Quick Start Pymrio Tutorial using WIOD

This notebook contains the interactive version of the quick start given in the Pymrio article (Stadler et al 2018 sub).

Pymrio requires a Python version >= 3.7. If you don't have Python installed, I recommend to use the Anaconda Scientific Python package.

Pymrio is available on

and on

Thus, two possibilities exist to install Pymrio and all required packages.

For using the version on PyPI use:

```
pip install pymrio --upgrade
```

To install from the Anaconda Cloud do:

```
conda install -c conda-forge pymrio
```

Further down in that notebood we will also use the country_converter package as well as seaborn and matplotlib for some plotting. You can install these packages with pip or conda analog to pymrio. Alternatively, you can also run this notebook in the cloud via binder following this link:

You can than import the Pymrio package with

In [1]:
```
import pymrio
```

In this example here, we will use the WIOD MRIO database.

First, the Pymrio MRIO download function is used to get the WIOD MRIO database with:

In [2]:
```
raw_wiod_path = '/tmp/wiod/raw'
pymrio.download_wiod2013(storage_folder=raw_wiod_path,
                         years=[2008])
```

Out[2]:
```
Description: WIOD metadata file for pymrio
MRIO Name: WIOD
System: IxI
Version: data13
File: /tmp/wiod/raw/metadata.json
History:
20201120 14:24:19 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/water/wat_may12.zip to wat_may12.zip
20201120 14:24:19 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
```

```
a13/materials/mat_may12.zip to mat_may12.zip
20201120 14:24:18 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/land/lan_may12.zip to lan_may12.zip
20201120 14:24:17 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/AIR/AIR_may12.zip to AIR_may12.zip
20201120 14:24:17 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/CO2/CO2_may12.zip to CO2_may12.zip
20201120 14:24:16 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/EM/EM_may12.zip to EM_may12.zip
20201120 14:24:15 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/EU/EU_may12.zip to EU_may12.zip
20201120 14:24:14 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
a13/SEA/WIOD_SEA_July14.xlsx to WIOD_SEA_July14.xlsx
20201120 14:24:13 - FILEIO -  Downloaded http://www.wiod.org/protected3/dat
```

This downloads the 2008 MRIO table from WIOD. Omitting the year parameter would result getting all years. The function returns a Pymrio meta data object, which gives information about the WIOD version, system (in this case industry by industry) and records about from where the data was received.

To parse the database into a Pymrio object use:

In [3]:
```
wiod = pymrio.parse_wiod(raw_wiod_path, year=2008)
```

The available data can be explored by for example:

In [4]:
```
wiod.get_sectors()
```

Out[4]:
```
Index(['AtB', 'C', '15t16', '17t18', '19', '20', '21t22', '23', '24', '25',
       '26', '27t28', '29', '30t33', '34t35', '36t37', 'E', 'F', '50', '51
',
       '52', 'H', '60', '61', '62', '63', '64', 'J', '70', '71t74', 'L', 'M
',
       'N', 'O', 'P'],
      dtype='object', name='sector')
```

or

In [5]:
```
wiod.get_regions()
```

Out[5]:
```
Index(['AUS', 'AUT', 'BEL', 'BGR', 'BRA', 'CAN', 'CHN', 'CYP', 'CZE', 'DEU
',
       'DNK', 'ESP', 'EST', 'FIN', 'FRA', 'GBR', 'GRC', 'HUN', 'IDN', 'IND
',
       'IRL', 'ITA', 'JPN', 'KOR', 'LTU', 'LUX', 'LVA', 'MEX', 'MLT', 'NLD
',
       'POL', 'PRT', 'ROU', 'RUS', 'SVK', 'SVN', 'SWE', 'TUR', 'TWN', 'USA
',
       'RoW'],
      dtype='object', name='region')
```

In [6]:
```
wiod.Z
```

Out[6]:

	region							
	sector	AtB	C	15t16	17t18	19	20	
region	sector							
AUS	AtB	4445.324330	41.919400	15625.681890	536.968630	154.395870	936.835140	27
	C	16.277934	3838.070873	189.934275	11.313686	3.253063	14.271582	5

	region						
	sector	AtB	C	15t16	17t18	19	20
region	sector						
	15t16	1049.495726	100.611347	6754.110522	68.387761	19.663697	14.570366
	17t18	36.908420	43.779214	108.986668	355.675875	102.268333	18.335691
	19	9.518107	11.289978	28.105965	91.723266	26.373410	4.728489
...	...	...	...	...	...	...	...
RoW	L	0.547432	0.780406	1.176073	0.200903	0.057766	0.248589
	M	1.319036	9.927575	11.742183	2.639874	0.759049	1.003295
	N	7.894845	0.291041	11.603507	2.279403	0.655404	1.749095

