one planet
economy network

The EUREAPA Technical Report

COOPERATION

OPEN:EU

EUREAPA Methodology

17 OCTOBER 2011

Julian Briggs, Stockholm Environment Institute, University of York. UK

7th Framework Programme for Research and Technological Development
The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement N° 227065. The contents of this report are the sole responsibility of the One Planet Economy Network and can in no way be taken to reflect the views of the European Union.

Contents

1. Links

- GTAP :Global Trade Analysis Project www.gtap.agecon.purdue.edu/
- IO: Input-output Analysis: en.wikipedia.org/wiki/Input-output_model
- NTNU: Norwegian University of Science and Technology www.ntnu.edu
- OPEN-EU: One Planet Economy Network www.oneplaneteconomynetwork.org
- SEI: Stockholm Environment Institute sei-international.org/

2. Introduction

EUREAPA is an online scenario modelling and policy assessment tool created for the One Planet Economy Network. It uses a sophisticated economic input-output model to understand the environmental pressures associated with consumption activities, OPEN-EU (2011).

This document describes the methodology used by EUREAPA which is based on Environmentally Extended Input-Output methodology, Miller and Blair (2009).

We describe the EUREAPA data flow (derivation) in several stages:

1. Data flow overview: from GTAP to web front-end
2. Data flow in the EUREAPA Application
3. Data flow in the EUREAPA Engine

We describe preparation of data inputs for the engine from the NTNU OPEN-EU model (Weinzettel et at 2011) in .

3. Notation and Terms

EUREAPA uses the following notation and terms for vectors, matrices and n-dimensional data-blocks. Standard IO notation, following Miller and Blair (2009), is given in parentheses:

- **DIM**: Direct Impact Multiplier: Production Impact Intensity, for greenhouse gases the units are kilo-tonnes CO2 equivalent per million US$ of production (ktCO2e/US$million) (Direct Impact Coefficients, D)
- **Energy-Carrier**: a medium of energy supply (fuels, electricity and renewables). See (E)
- **FD**: Final Demand: expenditure by household, government or capital investment by product , US$million (f or y)
- **Footprint**: the impact resulting from a level and pattern of consumption, for GHG the units are ktCO2e (x*)
- **Impact**: an impact, e.g CO2, CH4 (ktCO2e), Ecological Footprint Land Type Crop (global hectares gha), Grey Water (m^3) (E)
- **Leontief**: Leontief inverse matrix, dimensionless (L)
- **Product**: consumption category: based on production sectors with several sectors disaggregated, see (x_i)
- **Sector**: production sector, see
- **TIM**: Total Impact Multiplier: Consumption Impact Intensity, for greenhouse gases the units are kilo-tonnes CO2 equivalent per million US$ of consumption (ktCO2e/US$million) (DL)
- **x**: production: value of production by region and sector in US$million (x)
- **Z**: prefix for concordance matrices used to transform values and dimensions of data matrices

4. Data flow overview: from GTAP to web front-end

Illustration 1: EUREAPA Data flow

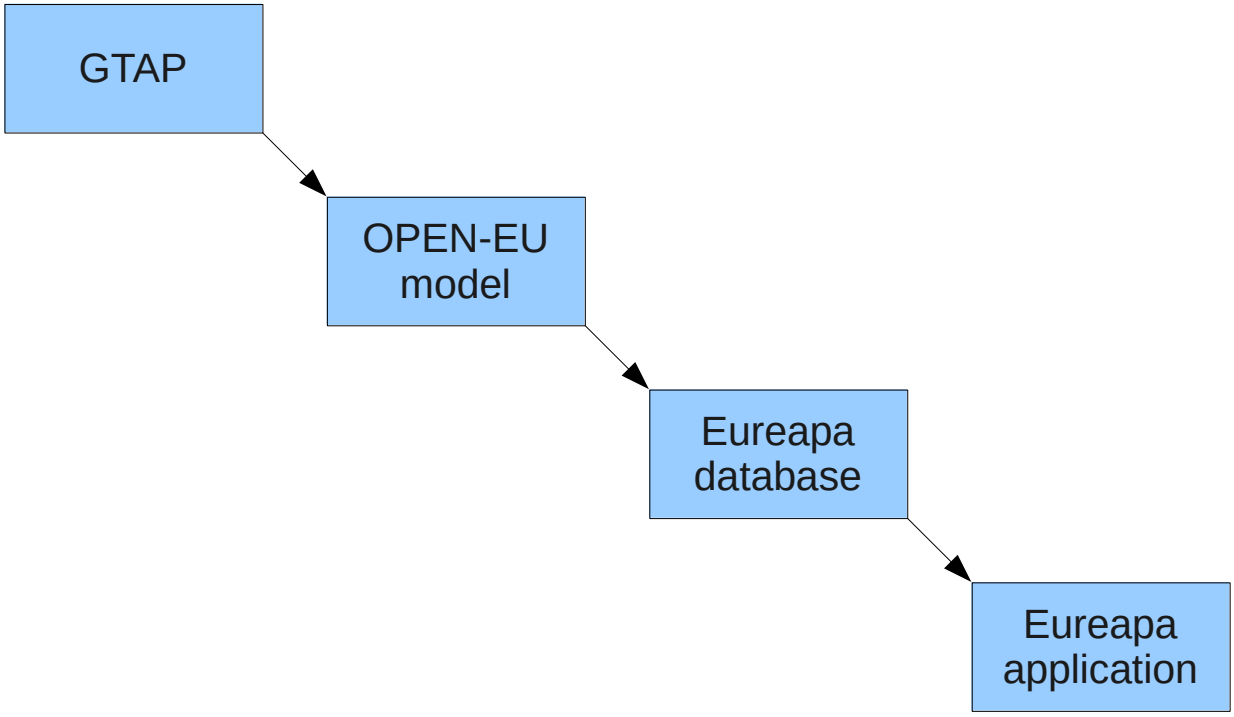


Illustration 4.1 shows an overview of the EUREAPA data flow. Stages are:

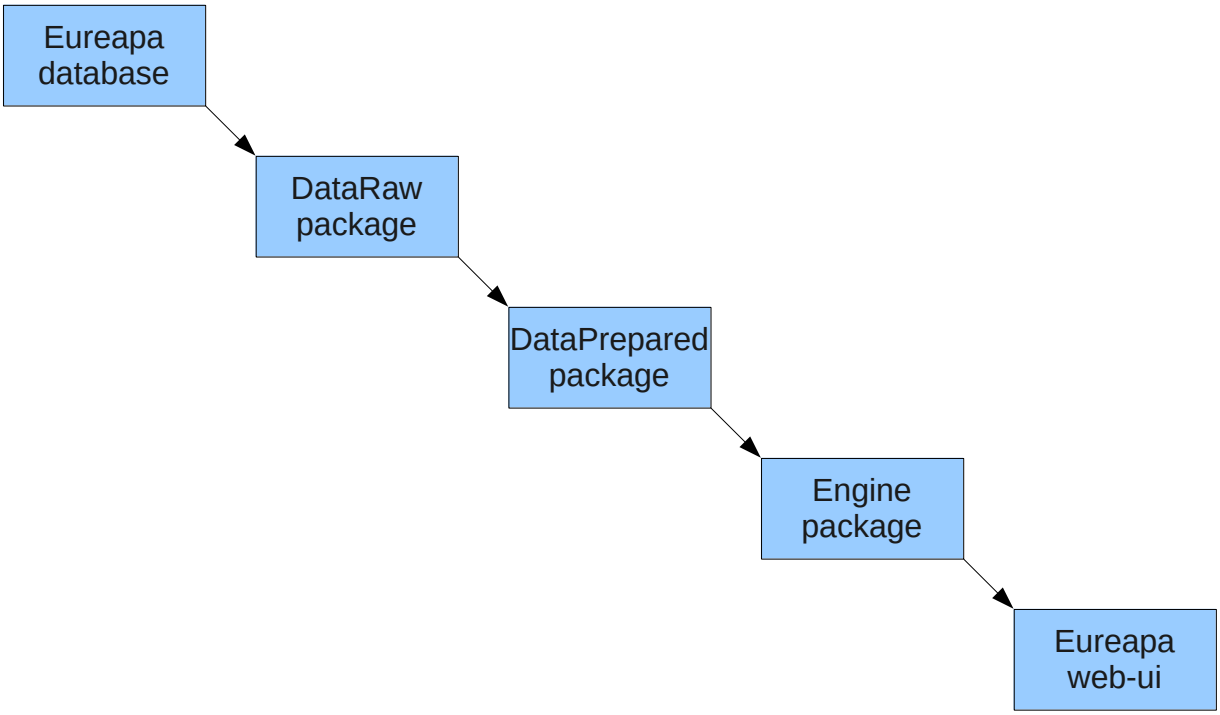
1. Global Trade Analysis Program 7 (GTAP7) database, Narayanan *et al* (2008) provides data for the ...
2. NTNU OPEN-EU Environmental Input-Output (EIO) model.
3. Stockholm Environment Institute (SEI) populates the EUREAPA database with OPEN-EU model results aggregated from 113 to 45 regions. This provides base-line data for the
4. EUREAPA Application which derives footprint results from the baseline data and user-specified scenarios.

