Achieving the Net-Zero Emission Target: A meta-analysis of Turkish Energy and Emission Scenarios

By Görkem Güngör and Latife Demirtaş

June 1, 2023

1 Meta-analysis of Turkish Energy and Climate Pathways

1.1 Scope and feature overview

The **Türkiye National Energy Plan** (TUEP) modeling horizon is 2035 based on the net-zero target in 2053.

The **pyam** package is used for analyzing, visualizing and working with timeseries data following the format established by the *Integrated Assessment Modeling Consortium* (IAMC); read the docs for more information.

1.2 Highlights

The main themes for the **Türkiye National Energy Plan** and the **Türkiye Hydrogen Strategy and Road-Map** modeling horizon 2035 are:

- Final renewable energy includes solar, biomass and geothermal
- Hydrogen and synthetic methane are clean fuels
- Hydrogen is produced in the electrolyser, whereas DAC using CCS is optional for producing synthetic methane after 2035
- Final natural gas is blended by 3.5% with hydrogen for final sectoral demand after 2035
- Secondary renewable electricity includes solar, wind, hydro, biomass and geothermal
- Although the emissions are not specified, the plan is based on the net-zero carbon emission target for 2053
- Battery storage has 2 hours charging period.

1.3 Capacity projections

Installed capacity	unit	2030	2035	2055
Solar power	GW		52.9 (59.71)	
Wind power	GW		29.6 (50.11)	
Nuclear power	GW		7.2 (4.81)	
New installed capacity	GW		96.9	
Total installed capacity	GW		189.7 (202.11)	
Battery storage	GW		7.5	
Electrolyser	GW	1.9	5.0	70.0
Demand side management	GW	0.9	1.7	

1 Capacity projections of Istanbul Policy Center for Net-Zero Scenario

1.4 Data

The timeseries data used in this notebook are manually assembled from official reports. The main official report is the *Türkiye National Energy Plan* (TUEP) of the Ministry of Energy and Natural Resources.

1.4.1 Scenarios in the data

The scenarios included in the official reports are:

- Energy Security Scenario from the Ministry of Energy and Natural Resources (2023) *Türkiye National Energy Plan*
- Baseline and Net-Zero Scenarios from Istanbul Policy Center (2021) *Turkey's Decarbonization Pathway*
- Baseline, Optimistic and Pessimistic Scenarios from TÜBİTAK-MAM (2012) *Mitigation / Adaptation scenarios and Climate Change policy portfolios for Turkey*

This notebook is intended for meta-analysis of Turkish energy and climate pathways from the literature.

---

[1]:
```
import numpy as np
import pyam
import matplotlib.pyplot as plt
```

```
<IPython.core.display.Javascript object>
```

1.5 Import data from file and inspect the scenario

We import the snapshot of the timeseries data from the file `data.csv`.

If you haven't cloned the GitHub repository to your machine, you can download the file from GitHub data.
Make sure to place the file in the same folder as this notebook.

[2]:
```python
df = pyam.IamDataFrame(data='data.csv')
```

```
pyam - INFO: Running in a notebook, setting up a basic logging at level INFO
pyam.core - INFO: Reading file data.csv
```

As a first step, we show an overview of the **IamDataFrame** content by simply calling `df` (alternatively, you can use `print(df)` or df.info()).

This function returns a concise (abbreviated) overview of the index dimensions and the qualitative/quantitative meta indicators (see an explanation of indicators below).

[3]:
```python
df
```

[3]:
```
<class 'pyam.core.IamDataFrame'>
Index:
 * model    : Gungor (2020), IPC (2021), MENR (2006), MENR (2023), TUBITAK
(2012) (5)
 * scenario : Baseline Scenario, CO2 Scenario, ... SSP3-RCP3.4-FIT (15)
Timeseries data coordinates:
   region   : Turkey (1)
   variable : Emissions|CO2, Final Energy, ... Secondary Energy|Electricity|Wind
(47)
   unit     : MW, Mt CO2/yr, Mtoe/yr, TWh/yr (4)
   year     : 2010, 2020, 2030, 2040, 2050, 2055, 2070 (7)
   type     : CGE, Linear Programming, Market Based Simulation, Regression
Analysis (4)
Meta indicators:
   exclude (bool) False (1)
```

In the following cells, we display the lists of all models, scenarios, regions, and the mapping of variables to units in the snapshot.