WIOD includes several satellite accounts, which are stored as child objects in Pymrio. For example, in order to see the AIR emissions provided by WIOD:

In [7]:
```
wiod.AIR.F
```

Out[7]:

	region						
	sector	AtB	C	15t16	17t18	19	20
stressor							
CO2		6.471152e+03	2.331841e+04	3256.861259	392.819896	91.570641	147.075293
CH4		3.226169e+06	1.370016e+06	1221.450093	41.723574	6.112471	64.722688
N2O		6.527106e+04	1.243851e+02	527.652440	10.773378	1.335362	14.793543
NOX		2.000881e+05	1.709849e+05	70375.533177	3875.234721	964.709338	9146.373832
SOX		1.976645e+04	4.713841e+04	45815.675397	1068.354291	265.958435	2521.542160
CO		1.496859e+06	7.159254e+05	227663.413138	16225.875707	4039.304699	38296.499606
NMVOC		3.824729e+05	2.409498e+05	141642.740887	5460.933412	1359.456610	12888.958231
NH3		4.049434e+05	4.575323e+02	112.157985	4.313657	0.449874	13.342974

8 rows × 1435 columns

WIOD, however, does neither provide any normalized data (A-matrix, satellite account coefficient data) nor any consumption based accounts (footprints).

In order to calculate them, one could go through all the missing data and compute each account. Pymrio provides the required function, for example to calculate the A-matrix:

In [8]:
```
x = pymrio.calc_x(Z=wiod.Z, Y=wiod.Y)
A = pymrio.calc_A(Z=wiod.Z, x=x)
```

In [9]:
```
A.head()
```

Out[9]:

	region								
	sector	AtB	C	15t16	17t18	19	20	21t22	23
region	sector								
AUS	AtB	0.095452	0.000346	0.220811	0.086780	0.096757	0.093637	0.009559	0.000000

region									
	sector	AtB	C	15t16	17t18	19	20	21t22	23
region	sector								
	C	0.000350	0.031718	0.002684	0.001828	0.002039	0.001426	0.002035	0.220910
	15t16	0.022535	0.000831	0.095444	0.011052	0.012323	0.001456	0.001731	0.001811
	17t18	0.000793	0.000362	0.001540	0.057481	0.064090	0.001833	0.001757	0.000776

Alternatively, Pymrio provides a function which iterates through all missing accounts and calculates them:

In [10]:
```
wiod.calc_all()
```

Out[10]:
```
<pymrio.core.mriosystem.IOSystem at 0x7fb60b1900d0>
```

At this point, a basic EE MRIO analysis is accomplished. For example, the regional consumption based accounts of the AIR emissions are now given by:

In [11]:
```
wiod.AIR.D_cba_reg
```

Out[11]:

region	AUS	AUT	BEL	BGR	BRA	CAN
stressor						
CO2	4.404070e+05	1.022100e+05	1.586176e+05	42924.986975	4.059629e+05	5.659664e+05
CH4	4.275465e+06	7.599975e+05	1.030354e+06	464018.748607	1.352464e+07	4.068558e+06
N2O	9.588178e+04	3.086814e+04	4.609171e+04	13203.713081	5.899229e+05	1.634371e+05
NOX	2.359815e+06	3.324339e+05	4.508892e+05	142917.818720	2.786076e+06	1.904551e+06
SOX	2.399335e+06	1.983047e+05	3.702525e+05	400357.951750	1.699074e+06	2.088103e+06
CO	2.173900e+07	1.371366e+06	2.167114e+06	703172.284772	2.681292e+07	7.525147e+06
NMVOC	3.101630e+06	3.582680e+05	5.920832e+05	190582.650539	5.323333e+06	2.131757e+06
NH3	3.851776e+05	9.254548e+04	1.245648e+05	45897.394639	1.345046e+06	4.204562e+05

8 rows × 41 columns

In [12]:
```
wiod.AIR.unit
```

Out[12]:

unit	unit
stressor	
CO2	Gg
CH4	t
N2O	t
NOX	Unnamed: 0
SOX	Unnamed: 0
CO	t
NMVOC	t
NH3	t