5. Data flow in the EUREAPA Application

Illustration 5.1 Shows data flow in the EUREAPA Application:

1. The EUREAPA database provides base-line data to the DataRaw package
2. The DataRaw package provides this data to the DataPrepared package which prepares (assembles and derives data blocks) for the EUREAPA Engine
3. The EUREAPA Engine derives footprints from the prepared base-line data and user-specified scenarios
4. Finally the EUREAPA web-user-interface displays the data to the end user.

Illustration 2: EUREAPA Application Data Flow



6. Data flow in the EUREAPA Engine

The EUREAPA Engine derives footprints for several Final Demand (FD) consumer types:

1. Households
2. Government
3. Capital Investment (also includes small amounts of Valuables, Stocks and Trade)

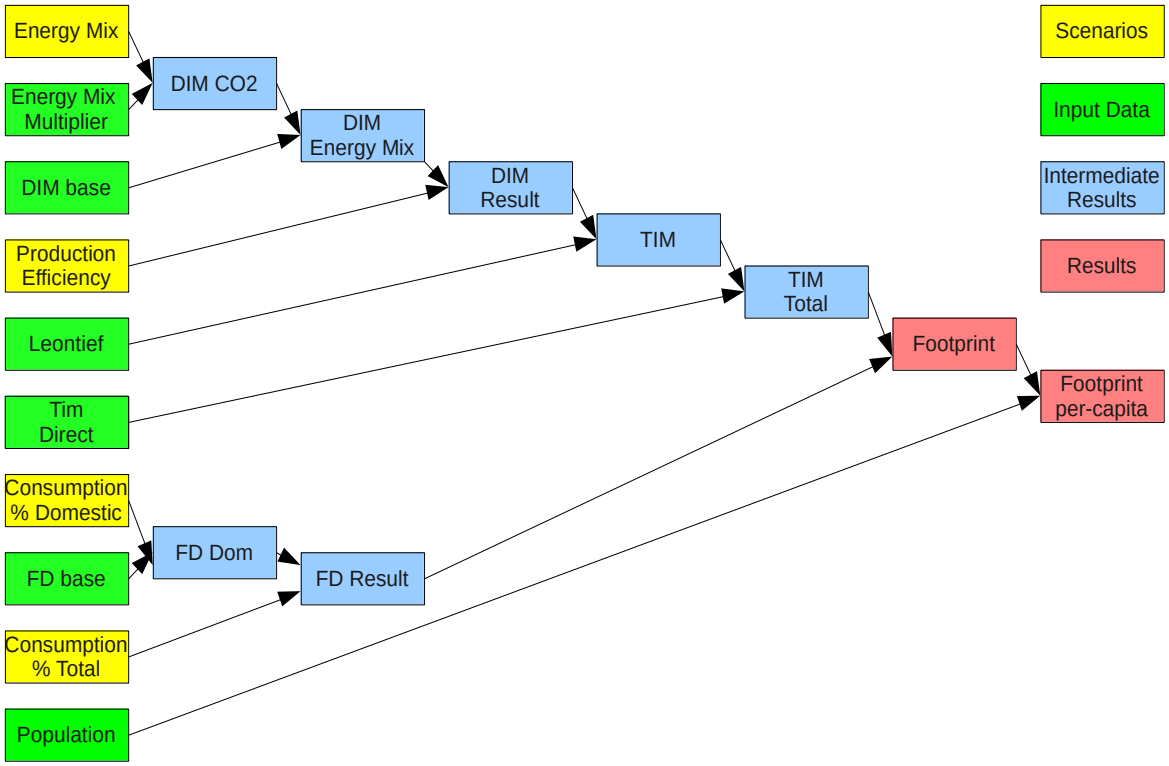
using:

- Impacts. See
- Energy Carriers. See
- Consumption Products (62): mostly as Production Sectors but with several disaggregated. See
- Production Sectors (57): GTAP 7 Data Base Sectors, e.g Wheat, Construction, Insurance,
- Regions (45). See

Base-line input data is updated with user-specified scenarios (energy-mix, production-efficiency, consumption) to derive intermediate results and final results. The engine provides per-capita results based on constant total final demand. The EUREAPA web-interface scales these by change in total expenditure and change in population.

EUREAPA represents data as large matrices (occasionally 3D or 4D arrays) which are selected, scaled and multiplied to derive intermediate and final results. The EUREAPA derives footprints from input data following the derivation chain shown in Illustration 6.1: EUREAPA Engine Data Flow.

Illustration 3: EUREAPA Engine Data Flow



Here we describe components of the EUREAPA Engine data flow:

1. Baseline input data:

   - Energy Mix Multiplier (EMM): CO2 intensity divided by Energy Mix(EM): by supply region, sector, and energy carrier. Such that DIM-CO2 = EMM * EM (see below). This matrix simplifies the derivation. The values in each column represent the CO2 intensity of the sector if it were to derive all its energy from that energy carrier.
   - DIM base: Baseline Direct Impact Multipliers: Impact intensities of production sectors by supply region, sector and impact. Units for CO2: kt CO2e/US$million.
   - Leontief Inverse Matrix: key element of Input-Output methodology. Used to derive TIM (see below) from DIM.
   - TIM (see below) direct: Direct impacts of consumption by each consumer type. E.g. CO2 emitted by stack or tailpipe owned by consumer: by FD region, supply region, sector, impact. E.g. land used to grow wheat bought by consumer.
   - FD base: Baseline Final Demand: Spending by consumer type, FD region, supply region and product.
   - Population: absolute population of FD region.

2. Scenarios data:

    - Energy Mix (EM): the proportions of energy supplied by: by supply region, sector and energy-carrier (e.g. coal, oil, gas, electricity, renewables)
    - Production Efficiency: the relative production efficiency (compared to baseline) by 3 footprint types (Carbon, Ecological and Water) by sector and region
    - Consumption Scenario % of total consumption: by consumer type, FD region, supply region, product.
    - Consumption Scenario % of consumption satisfied by domestic production: by consumer type, FD region, supply region, product.

3. Intermediate Results

    - DIM CO2: CO2 (from combustion) intensities by energy-carrier, region and sector-region. Summing these gives the CO2 (combustion) DIM. CO2 (process) is the non-combustion CO2 emitted by a few sectors. E.g. cement production where CO2 is released from heated limestone.
    - DIM-energy-mix: the DIM resulting from applying the energy mix scenario
    - DIM-Result: the DIM resulting from applying the production-mix scenario to the DIM-energy-mix
    - TIM: Total Impact Multiplier: Impact Intensities of consumption: by supply region, product and impact
    - TIM-Total: sum of indirect (TIM) and direct (TIM-direct) intensities: e.g, kt CO2e/US$ million: by FD region, supply region, sector, impact
    - FD-dom: Final Demand resulting from applying the consumption mix (proportion satisfied by domestic production) scenario
    - FD-result: Final Demand resulting from applying the consumption mix (proportion of total) scenario

4. Results:

    - Footprint: the footprints (CF, EF, WF): by consumer type, FD region, product, impact
    - Footprint per-capita: Footprint divided by the population of the FD region: by consumer type, FD region, product, impact

For further details of implementation of the methodology by the EUREAPA Engine, see the EUREAPA Engine Programmer Guide.

7. EUREAPA Engine Methodology

This section presents the EUREAPA Methodology by describing the derivation of each component in the derivation chain to derive footprints from the base-line data and user-specified scenarios.

