PYMRIO - MULTI REGIONAL INPUT OUTPUT ANALYSIS IN PYTHON

Konstantin Stadler
konstantin.stadler@ntnu.no
Industrial Ecology Programme, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Background

- In the last decades, globalization has led to complex supply chains spanning multiple countries. As a consequence, the environmental impacts of consumption are more and more determined by impacts occurring abroad.
- Environmental extended multi regional input output (EE MRIO) analysis emerged as the prevailing tool to account for all these impacts. This so called consumption based accounting or footprinting approach has been used in numerous studies.
- Compiling global MRIO is a complex task, and to date six such tables / databases have been compiled. Most of these databases are freely available. However, their structures differ from model to model, and handling/analysing MRIOs needs a certain degree of training.
- The pymrio module aims to ease the handling of global MRIOs. With just a couple of commands it is possible to calculate footprints of products, regions or capita. It allows to aggregate countries into regions (eg. EU) and to modify the sector classifications.

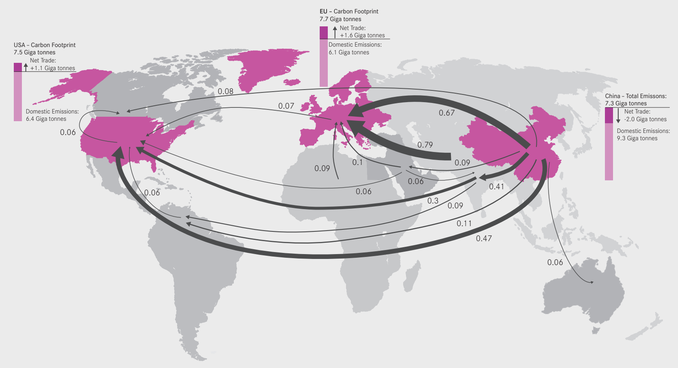

Fig. 1: Embodied carbon in trade based on the global MRIO EXIOBASE 2.
The figure shows the biggest net flows from the producing to the consuming regions.

MRIOs

- Input output (IO) tables describe whole economic systems. They express how much each sector produces (rows) and which input are needed for this production (columns).
- Environmental (and social) extensions enable the identification of primary resource use and required emissions. In an EE MRIO analysis these are subsequently allocated to the final consumers.
- The mathematical background for the analysis of MRIO has been developed decades ago, but up to now no tool for analysing these databases has been published.

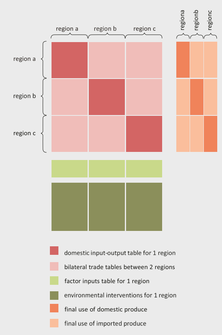

Fig. 2: Example MRIO with three regions

Implementation

The guiding principle for the development of pymrio was that an EE MRIO database can be represented as an object. Parsing a specific EE MRIO database into an instance of such an object guarantees a consistent API for every (MR)IO analysis.

- The top level object IOSystems includes the monetary part of the EE MRIO.
- Extensions are implemented as own objects within IOSystem.
- Internally, all data tables are stored as pandas DataFrames (http://pandas.pydata.org/).
- Meta data (base year, version, price system, ...) can be stored together with the data.

Functionalities

Parsing. So far, a parser for the recently published EXIOBASE 2 global MRIO has been implemented. Parser for other MRIO databases (e.g. WIOD, EXIOBASE 1, GRAM) are planned.

Calculation. One top level method checks for missing parts within the parsed MRIO and calculates all necessary tables for an EE MRIO analysis.

Restructuring extensions. Pymrio provides methods to delete and add certain extensions for a specific analysis and to cherry pick certain parts of an extension.

Aggregation. Aggregation of a MRIO can either be obtained by using some of the predefined correspondence matrices by passing a correspondence vector/matrix.

Exporting and saving. Since pymrio is based on pandas DataFrames, all output functions provided by pandas can be used to export tables. In addition, two top level functions save/load all data to/from either csv or pickle format.