Pymrio can be linked with the country converter coco to ease the aggregation of MRIO and results into different classifications. Using the country converter, WIOD can be aggregated into EU and non-EU countries with singling out Germany by:

In [13]:
```
import country_converter as coco
wiod.aggregate(region_agg = coco.agg_conc(original_countries='WIOD',
                                          aggregates=[{'DEU': 'DEU', 'GBR'
                                          missing_countries='Other',
                                          merge_multiple_string=None))
```

Out[13]:
```
<pymrio.core.mriosystem.IOSystem at 0x7fb60b1900d0>
```

We rename the EU account to reflect that is does not include Germany:

In [14]:
```
wiod.rename_regions({'EU':'Rest of EU'})
```

Out[14]:
```
<pymrio.core.mriosystem.IOSystem at 0x7fb60b1900d0>
```

The regional footprint account are now:

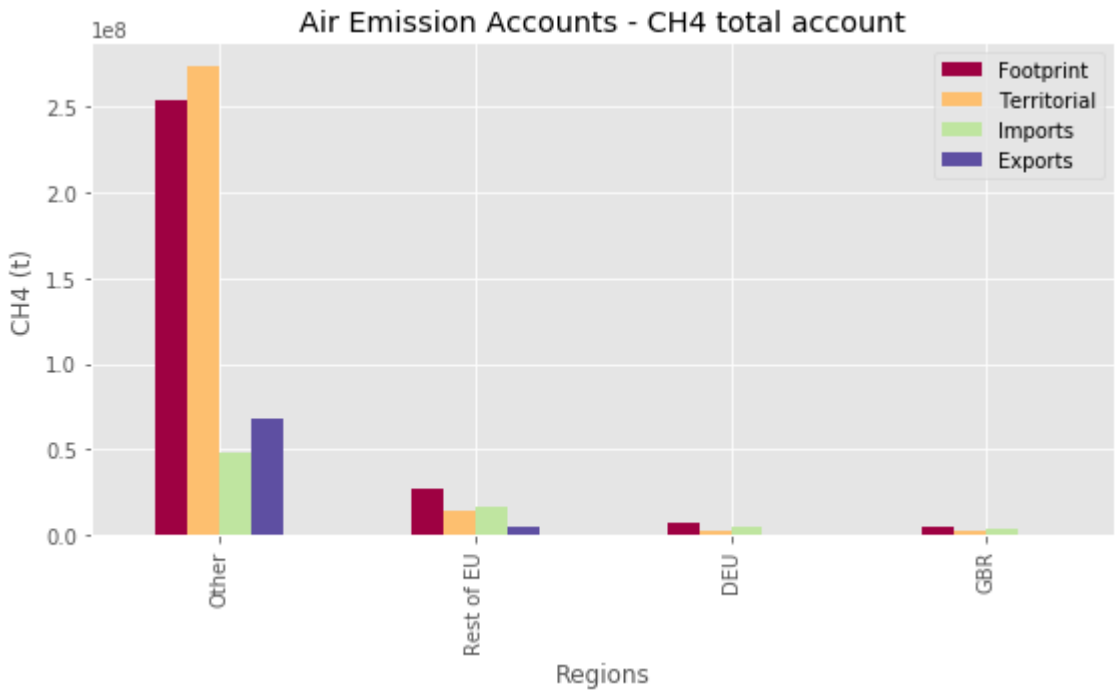
In [15]:
```
wiod.AIR.D_cba_reg
```

Out[15]:

region	Other	Rest of EU	DEU	GBR
stressor				
CO2	2.436179e+07	3.472823e+06	1.054136e+06	7.397044e+05
CH4	2.540661e+08	2.711250e+07	6.668537e+06	5.235498e+06
N2O	9.705186e+06	1.128531e+06	2.914646e+05	2.118832e+05
NOX	1.043111e+08	1.093267e+07	2.701648e+06	2.164933e+06
SOX	1.037493e+08	8.344435e+06	1.951840e+06	1.421854e+06
CO	7.661455e+08	5.466639e+07	1.099191e+07	1.068169e+07
NMVOC	1.280392e+08	1.577316e+07	2.923060e+06	2.986943e+06
NH3	2.672782e+07	3.493227e+06	8.505438e+05	5.900984e+05