7.1 Energy Mix

The Energy-Mix represents the contribution (as a proportion) that each energy-carrier makes to the total energy-use (in Mtoe, TJ etc.) within a sector-region. For example (with only 2 energy-carriers): coal 40%, gas 60%. The energy-intensities (ktCO2e/Mtoe) of Energy carriers differ, so changing the energy-mix changes the DIM for CO2 (combustion).

7.1.1 BASE-LINE ENERGY MIX

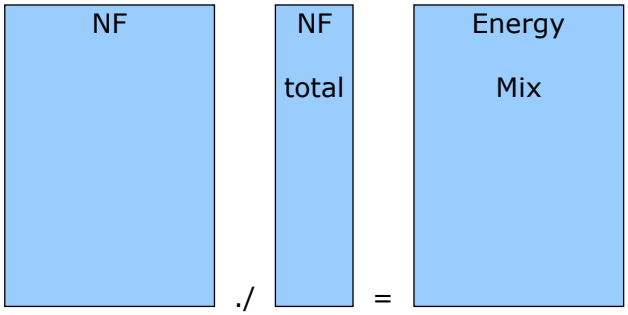
To derive the baseline Energy Mix:

- EM = NF / NF_Total

Where :

- EM: Energy Mix: proportion of total energy (dimensionless)
- NF: Energy Use (NF_coal, NF_gas .. NF_renewables): energy intensity (Mtoe/US$m)
- NF_Total: Total Energy Use (Mtoe/US$million)

thus:



The dimensions of all matrices are: sectorRegion(2565) x Energy Carrier(7)

7.1.2 ENERGY-MIX SCENARIO

Users take the base-line energy mix as a starting point for specifying a scenario Energy Mix. For example a scenario of de-carbonising the electricity generation sector by switching from coal fired power stations to renewables. This changes the DIMs for CO2 (and EF_Carbon which is handled later in the derivation chain). Here is a worked example (with 2 energy-carriers) of de-carbonising electricity generation:

Baseline CO2 DIM: 3(coal), 2(gas).

Total 5ktCO2e/US$million

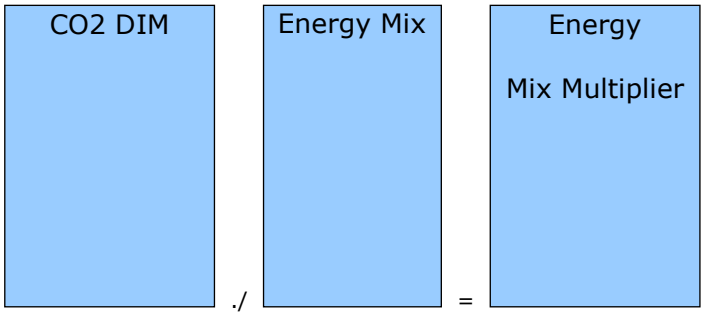
Baseline Energy-Mix: 40%(coal), 60%(gas)

End Energy-Mix: 20%(coal), 80%(gas)

End CO2 DIM: 3*(20%/40%) = 1.5(coal), 2*(60%/80%) = 2.67(gas). Total 4.17 ktCO2e/US$ million.

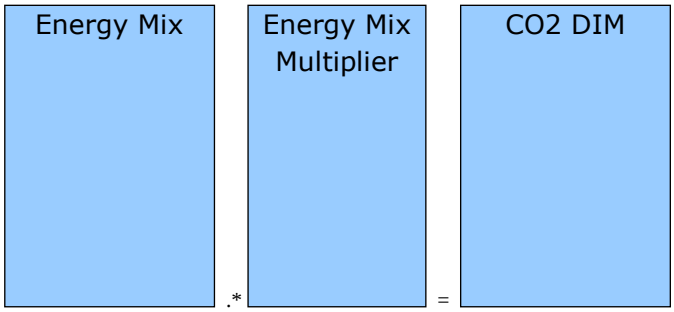
7.2 Energy Mix Multiplier

The Energy Mix Multiplier (EMM) is derived by dividing the base-line CO2 DIM by the base-line Energy Mix, thus:



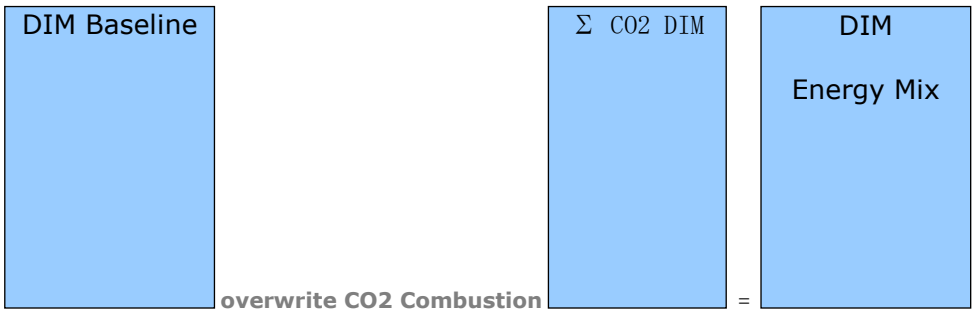
7.3. DIM CO2

DIM CO2 is derived by multiplying the Energy Mix (Scenario) by the Energy Mix Multiplier, thus:



7.4 DIM-ENERGY-MIX

The DIM Energy Mix is derived by updating the CO2column in the base-line DIM with the row sums of the CO2 DIM, thus:



We then derive the EF-Carbon values by multiplying CO2 values by a constant factor.

7.5 Production Efficiency

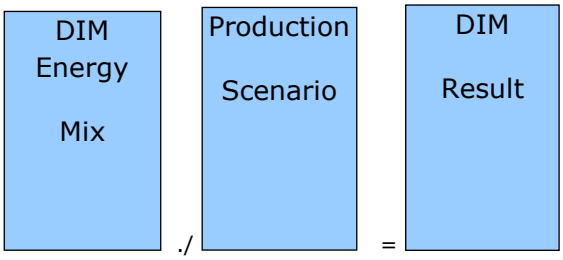
Production efficiency is the intensity of an industrial sector in a region, eg US steel production. Doubling the efficiency means halving the impacts from unit value (say US$million) of production, i.e. halving the DIMs. This is acts on the DIMs after they have been updated by the Energy-Mix Scenario.

The production scenario allows users to change 3 types of production efficiency by sector and (supply) region which affect different DIMs:

1. Carbon Footprint: scales the GHG (CO2, CH4, N2O, Fgas) and EF Carbon DIMs
2. Ecological Footprints: scales the EF real (excluding EF Carbon) land type DIMs
3. Water Footprint: Scales the Water DIMs

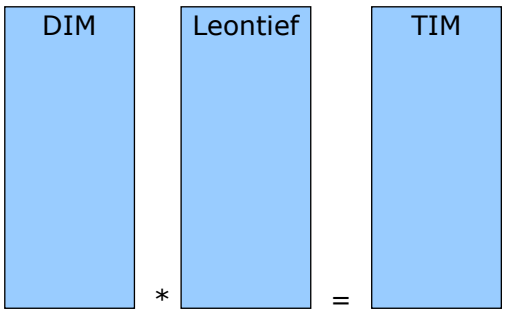
We derive the DIM results as affected by the production-efficiency scenario.

1. GHG DIM = GHG DIM * GHG Production Scenario
2. EF DIM = EF DIM * EF Production Scenario
3. WF DIM = WF DIM * WF Production Scenario



7.5.1 TIM

We derive the Indirect Total Impact Multiplier (TIM) by multiplying the DIM-Result by the Leontief Inverse Matrix., thus:



We transform the TIM from by sector to product by disaggregating several sectors (repeat rows). See .

7.6 TIM-Direct

The 4D array of direct Total Impact Multipliers (TIM-Direct) has values for impact intensities by Final Demand Region, Supply-Region, Product and Impact. Direct impacts are of several types:

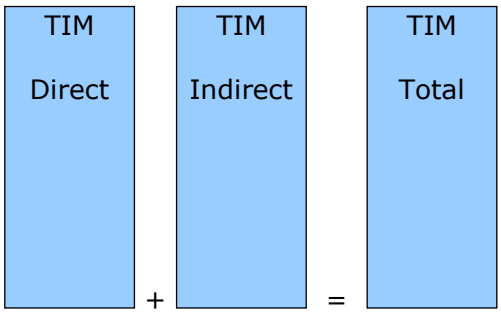
- Direct greenhouse gases (GHG) impacts are those emitted from a chimney or tail-pipe owned by a householder.
- Direct ecological footprint (EF) impacts are the notional bio-productive area of land (in global-hectares, gha) needed to supply or absorb the impact. For physical EF land types (EFCrop, EFPasture, EFForest. EFMarine) this land (or sea) is located in the region of supply. For Built land these are the land occupied by building (and garden) owned by the consumer.

