions.

There are two ways to use the WIOD R-package. The first is to download the data from the internet at use and the other is to create and use a local copy. Creating a local copy is advisable, since the data-files are relatively large. Since loading from the internet is the easiest to explain, we start with that. After that, we explain how to make a local copy and use that in a way very similar to loading from the internet.

3.1 Load from the Internet

In this section, we explain how to load the data using the internet and how the data itself is formatted. We start by downloading a international input-output table for a single year and version. Using this, we will explain the contents of an international. Next, we will explain how to load in a whole database of international input-output tables that is better suited for time-series analysis.

3.1.1 Load a Single IOT

To load a single IOT, we can use the following function call `iot <- load_iot("<version>", year)`. For example, we can load international input-output table for 2000 from the World Input Output Database (WIOD) 2016 edition international input-output table using

```
iot <- load_iot("WIOD2016", 2000)
```

This will load a large list that called `iot`. This list is the basic building block of the package as the build-up of these lists are the same for all different versions.

The content of this list is:

- `I`: a matrix with the intermediate inputs use in one industry from another industry.
- `FD`: a matrix with different the final demand uses of the products of a sector.
- `S`: a vector with the gross output of the sectors.
- `VA`: the value added of the sectors.

iot	list [12]	List of length 12
I	double [2464 x 2464]	5.88e+03 7.81e+01 1.09e+01 1.42e+01 8.20e+02 1.49e+01 1.87e+02 4.14e+01 1.85e-02 ...
FD	double [2464 x 220]	2.78e+03 7.02e+01 3.50e+02 1.38e+02 1.06e+04 5.89e+02 0.00e+00 0.00e+00 ...
S	double [2464 x 1]	28948 1506 1512 34013 34168 3734 ...
VA	double [2464 x 1]	12594 477 851 20542 9094 1229 ...
c	double [1]	44
cf	double [1]	44
n	double [1]	56
f	double [1]	5
year	integer [1]	2000
version	character [1]	'WIOD2016'
countries	character [44 x 1]	'AUS' 'AUT' 'BEL' 'BGR' 'BRA' 'CAN' ...
industries	character [56 x 2]	'A01' 'A02' 'A03' 'B' 'C10-C12' 'C13-C15' 'Crop and animal production, hunting a ...

Fig. 1: The content of a Input-Output Table after loading it

After that follow some parameter that can be used by the program.

- `c`: number of countries in the input-output table
- `cf`: number of countries to which final demand goes.
- `n`: the number of industries.
- `f`: the number of final demand categories.
- `year`: the year for which the international input-output table is.
- `version`: the version of the international input-output table.
- `countries`: vector with the countries in the order in which they appear in the input-output table.
- `industries`: vector with the industries in the order in which they appear in the input-output table.

3.1.2 List of IOTs

To obtain time-series, it is useful to load lists of IOTs instead of working with different single IOTs. The package comes with a function that can load such a list. To use the functions to export the results, one should also such lists instead of separate IOTs.

One can load such a list of IOTs via the following command `iots <- load_iots("<version>")`. To load the whole WIOD 2016 database at once, one can use the following code:

```
WIOD2016 <- load_iots("WIOD2016").
```

Sometimes you would only want to load the data for a few years. We can do so by adding a vector with the years you want. To download the data of the WIOD 2016 database from 2005 to 2010 only, the following can be used:

```
WIOD2016_0510 <- load_iots("WIOD2016", 2005:2010)
```

3.2 Load Extra Data

In some cases, supplemental data needs to be downloaded. For example, the WIOD project database also consists of the Socio-Economic Accounts. This data can be added to the list via one of two functions `load_extra_iot` or `load_extra_iots`. To load the Socio-Economic Accounts (SEA) to the already loaded `iot`, you can use:

```
iot <- load_extra_iot(iot, "SEA")
```

To load the Socio-Economic Accounts to a list of IOTs, i.e. to the already loaded `WIOD2016`, you can use:

```
WIOD2016 <- load_extra_iots(WIOD2016, "SEA")
```