To visualize for example the CH4 accounts:

In [16]:
```
import matplotlib.pyplot as plt
with plt.style.context('ggplot'):
    wiod.AIR.plot_account('CH4', figsize=(8,5))
    plt.savefig('/tmp/wiod/airch4.png', dpi=300)
    plt.show()
```



To calculate the source (in terms of regions and sectors) of a certain stressor or impact driven by consumption, one needs to diagonalize this stressor/impact. This can be done with Pymrio by:

In [17]:
```
diag_CH4 = wiod.AIR.diag_stressor('CH4')
```

and be reassigned to the aggregated WIOD system:

In [18]:
```
wiod.CH4_source = diag_CH4
```

In the next step the automatic calculation routine of Pymrio is called again to compute the missing accounts in this new extension: and be reassigned to the aggregated WIOD system:

In [19]:
```
wiod.calc_all()
```

Out[19]:
```
<pymrio.core.mriosystem.IOSystem at 0x7fb60b1900d0>
```

The diagonalized CH4 data now shows the source and destination of the specified stressor (CH4):

In [20]:
```
wiod.CH4_source.D_cba.head()
```

Out[20]:

region		region					
		sector	AtB	C	15t16	17t18	19
region	sector						
Other	AtB	6.120041e+07	8.455234e+04	3.658411e+07	2.988418e+06	867172.684194	230646
	C	1.008359e+06	6.714292e+06	2.047332e+06	6.420307e+05	117665.005135	33160
	15t16	3.968202e+03	6.218228e+01	8.736127e+04	5.178453e+02	639.802755	24
	17t18	9.369869e+01	2.287802e+01	2.464623e+02	1.185856e+04	200.126039	4
	19	5.587156e+00	1.040420e+00	1.268140e+01	1.252185e+02	1231.987687	0

5 rows × 140 columns

In this square footprint matrix, every column represents the amount of stressor occurring in each region - sector driven by the consumption stated in the column header. Conversly, each row states where the stressor impacts occurring in the row are distributed due (from where they are driven).

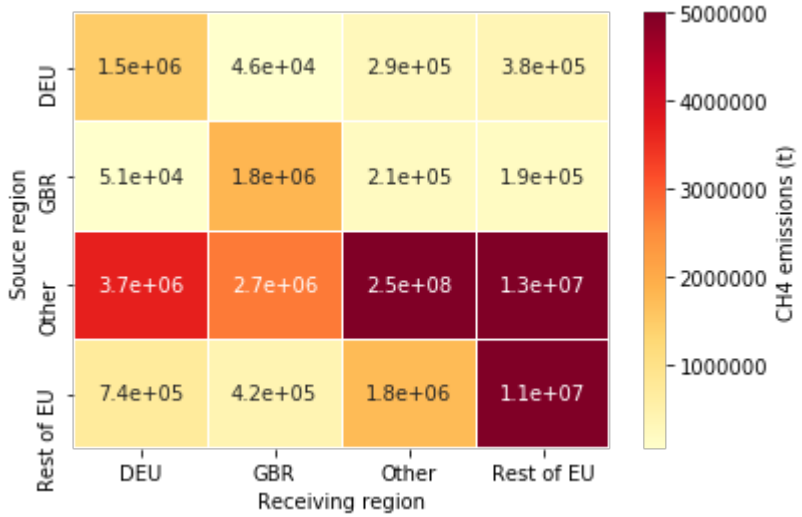
In [21]:
```
CH4_source_reg = wiod.CH4_source.D_cba.groupby(
    level='region', axis=0).sum().groupby(
    level='region', axis=1).sum()
```

In [22]:
```
CH4_source_reg
```

Out[22]:

region	DEU	GBR	Other	Rest of EU
region				
DEU	1.485343e+06	4.634238e+04	2.892830e+05	3.819713e+05
GBR	5.139252e+04	1.833541e+06	2.112405e+05	1.879226e+05
Other	3.696832e+06	2.711860e+06	2.457410e+08	1.317725e+07
Rest of EU	7.402886e+05	4.186665e+05	1.756700e+06	1.128755e+07

In [23]:
```
import seaborn as sns
CH4_source_reg.columns.name = 'Receiving region'
CH4_source_reg.index.name = 'Souce region'
sns.heatmap(CH4_source_reg, vmax=5E6,
            annot=True, cmap='YlOrRd', linewidths=0.1,
            cbar_kws={'label': 'CH4 emissions ({})'.format(wiod.CH4_source
plt.savefig('/tmp/wiod/airch4_source_reg.png', dpi=300)
plt.show()
```