The NTNU OPEN-EU model provides Direct impact intensities as a single matrix (2D) of Sector-Region x Impact-RegionFd, Fhh. We reshape this to a single 4D array: Sector x Supply-Region x Impact x Region of Final Demand (RegionFd) and rename it Tim-Direct.

We transform the TIM-Direct from by sector to product by disaggregating several sectors (repeat rows). See .

7.7 TIM-Total

We derive TIM-Total by adding Tim-direct and Tim (Indirect), thus:

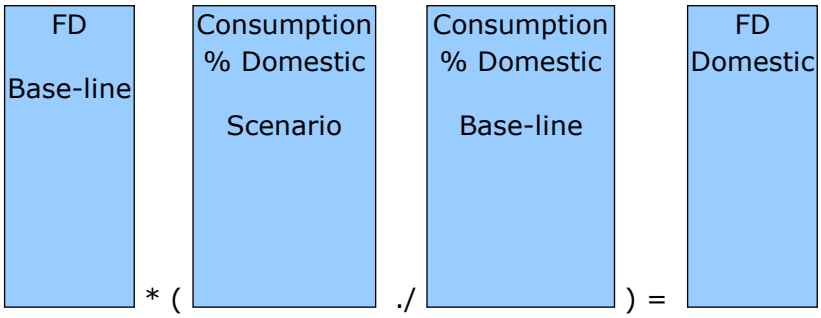


7.8 Consumption: % Domestic

Consumption: % Domestic represents the proportion of total consumption of a product in a Final Demand Region that is satisfied by domestic production. The base-line Consumption: % Domestic is derived by dividing the consumption satisfied by domestic production by the total consumption (satisfied by production from all regions). Users take the base-line Consumption: % Domestic as a starting point for domestic/import scenarios.

7.9 FD Domestic

We derive FD domestic by scaling consumption satisfied by domestic production by the change in Consumption % Domestic, thus:



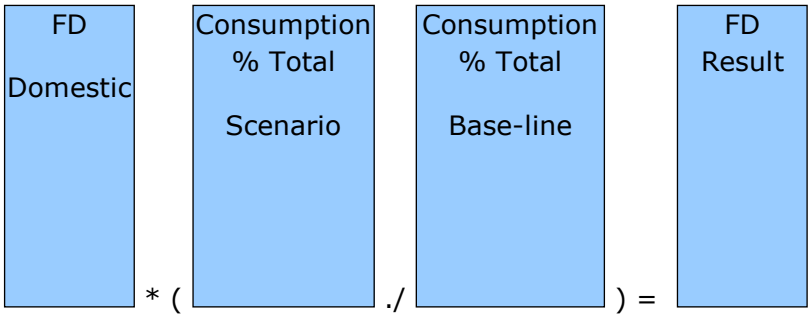
Consumption satisfied by imports (satisfied by each non-domestic region) are similarly scaled to ensure total consumption is unchanged.

7.10 Consumption % Total

Consumption % Total is the proportion of total consumption expenditure on each product, across all products (for each Final Demand Region and consumer type (household, government, capital)). For example expenditure on Wheat may be 2% of total household expenditure in UK. We derive base-line Consumption % Total by dividing the expenditure on a product (satisfied by domestic production and imports) by the total expenditure for each consumer type.

7.11 FD Result

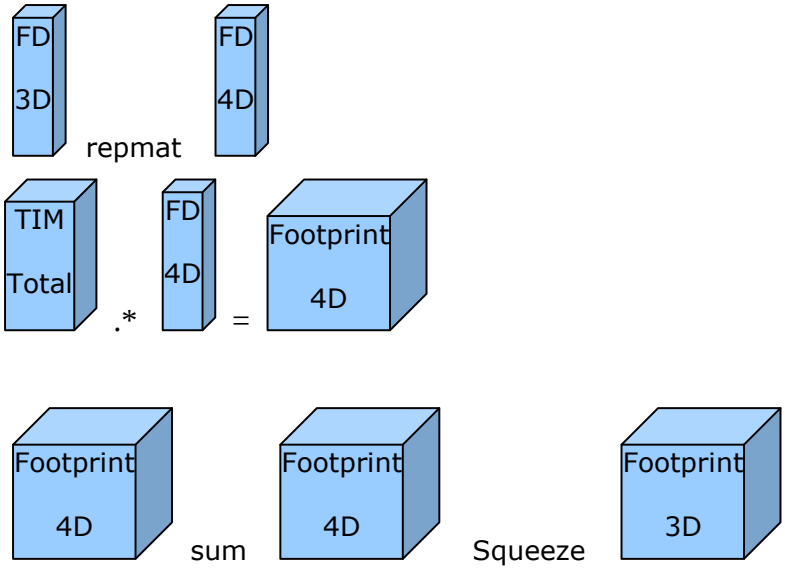
We derive FD Result by scaling FD Domestic by the change in Consumption % Domestic, thus:



7.12 Footprint

We derive footprint (FP) by multiplying TIM-Total by Final Demand (for each FD type: household, government and capital). To make dimensions match, we append Impact as a 4th dimension to FD. We then aggregate across sector to give a final 3D footprint.

1. FD 3D repmat FD 4D
2. TIM-Total .* FD = FP (4D)
3. FP 4D sum and squeeze to give FP 3D



Where:

- Repmat means repeating a data block in a given dimension
- Sum is summing across supply-regions
- Squeeze means removing the singleton dimension resulting from the sum

7.13 Footprint Per-capita

Finally we derive Footprint Per-capita by dividing the absolute footprint by the base-line population.

7.14 EUREAPA Engine Output

The engine provides per-capita results based on constant total final demand. The EUREAPA web-interface scales these by change in total expenditure and change in population.

8. References

Miller R. and Blair P. 2009. Input-Output Analysis: Foundations and Extensions. Cambridge University Press. UK

Narayanan, G. Badri and Terrie L. Walmsley, Eds. 2008. *Global Trade, Assistance, and Production: The GTAP 7 Data Base*, Center for Global Trade Analysis, Purdue University. Available online at: http://www.gtap.agecon.purdue.edu/databases/v7/v7_doco.asp

OPEN-EU 2011website http://www.oneplaneteconomynetwork.org/eureapa.html (accessed 19 September 2011).

Weinzettel et al 2011. Footprint Family Technical Report: Integration into MRIO model. Technical Document 7 February 2011. Jan Weinzettel, Kjartan Steen-Olsen, Alessandro Galli, Gemma Cranston, Ertug Ercin, Troy Hawkins, Tommy Wiedmann, Edgar Hertwich

9. Appendix 1: Impacts

1	CF CO2 Combustion
2	CF CO2 Process
3	CF CH4
4	CF N2O
5	CF FGases
6	EF crop
7	EF grazing
8	EF forest
9	EF marine
10	EF carbon
11	EF built
12	WF green
13	WF blue
14	WF grey

10. Appendix 2: Energy-carriers

8	NF_coal
9	NF_oil
10	NF_gas
11	NF_petroleum_products
12	NF_manufactured_gas
13	NF_electricity
14	NF_renewables

11. Appendix 3: Consumption Products

1	Paddy rice	
2	Wheat	
3	Cereal grains nec	
4	Vegetables fruit nuts	
5	Oil seeds	
6	Sugar cane sugar beet	
7	Plant-based fibers	
8	Crops nec	

9	Bovine cattle sheep and goats horses	
10	Animal products nec	
11	Raw milk	
12	Wool silk-worm cocoons	
13	Forestry	
14	Fishing	
15	Coal	
16	Oil	
17	Gas	
18	Minerals nec	
19	Bovine meat products	
20	Meat products nec	
21	Vegetable oils and fats	
22	Dairy products	
23	Processed rice	
24	Sugar	
25	Food products nec	
26	Beverages	Disaggregated: 26 Beverages and tobacco products
27	Tobacco products	Disaggregated: 26 Beverages and tobacco products
28	Textiles	
29	Wearing apparel	
30	Leather products	
31	Wood products	
32	Paper products publishing	
33	Petroleum coal products for home energy	Disaggregated: 32 Petroleum coal products
34	Petroleum coal products for transport	Disaggregated: 32 Petroleum coal products
35	Chemical rubber plastic products	
36	Mineral products nec	
37	Ferrous metals	
38	Metals nec	
39	Metal products	
40	Motor vehicles and parts	
41	Transport equipment nec	
42	Electronic equipment	
43	Machinery and equipment nec	
44	Manufactures nec	