3.3 Make a Local Copy

To use a local copy instead of data from the internet, one can first download the data from the internet. To do so, first set the directory where to store the international input-output tables using the `change_dir_data` function. For example, if one wants to store the data `c:`
`data_wiod`, use

```
change_dir_data("c:/data_wiod")
```

Then, to download a full dataset and its social-economic accounts, for example the WIOD 2016 data, one can use:

```
download_iots("WIOD2016")
download_extra_iots("SEA", "WIOD2016")
```

After this, the downloaded data can be accessed in the same way as using the online data. To access the data again in a new session, use `change_dir_data` again. Even manually downloading is not necessary. After using `change_dir_data` is used, the load functions described in the sections above will, if possible, automatically download missing data. Hence,

```
change_dir_data("c:/data_wiod")
WIOD2016 <- load_iots("WIOD2016")
```

is enough to download and load all the data from the World Input-Output Database 2016.

If you do no longer want to use local data, you can simply change back to using internet data via:

```
reset_dir_data()
```

4 Use International Input Output Tables

We want to use International Input-Output Tables to generate new measures. For this, one can either build an own function as described in 6 or use one of the build-in functions. The functions are build to work with a single international input-output table. However, with the function `on_iots` one can also use these functions on list of international input-output tables.

4.1 Use of package with single IOT

Here, we describe how to use a function and how these functions will behave in general. To do so, we consider the build-in function gii. This function calculates the global import intensity of a specific sector as in [[CITATION]]. We will use function with `iot`, which we already downloaded:

```
iot <- gii(iot)
```

iot	list [15]	List of length 15
I	double [2464 x 2464]	5.88e+03 7.81e+01 1.09e+01 1.42e+01 8.20e+02 1.49e+01 1.87e+02 4.14e+01 1.85e-02 ...
FD	double [2464 x 220]	2.78e+03 7.02e+01 3.50e+02 1.38e+02 1.06e+04 5.89e+02 0.00e+00 0.00e+00 ...
S	double [2464 x 1]	28948 1506 1512 34013 34168 3734 ...
VA	double [2464 x 1]	12594 477 851 20542 9094 1229 ...
c	double [1]	44
cf	double [1]	44
n	double [1]	56
f	double [1]	5
year	integer [1]	2000
version	character [1]	'WIOD2016'
countries	character [44 x 1]	'AUS' 'AUT' 'BEL' 'BGR' 'BRA' 'CAN' ...
industries	character [56 x 2]	'A01' 'A02' 'A03' 'B' 'C10-C12' 'C13-C15' 'Crop and animal production, hunting a ...
L	double [2464 x 2464]	1.28e+00 4.96e-03 1.03e-03 1.72e-02 4.82e-02 1.22e-03 1.70e-01 1.03e+00 4.34e-04 ...
gii	double [2464 x 1]	0.134 0.234 0.157 0.129 0.157 0.357 ...
gii_descr	character [2464 x 3]	'AUS' 'AUS' 'AUS' 'AUS' 'AUS' 'AUS' 'A01' 'A02' 'A03' 'B' 'C10-C12' 'C13-C15' 'C ...

Fig. 2: Content of `iot` after the command

We store the output of this function to the same list that we also used as argument for the function. We could have stored it as well into another list, which will then be a copy of the old list with new results added.

In figure 2, we show the contents of `iot` after using the function `gii`. Most part is as discussed in section 3.1.1. Three elements are added. The first is `L`, which is the Leontief inverse. As calculating the Leontief inverse is computationally intensive, we store the Leontief inverse for later use. Second, the element `gii` is added. In `gii`, the results of our calculations are stored. Third `gii_descr` is added. In `gii_descr`, we store the description of the data that can be used by our export functions.

The function `gii` also has two other arguments, namely `regions` and `industries`. These arguments add region and industry categories to `gii_descr`. To get a region categorization, one can use the function `countrycat`. As an example, we add a NAFTA and BeNeLux categorization:

```
NAFTA <- c("USA", "MEX", "CAN")
BENELUX <- c("BEL", "NLD", "LUX")
regions <- countrycat(list(NAFTA, BENELUX), iots)
```

to create a region categorization. Finally, we use the `regions` argument as

```
iot <- gii(iot, regions = regions)
```

to include the region categories to the data description.