Storing the MRIO database can be done with

In [24]:
```
storage_path = '/tmp/wiod/aly'
wiod.save_all(storage_path)
```

Out[24]:
```
<pymrio.core.mriosystem.IOSystem at 0x7fb60b1900d0>
```

From where it can be received subsequently by:

In [25]:
```
wiod = pymrio.load_all(storage_path)
```

The meta attribute of Pymrio mentioned at the beginning kept track of all modifications of the

system. This can be shown with:

In [26]:
```
wiod.meta
```

Out[26]:
```
Description: WIOD metadata file for pymrio
MRIO Name: WIOD
System: industry-by-industry
Version: data13
File: /tmp/wiod/aly/metadata.json
History:
20201125 14:17:05 - FILEIO -  Added satellite account from /tmp/wiod/aly/fa
ctor_inputs
20201125 14:17:05 - FILEIO -  Added satellite account from /tmp/wiod/aly/SE
A
20201125 14:17:05 - FILEIO -  Added satellite account from /tmp/wiod/aly/AI
R
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/CO
2
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/EM
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/EU
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/la
n
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/ma
t
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/wa
t
20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/CH
4_source
 ... (more lines in history)
```

Custom notes can be added to the meta with:

In [27]:
```
wiod.meta.note("Custom note")
```

The history of the meta data can be filtered for specific entries like:

In [28]:
```
wiod.meta.file_io_history
```

Out[28]:
```
['20201125 14:17:05 - FILEIO -  Added satellite account from /tmp/wiod/aly/
factor_inputs',
 '20201125 14:17:05 - FILEIO -  Added satellite account from /tmp/wiod/aly/
SEA',
 '20201125 14:17:05 - FILEIO -  Added satellite account from /tmp/wiod/aly/
AIR',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
CO2',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
EM',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
EU',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
lan',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
mat',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
wat',
 '20201125 14:17:04 - FILEIO -  Added satellite account from /tmp/wiod/aly/
CH4_source',
 '20201125 14:17:04 - FILEIO -  Loaded IO system from /tmp/wiod/aly',
 '20201125 14:17:03 - FILEIO -  Saved WIOD to /tmp/wiod/aly',
 '20201125 14:17:00 - FILEIO -  Extension wat parsed from /tmp/wiod/raw',
 '20201125 14:16:59 - FILEIO -  Extension mat parsed from /tmp/wiod/raw',
 '20201125 14:16:58 - FILEIO -  Extension lan parsed from /tmp/wiod/raw',
```

```
 '20201125 14:16:57 - FILEIO -  Extension EU parsed from /tmp/wiod/raw',
 '20201125 14:16:55 - FILEIO -  Extension EM parsed from /tmp/wiod/raw',
 '20201125 14:16:53 - FILEIO -  Extension CO2 parsed from /tmp/wiod/raw',
 '20201125 14:16:51 - FILEIO -  Extension AIR parsed from /tmp/wiod/raw',
 '20201125 14:16:50 - FILEIO -  SEA file extension parsed from /tmp/wiod/ra
w',
 '20201125 14:16:39 - FILEIO -  WIOD data parsed from /tmp/wiod/raw/wiot08_
row_sep12.xlsx',
 '20201120 14:24:19 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/water/wat_may12.zip to wat_may12.zip',
 '20201120 14:24:19 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/materials/mat_may12.zip to mat_may12.zip',
 '20201120 14:24:18 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/land/lan_may12.zip to lan_may12.zip',
 '20201120 14:24:17 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/AIR/AIR_may12.zip to AIR_may12.zip',
 '20201120 14:24:17 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/CO2/CO2_may12.zip to CO2_may12.zip',
 '20201120 14:24:16 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/EM/EM_may12.zip to EM_may12.zip',
 '20201120 14:24:15 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/EU/EU_may12.zip to EU_may12.zip',
 '20201120 14:24:14 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
ata13/SEA/WIOD_SEA_July14.xlsx to WIOD_SEA_July14.xlsx',
 '20201120 14:24:13 - FILEIO -  Downloaded http://www.wiod.org/protected3/d
```

This tutorial gave a short overview about the basic functionality of Pymrio. For more information about the capabilities of pymrio check the online documentation.