45	Electricity	
46	Gas manufacture distribution	
47	Water	
48	Construction	
49	Trade in transport	Disaggregated: 47 Trade
50	Trade in food	Disaggregated: 47 Trade
51	Trade in goods	Disaggregated: 47 Trade
52	Trade in services	Disaggregated: 47 Trade
53	Transport nec	
54	Water transport	
55	Air transport	
56	Communication	
57	Financial services nec	
58	Insurance	
59	Business services nec	
60	Recreational and other services	
61	Public Administration Defense Education Health	
62	Dwellings	

12. Appendix 4: Production Sectors

1	PDR	Paddy rice
2	WHT	Wheat
3	GRO	Cereal grains nec
4	V_F	Vegetables fruit nuts
5	OSD	Oil seeds
6	C_B	Sugar cane sugar beet
7	PFB	Plant-based fibers
8	OCR	Crops nec
9	CTL	Bovine cattle sheep and goats horses
10	OAP	Animal products nec
11	RMK	Raw milk
12	WOL	Wool silk-worm cocoons
13	FRS	Forestry
14	FSH	Fishing
15	COA	Coal

16	OIL	Oil
17	GAS	Gas
18	OMN	Minerals nec
19	CMT	Bovine meat products
20	OMT	Meat products nec
21	VOL	Vegetable oils and fats
22	MIL	Dairy products
23	PCR	Processed rice
24	SGR	Sugar
25	OFD	Food products nec
26	B_T	Beverages and tobacco products
27	TEX	Textiles
28	WAP	Wearing apparel
29	LEA	Leather products
30	LUM	Wood products
31	PPP	Paper products publishing
32	P_C	Petroleum coal products
33	CRP	Chemical rubber plastic products
34	NMM	Mineral products nec
35	I_S	Ferrous metals
36	NFM	Metals nec
37	FMP	Metal products
38	MVH	Motor vehicles and parts
39	OTN	Transport equipment nec
40	ELE	Electronic equipment
41	OME	Machinery and equipment nec
42	OMF	Manufactures nec
43	ELY	Electricity
44	GDT	Gas manufacture distribution
45	WTR	Water
46	CNS	Construction
47	TRD	Trade
48	OTP	Transport nec
49	WTP	Water transport
50	ATP	Air transport
51	CMN	Communication
52	OFI	Financial services nec
53	ISR	Insurance

54	OBS	Business services nec
55	ROS	Recreational and other services
56	OSG	Public Administration Defense Education Health
57	DWE	Dwellings

13. Appendix 5: Regions

0	Austria	Austria
1	Belgium	Belgium
2	Bulgaria	Bulgaria
3	Cyprus	Cyprus
4	Czech Republic	Czech Republic
5	Denmark	Denmark
6	Estonia	Estonia
7	Finland	Finland
8	France	France
9	Germany	Germany
10	Greece	Greece
11	Hungary	Hungary
12	Ireland	Ireland
13	Italy	Italy
14	Latvia	Latvia
15	Lithuania	Lithuania
16	Luxembourg	Luxembourg
17	Malta	Malta
18	Netherlands	Netherlands
19	Poland	Poland
20	Portugal	Portugal
21	Romania	Romania
22	Slovak Republic	Slovak Republic
23	Slovenia	Slovenia
24	Spain	Spain
25	Sweden	Sweden
26	United Kingdom	United Kingdom
27	Australia	Australia
28	Brazil	Brazil
29	Canada	Canada
30	China	China

31	India	India
32	Indonesia	Indonesia
33	Japan	Japan
34	Mexico	Mexico
35	Norway	Norway
36	Russian Federation	Russian Federation
37	South Africa	South Africa
38	South Korea	South Korea
39	Switzerland	Switzerland
40	Taiwan	Taiwan
41	Turkey	Turkey
42	United States	United States
43	Annex Countries (I or II or B)	Annex Countries (I or II or B)
44	ROW	ROW

14. Appendix 6: EUREAPA Data Preparation Methodology

We use the following data preparation steps to derive EUREAPA data inputs from the NTNU model outputs. In the diagrams:

- blocks represent matrices
- matrix multiplication
- .* ./ cell-wise (dot) matrix multiplication and division

To derive baseline input data, we aggregate NTNU intensity (F) and production (x) matrices with 113 supply regions and 36 impacts to EUREAPA DIM (with CO2 intensities by energy carrier) and x matrices with 45 supply regions and 14 impacts (7 real impacts + 7 CO2 DIM (by energy carrier) + 7 energy use DIM (by energy carrier) . Energy carriers (EC) and impacts are shown in and . Derivation uses Octave thus:

1. Derive emissions as intensities * industrial output: E = F_ag .* diag(x)
2. Aggregate emissions: E_ag = Z_2565_6441 * E
3. Aggregate industrial output: x_ag = Z_2565_6441 * x
4. Derive DIM as intensities / industrial output: = E_ag ./ diag(x_ag)
5. Adjust the impacts in the DIM: insert CF intensities electricity and renewables, drop energy carriers: F_ag = F * Z_36_27
6. Update the CO2 DIM column with the sum of the EC carbon footprint (CF) DIM columns

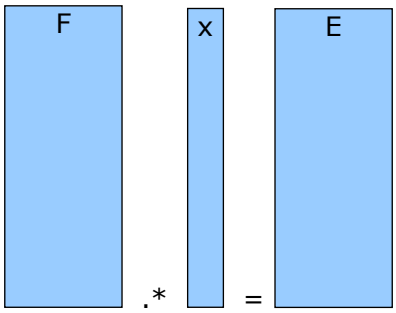
7. Update the Ecological Footprint Fossil Fuel (EFFF) DIM column by scaling it by the change in the CO2 DIM column

Where:

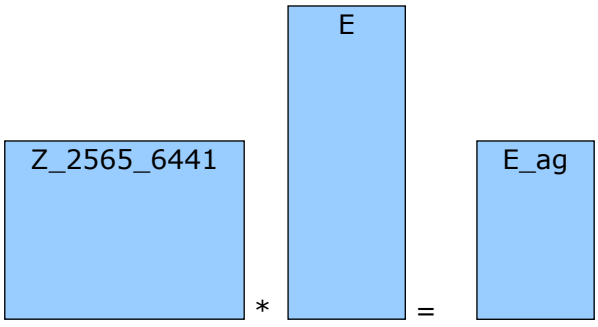
- Input data:
  - F: Intensities (NTNU version of DIM) (ktCO2/US$m,or gha/US$m or Mtoe/US$m): Sector(57)Region(113) x Impact(36)
  - x: Industrial production (US$m): Sector(57)Region(113)
- Concordance Matrices:
  - Z_2565_6441: aggregates from 57x113 to 57x45 sectors and regions
  - Z_36_27: aggregates from 36 to 27 impacts
- Output data:
  - DIM: Direct Intensity Multiplier (ktCO2/US$m,or gha/US$m or Mtoe/US$m): Sector(57)Region(45) x Impact(36)
  - x: Industrial production (US$m): Sector(57)Region(45)

We show this diagrammatically below.