4.2 Use of package with list of IOTs

If we use a list of international input-output tables, we can use the function `on_iots`. The function `on_iots` executes a function on all the input-output tables in the list. To use the function `gii` on all elements of the list `WIOD2016`, we can use

```
WIOD2016 <- on_iots(gii, WIOD2016)
```

To add extra arguments, like te region categories as before, one can add extra arguments to the end of the function call, i.e.:

```
WIOD2016 <- on_iots(gii, WIOD2016, regions = regions)
```

5 Export Results

In the package, results are stored in a list of lists. Without some help, it is therefore not easy to extract the data for further analysis. To help us, the package includes two functions: `export_dataframe` and `export_csv`. The first function exports the data to an R dataframe. The second function exports the data into a CSV file, such that the results can be loaded into another program for further analysis.

5.1 Export to Dataframe

To extract the earlier calculated global import intensities from `WIOD2016` into a dataframe, we can use:

```
gii_df <- export_dataframe("gii", WIOD2016)
```

This will generates a dataframe in "wide" format. Every year will have a separate column with its data. We can also export the data into "long" format. This does not store every year into a different variable, but gives the year for which it is in a separate column. This can be done via:

```
gii_df_long <- export_dataframe("gii", WIOD2016, long = TRUE)
```

This function can be used to export the results of different functions. To add, for example, the total final demand for certain sectors, one can use

```
WIOD2016 <- on_iots(fd_total, WIOD2016)
gii_fvas_df <- export_dataframe(c("gii", "fd_total"), WIOD2016)
```

If `gii` and `fd_total` would not have been of the same length, an error occurs. Further, the data description for `gii` will be used as this was the measure that is at the first position.

5.2 Export into a CSV file

The function `export_csv` shares the same basic setup. The most simple way to use it is

```
export_csv("gii", WIOD2016)
```

If one runs this code, a window will open to ask where to store the results. One can also add the file-path or filename directly via the directory argument:

```
export_csv("gii", WIOD2016, directory = "C://Some/Dir/and/File.csv")
```

6 Make Your Own Functions

The strength of this package is that you only need to write the function, while the rest of the package takes care of managing the data. The functions can be written for a single input-output table, while the rest of the package makes it easy to obtain results for a whole database of input-output tables.

To aid in writing your own functions, we show the source code of the `gii` function:

```
gii <- function(iot, regions = "None", industries = "None"){
        # Do necessary preliminary calculations
        A <- coefficient(iot)
        iot <- leontief(iot)

        # Get results
        temp = matrix(1, iot$c, iot$c) # Create a c x c matrix with ones
        for (i in 1:iot$c){
                temp[i,i]=0 # make the diagonal into 0's
        }
        S = kronecker(temp, matrix(1, iot$n, iot$n))
        result <- t(colSums(S * A) %*% iot$L)

        # Store results to the list
        iot$gii <- result
        iot$gii_descr <- countryindustry_descr(iot, regions, industries)

        # Return the list
        return(iot)
}
```

The code is split in four parts. We will discuss these parts shortly.

Do necessary preliminary calculations Here some necessary calculations are done. We need to find the so called coefficient matrix and the Leontief matrix to calculate the GII. We do not store the coefficient matrix, `A` in the IOT list, but we do so for the Leontief inverse `L`. We want to calculate the Leontief inverse only once as it is computationally heavy and therefore we store it for later use. Finding the coefficient matrix is not computationally heavy, so we can find it every time we need it.

Get results Here, we do some calculations to get the necessary results. One can use elements from the IOT list using `iot$element`. Observe that we use `colSums()` to find the sum over all elements of the columns. In R, element-wise manipulation is generally much faster than matrix manipulations. One should thus avoid matrix manipulation.

Store results to the list The results are stored to the list of the input-output table via `iot$gii <- result`. Next, we add a description via another function `iot$gii_descr <- countryindustry_descr(iot, regions, industries)`.

Return the list The list `iot` that contains now, next to the original content, the Leontief inverse, the GII calculations and its accompanying data description is returned as the result of the function call.