Visualization and reporting. The visualization method extracts data from various datatables and allows to compare different perspectives (territorial, footprint, impacts embodied in trade) among countries. In addition, a report can be generated (html, latex, rst) for all stressor/impact data available in the MRIO database, either for absolute accounts or per capita.

Next steps

- Parser for various other global MRIOs.
- Automatic analysis of a MRIO time series.
- Test case for all mathematical function.
- Implement parallel programming techniques for some computational intensive functions.
- More graphical output capabilities. For example flow maps of impacts embodied in trade and choropleth maps for footprints.
- Advanced report functions for specific sectors/products.

Example

```
import pymrio   # the module
# parse EXIOBASE 2 from the raw data download
pxp = pymrio.parse_exiobase22(
      path = r'S:\rawexio',
      iosystem = 'pxp',
      version = 'exiobase 2.2',
      charact = r'S:\rawexio\charac.xlsx')

# The following command checks for missing
# parts in the system and calculates them.
pxp.calc_all()

# Aggregate the 48 countries of EXIOBASE
# to EU, BRIC and keep the remaining ones
reg_vector = mr.build_agg_vec(
        ['EU','BRIC','orig_regions'], path='exio2')
pxp.aggregate(region_agg=reg_vector,
        inplace = True, recalc = True)

# Visualize some of the data
pxp.impacts.plot_account(['global warming (GWP100)'])
```

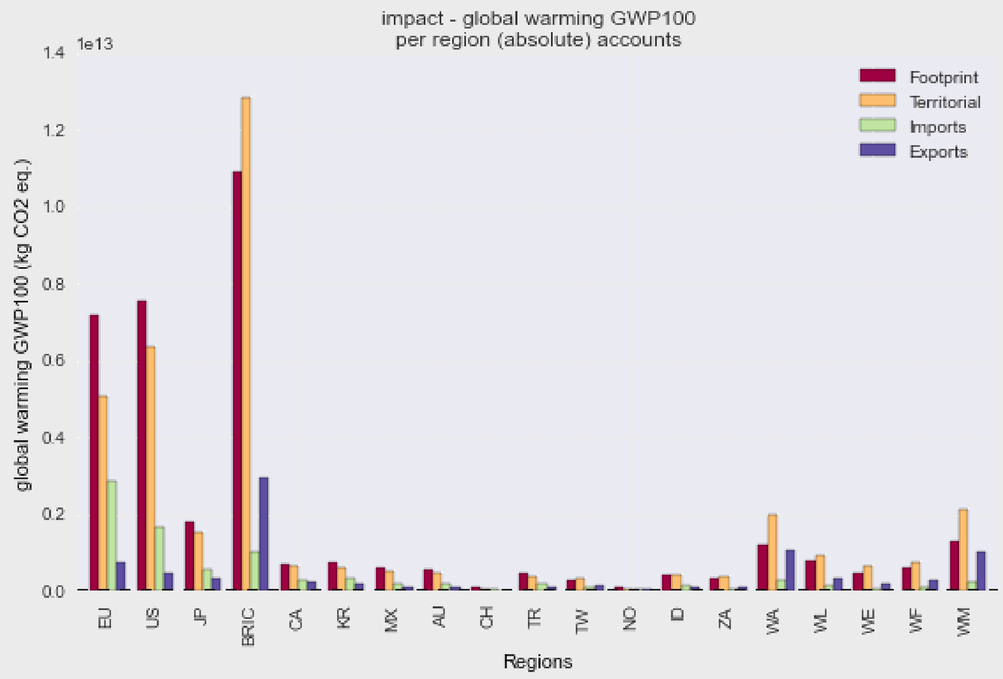

Fig. 3: Carbon footprint measured as GWP

Resources

pymrio: http://konstantinstadler.github.io/pymrio/

The Global Resource Footprint of Nations:
http://creea.eu/index.php/7-project/8-creea-booklet

EXIOBASE: http://exiobase.eu/

Funding

- EU 7th framework program DESIRE (http://fp7desire.eu/)
- NTNU