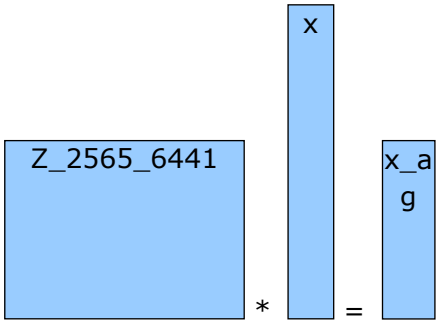
1. E(6441x36) = F(6441x36) .* diag(x(6441x1))



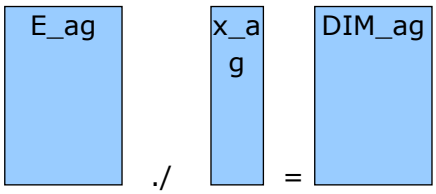
2. E_ag(2565x36) = Z_2565_6441 * E(6441x36)



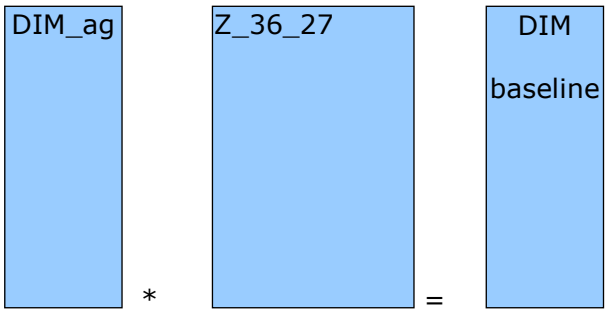
3. x_ag(2565) = Z_2565_6441 * x(6441)



4. Dim_ag(2565x36) = E_ag(2565) ./ diag(x_ag(2565))



5. DIM(2565x27) = DIM_ag(2565x36) * Z_36_27



OPEN:EU

EUREAPA Engine Programmer's Guide

17 OCTOBER 2011

Julian Briggs, Stockholm Environment Institute, University of York. UK

7th Framework Programme for Research and Technological Development
The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement N° 227065. The contents of this report are the sole responsibility of the One Planet Economy Network and can in no way be taken to reflect the views of the European Union.

Contents

1. Introduction

This EUREAPA Engine Programmer Guide describes the EUREAPA Engine, the Java program which implements the EUREAPA Engine Methodology described in the EUREAPA Engine Methodology Guide. It assumes familiarity with the Java programming language and access to the source code.

1.1 EUREAPA Data Derivation Overview

Illustration 1 EUREAPA Data Derivation Overview:

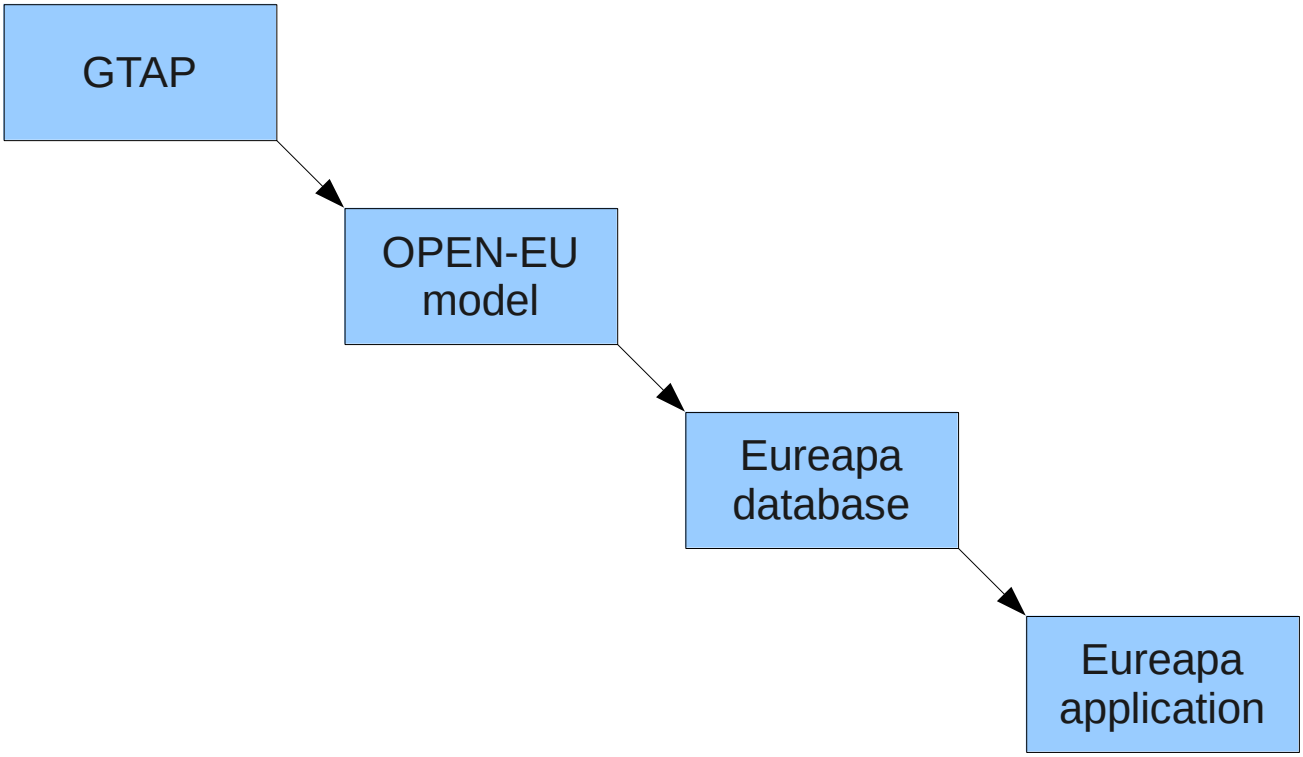


Illustration 1 locates the EUREAPA Application in the overall EUREAPA Data Derivation chain. For details of other components of the overall derivation chain see the EUREAPA Engine Methodology Guide.

1.2 EUREAPA Packages

Illustration 2: EUREAPA Packages

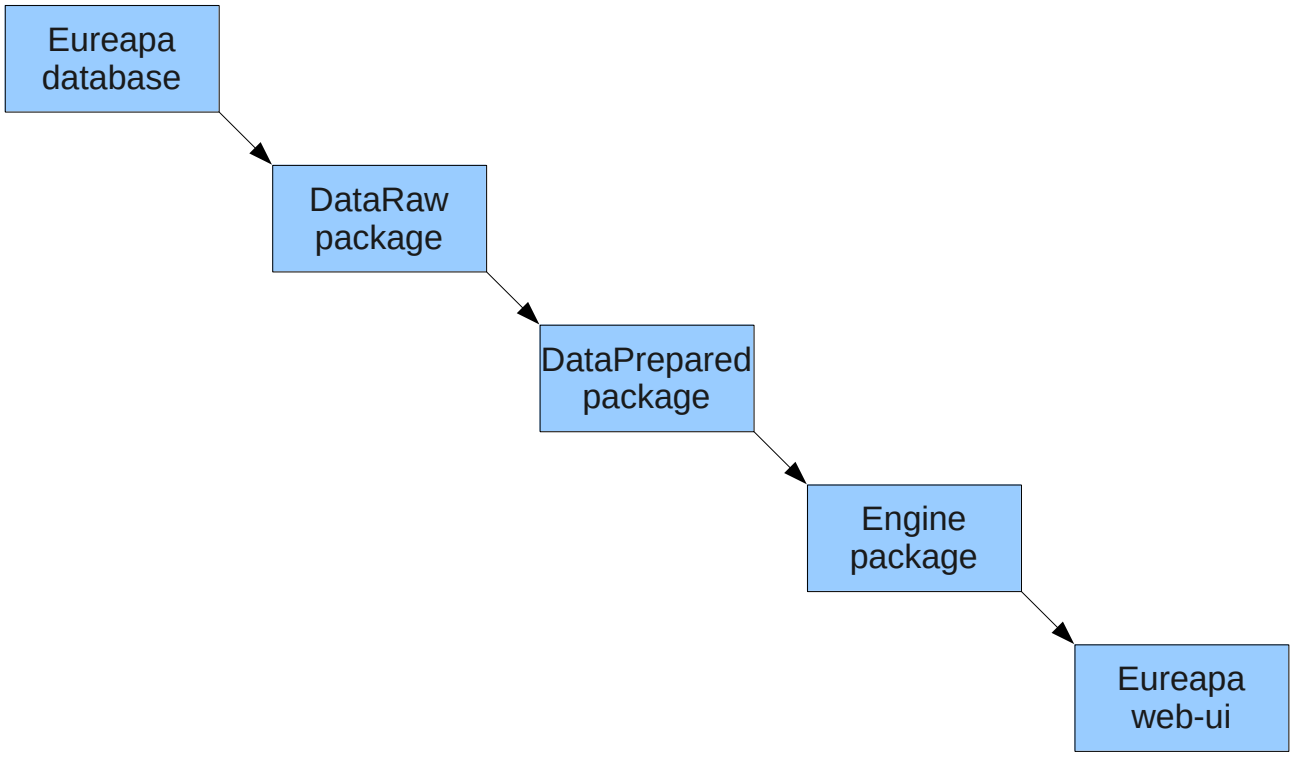


Illustration 2 shows the packages in the EUREAPA Engine application. The EUREAPA packages handle the data flow thus:

1. Database: stores data. We populate the EUREAPA Database (by running the EUREAPA Engine in database init mode) with aggregated data provided by the NTNU OPEN-EU model.
2. DataRaw: loads data from the database
3. DataPrepared: prepares baseline data, including initial (no-change) scenarios, from DataRaw
4. Engine: derives results from baseline prepared data and scenarios.
5. EUREAPA web-ui: web user interface. This is not described in this document.

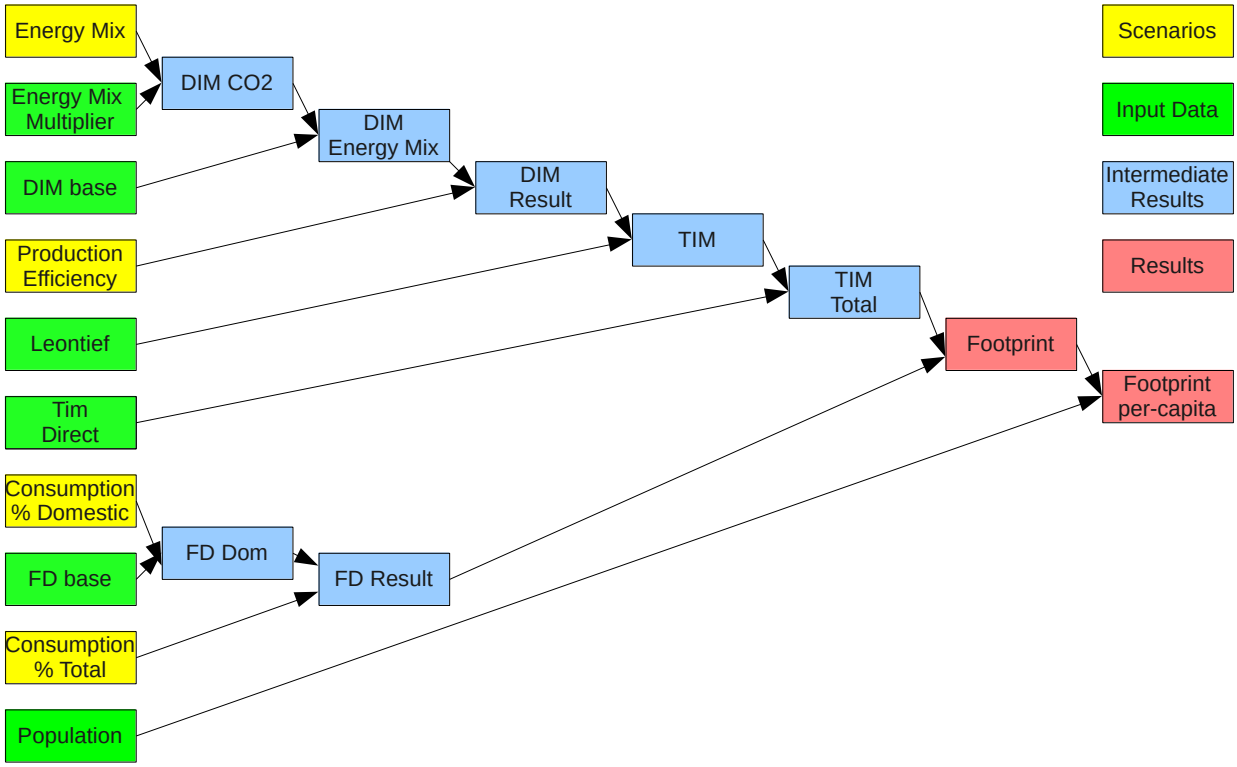
Each package:

- Gets Input-Output (IO) objects using getter methods in the previous package
- Constructs new IO objects
- Provides IO objects through getter methods to the next package

All these IO objects are immutable. They wrap a single data array property which is final and initialised at construction. The getter returns a copy so users of the copy cannot corrupt the original. The constructors perform the steps in the derivation chain. Often the constructor uses a helper static method to derive the new matrix which is then passed to the standard constructor.

We describe each of these packages in Sections below.

Illustration 3: EUREAPA Engine Data Flow



2. Engine package

This package derives footprint-per-capita results from baseline data and user-specified scenarios. Illustration 3 Shows the data derivation chain in the EUREAPA Engine package. The package API provides methods to set scenarios and get footprint results.

The Engine contains:

1. Constructor: which initialises the baseline IO properties by calling getters in DataPrepared
2. Methods
    1. API setter methods to set scenarios
    2. API getter methods to get results
    3. Run method to update results
3. Properties which hold the IO data objects

2.1 Constructor

The Engine constructor calls `run` to derive and initialise intermediate and baseline final results. This initial run uses the baseline scenario data which are no-change scenarios.

2.2 Properties

The IO data objects are organised in a class hierarchy:

- Io
  - Io2d
    - Dim
      - DimEnergyMix
      - DimProductionEfficiency
    - Fd
      - FdPropDomestic
      - FdPropTotal
    - EnergyMix
    - EnergyMixMultiplier

The engine initialises its baseline properties from DataPrepared, then derives baseline intermediate and final results using the derivation chain described below.

2.2.1 INTERMEDIATE RESULTS:

1. Dim-EnergyMix: derived by applying the energy mix scenario to DimBase
2. Dim-ProductionEfficiency: derived by applying the production-efficiency scenario to Dim EnergyMix
3. Fd-Domestic: derived by applying the Consumption Scenario (proportion Domestic) to the baseline FD. For 3 consumer types
4. Fd-Result: derived by applying the Consumption Scenario (proportion Domestic) to the Fd-Domestic. For 3 consumer types
5. TIM: derived from the DIM-ProductionEfficiency and the FD-Result
6. TIM-Total: the sum of TIM and Tim-direct: regionFd x region x product x impact

2.2.2 FINAL RESULTS:

1. Footprint: derived as TIM-Total * Fd-Result: regionFd x product x impact
2. Footprint-percapita: derived as the Footprint / Population: regionFd x product x impact

2.3 Methods

The Engine has:

1. API setter methods each update one of the 3 scenarios then invalidate (immediate) dependent (intermediate) results by nulling them:
   - setEnergy-mix: nulls dimEnergyMix
   - setProduction-efficiency nulls dimProductionEfficiency
   - setConsumption: nulls footprint
2. API result getter methods of 2 types:
   - API scenario getter methods return the current scenario data (energy-mix, productionEfficiency, consumption). Initially these are the no-change baseline scenarios. Thereafter they reflect setters' updates to the scenarios.
   - API results getter methods which call run to evaluate the derivation chain then return the footprint-per-capita for the given consumer type.
3. A single run method to update results. This :
   - Checks if dimEnergyMix is null, if so it calls deriveDimEnergyMix to update it
   - Checks if dimProductionEfficiency is null, if so it calls deriveDimProductionEfficiency to update it
   - Checks if footprint is null, if so it calls deriveFootprint to update it

2.4 API

Table 1 gives an overview of the Application Programmer Interface (API), for further details, see the Javadoc.

To run scenarios, set one or more of 3 scenario types either for all regions or for a specified Region of interest (RegionFd), eg UK:

- Consumption Scenario: also specify the consumer type: government, household, capital and VST
- Production Scenario
- Energy Mix Scenario

Scenario Type	Get Scenario		Set Scenario	
	all regions	**one region**	**all regions**	**one region**
Consumption	getConsumptionScenario4d	GetConsumptionScenario3d	SetConsumptionScenario4d	SetConsumptionScenario3d
Energy Mix	getEnergyMixScenario4d	getEnergyMixScenario3d	SetEnergyMixScenario4d	SetEnergyMixScenario3d
Production	getProductionScenario4d	getProductionScenario3d	SetProductionScenario4d	setProductionScenario3d

Table 1: Scenario Get and Set methods

3. DataPrepared

Illustration 4: Eureapa Engine Data Preparation

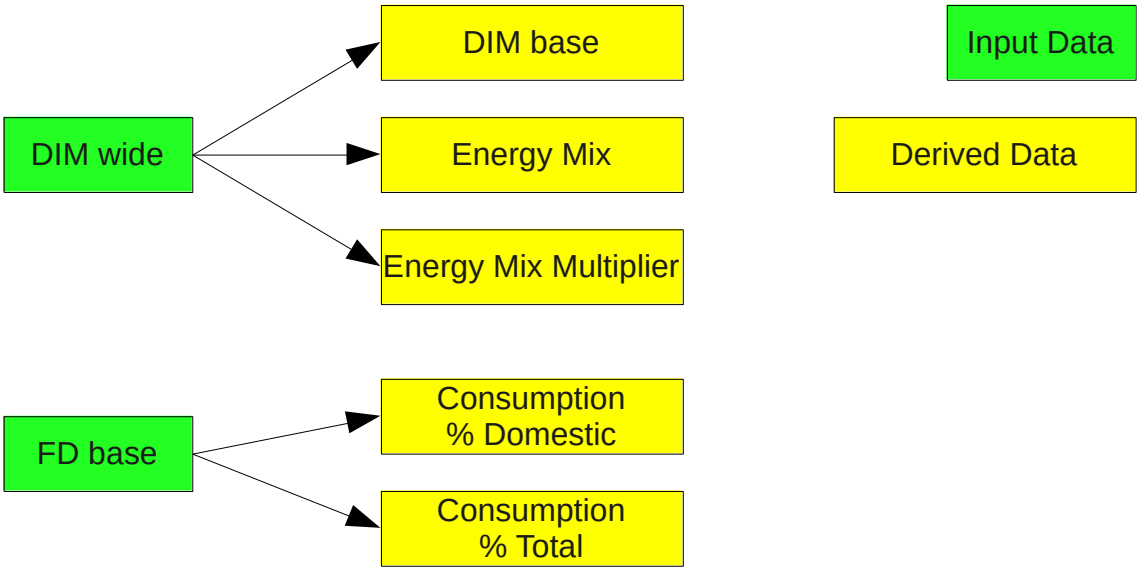


Illustration 4 shows the IO objects derived in the DataPrepared package.

The DataPrepared object prepares data for the engine. It contains:

- Constructor: initialises properties by getting data from Data Raw and preparing it
- Properties: Prepared data: matrices derived from the raw data, eg DimBase derived from DimWide
- Getter Methods provide data to the Engine

Also at this level are the CSV reader, CSV writer and Database initialiser.

Input data are prepared from the raw data, stored in properties in DataPrepated and exposed to the Engine by getters which provide copies of the data.

3.1 Properties

1. Consumption Scenario: 2 components: proportion of total and proportion domestic. Baseline values derived from Fd Base. For 3 consumer types

2. DimBase: region-sector(45x57) x impact(7): the first 7 columns of DimWide

3. Energy-Mix: region, sector,and energy carrier. Baseline values derived from the last 7 columns of dimWide divided by their row total

4. Energy Mix Multiplier (EMM): region, sector,and energy carrier. The DIM-co2 (middle 7) columns divided by the energy mix.

5. Production Efficiency: the relative production efficiency (compared to baseline) by 3 footprint types (Carbon, Ecological and Water) by sector and region. Baseline production efficiency set to ones.

4. DataRaw

DataRaw loads raw IO data from the database:

1. Methods: to load from database (using the Hibernate database persistence framework www.hibernate.org), getters to provide data to DataPrepared

2. Properties: Raw IO data loaded from the database

The raw input data are instantiated directly from the database by Hibernate in the loader. (For testing the loader is replaced by loaderMock which provides small sized test data.)

Raw Hibernate Entities loaded from the database: these contain baseline (2004) data as a single 2D array (except for DataSizes an Population).

Properties:

- DataSizes: contains int properties for the sizes of each dimension of the IO tables (n Sectors, n Regions etc.) and several indices and conversion factors

- DimWide: 2D array: region-sector(45x57) x {impact(7), DIM-co2(7), DIM-energy(7)

- FdBase: 2D array: regionFd(45) x region(45)product(62). For 3 consumer types

- Leontief: 2D array: region(45)product(62) x region(45)product(62)

- Population: of each region as a 1D int array

- TIM-direct: FD region, supply region, sector, impact. We use this baseline, unchanged.

- ZproductSector: a concordance matrix to transform Tim from 57 sectors to 62 products

5. Database

The EUREAPA database is currently MySQL. We initialise this from CSV files by running the EUREAPA application in *database initialisation* mode (see below).

6. Checks, asserts, tests

To provide a robust reliable results, the EUREAPA Application makes extensive use of checks, asserts and tests.

6.1 Checks and asserts

All IO object constructors and most methods check their arguments. Array argument sizes are checked against the expected sizes held as properties in the DataSizes object. A single generic method dataSizes.checkSize handles all the checking, throwing a meaningful Illegal Argument Exception on mismatch. To enable this most methods take a DataSizes argument. Methods which take external data, user input or database, always check their arguments. Other methods wrap the check in an assert which is only called in assertions are enabled (flag -ea to Java on command line).

6.2 Tests

EUREAPA is thoroughly unit tested. A full suite of tests is generated using the Netbeans tool (netbeans.org)"Create Unit Tests". The derivation methods are tested with simple values, sometimes using the JUnit parametrised tests. Higher level methods may be tested with zero filled arrays as a sanity check. A fixture class Fixture provides simple, small values for many test classes.

6.3 Vulnerabilities in Input Data

We prepare input data using Octave scripts. These write the data as headed csv files using the Octave function cell2csv (in the miscellaneous package). Cell2csv converts doubles with the C++ function:

```
sprintf(tmp, "%g", c(i, j).double_value());
```

This uses the format string "%g" for generic output format. The result is that precision is 6 and values > 1e6 are formatted in scientific notation (1.23456+E006). This is all fine.

However these scientific notation values in a CSV file are truncated to 2 decimal places when saved by Excel and Open Office Calc with considerable loss of precision.

To avoid this loss of precision, we regenerate the input data by running the Octave scripts to do so. Then immediately run the EUREAPA engine in the mode to populate the database.

Similarly the EUREAPA Engine emits values to CSV file using Double.toString(value) to convert doubles to Strings. Double.toString formats values > 1e7 and <1e-3 in scientific notation.

We briefly explored using other formatters, e.g. String.format("%.9f", value) but these lose precision on small and large numbers. See: http://download.oracle.com/javase/1.5.0/docs/api/java/util/Formatter.html

7. Installation

To install the engine link to the eureapa.jar file.

To initialise the database from CSV files, call main with argument - - init. (Set the MySQL server parameter: max_allowed_packet=64M to support transfer of the Leontief long blob which is ~ 50Mb).

LIBRARIES

The EUREAPA engine uses:

1. mysql-connector-java-5.1.13-bin.jar: in /usr/local/netbeans-7.0/ide/modules/ext
2. Netbeans Standard Libraries: Hibernate and Hibernate-JPA:
3. Uses AnnotationConfiguration rather than Configuration to create session factory.
4. Jama_1.0.2: Matrix support for RMatrix
5. RMatrix: an SEI REAP library for write to CSV files

PATHS

In normal run mode EUREAPA reads all data from the database.

In database initialisation mode (first command line argument is "--init"), EUREAPA sets EUREAPA_HOME, the default path to the parent directory containing CSV files (data files, test scenarios and results). The default is based on the OS reported by System.getProperty("os.name").toLowerCase() and containing: "linux", "mac", "win"

1. private static final File eureapaDataInputDefaultLinux = new File(<EUREAPA_HOME>"/eureapa/data/input");
2. private static final File eureapaDataInputDefaultMac = new File(<EUREAPA_HOME>"/eureapa/data/input");
3. private static final File eureapaDataInputDefaultWindows = new File(<EUREAPA_HOME>"\eureapa\\data\\input");

EUREAPA_HOME may be overridden by giving a path on the command line.

RUN

Flags:

- - - init: initialises the database from CSV files
- - - version reports the version
- arg staring with jdbc, db connecion url, e.g.: jdbc:mysql://144.32.110.17:3306/julian2
- remaining arg gives the path to EUREAPA_HOME where live the csv input files (data and scenario) and results

8. Usage

To run off baseline and scenario results, set values in the CSV files in EUREAPA_HOME/data/input then call main. EUREAPA emits footprint results, as CSV files, to the results directory.

To see intermediate results set the logging level (in logging.cfg) for Main to FINE. EUREAPA will then emit many of the intermediate results, as CSV files, to the results directory.

7th Framework Programme for Research and Technological Development.
The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement N° 227065.

Project Partners

One Planet Economy Network
C/o WWF-UK, Panda House
Weyside Park, Catteshall Lane
Godalming, Surrey GU7 1XR, UK

Tel: +44 (0)14 83 41 24 98
Email: info@oneplaneteconomynetwork.org
Web: oneplaneteconomynetwork.org