[4]:
```python
df.model
```

[4]:
```
['Gungor (2020)', 'IPC (2021)', 'MENR (2006)', 'MENR (2023)', 'TUBITAK (2012)']
```

[5]:
```python
df.scenario
```

[5]:
```
['Baseline Scenario',
 'CO2 Scenario',
 'Cogeneration Scenario',
 'Demand Efficiency Scenario',
 'Low Demand Scenario',
 'Net-Zero Scenario',
 'No-Nuclear Scenario',
 'Optimistic Scenario',
 'Pessimistic Scenario',
 'SSP1-Baseline-FIT',
 'SSP1-RCP2.6-FIT',
```

```
 'SSP2-Baseline-FIT',
 'SSP2-RCP2.6-FIT',
 'SSP3-Baseline-FIT',
 'SSP3-RCP3.4-FIT']
```

[6]:
```
df.region
```

[6]:
```
['Turkey']
```

[7]:
```
df.unit_mapping
```

[7]:
```
{'Emissions|CO2': 'Mt CO2/yr',
 'Final Energy': 'Mtoe/yr',
 'Final Energy|Agriculture': 'Mtoe/yr',
 'Final Energy|Agriculture|Electricity': 'TWh/yr',
 'Final Energy|Commercial': 'Mtoe/yr',
 'Final Energy|Electricity': ['TWh/yr', 'Mtoe/yr'],
 'Final Energy|Gases': 'Mtoe/yr',
 'Final Energy|Heat': 'Mtoe/yr',
 'Final Energy|Hydrogen': 'TWh/yr',
 'Final Energy|Industry': 'Mtoe/yr',
 'Final Energy|Industry|Electricity': 'TWh/yr',
 'Final Energy|Liquids': 'Mtoe/yr',
 'Final Energy|Non-Energy': 'Mtoe/yr',
 'Final Energy|Other': 'Mtoe/yr',
 'Final Energy|Renewables': 'Mtoe/yr',
 'Final Energy|Residential': 'Mtoe/yr',
 'Final Energy|Residential|Electricity': 'TWh/yr',
 'Final Energy|Services|Electricity': 'TWh/yr',
 'Final Energy|Solids': 'Mtoe/yr',
 'Final Energy|Transportation': 'Mtoe/yr',
 'Final Energy|Transportation|Electricity': 'TWh/yr',
 'Primary Energy': ['MW', 'Mtoe/yr'],
 'Primary Energy|Biomass': 'Mtoe/yr',
 'Primary Energy|Coal': 'Mtoe/yr',
 'Primary Energy|Gas': 'Mtoe/yr',
 'Primary Energy|Geothermal|Electricity': 'Mtoe/yr',
 'Primary Energy|Geothermal|Heat': 'Mtoe/yr',
 'Primary Energy|Hydro': 'Mtoe/yr',
 'Primary Energy|Nuclear': 'Mtoe/yr',
 'Primary Energy|Oil': 'Mtoe/yr',
 'Primary Energy|Renewables': 'Mtoe/yr',
 'Primary Energy|Solar': 'Mtoe/yr',
 'Primary Energy|Wind': 'Mtoe/yr',
 'Secondary Energy|Electricity': 'TWh/yr',
 'Secondary Energy|Electricity|Coal': 'TWh/yr',
 'Secondary Energy|Electricity|Fossil': 'TWh/yr',
```

```
 'Secondary Energy|Electricity|Gas': 'TWh/yr',
 'Secondary Energy|Electricity|Gases': 'TWh/yr',
 'Secondary Energy|Electricity|Hydro': 'TWh/yr',
 'Secondary Energy|Electricity|Nuclear': 'TWh/yr',
 'Secondary Energy|Electricity|Oil': 'TWh/yr',
 'Secondary Energy|Electricity|Other': 'TWh/yr',
 'Secondary Energy|Electricity|Renewables': 'TWh/yr',
 'Secondary Energy|Electricity|Renewables|Solar': 'TWh/yr',
 'Secondary Energy|Electricity|Renewables|Wind': 'TWh/yr',
 'Secondary Energy|Electricity|Solar': 'TWh/yr',
 'Secondary Energy|Electricity|Wind': 'TWh/yr'}
```

We convert the units **Mtoe/yr** and **TWh/yr** to **EJ/yr** compliant with the IAMC template.

[8]:
```
df.convert_unit('Mtoe/yr', to='EJ/yr', inplace=True)
df.convert_unit('TWh/yr', to='EJ/yr', inplace=True)
df.convert_unit('MW', to='EJ/yr', inplace=True)
```

[9]:
```
df.unit_mapping
```

[9]:
```
{'Emissions|CO2': 'Mt CO2/yr',
 'Final Energy': 'EJ/yr',
 'Final Energy|Agriculture': 'EJ/yr',
 'Final Energy|Agriculture|Electricity': 'EJ/yr',
 'Final Energy|Commercial': 'EJ/yr',
 'Final Energy|Electricity': 'EJ/yr',
 'Final Energy|Gases': 'EJ/yr',
 'Final Energy|Heat': 'EJ/yr',
 'Final Energy|Hydrogen': 'EJ/yr',
 'Final Energy|Industry': 'EJ/yr',
 'Final Energy|Industry|Electricity': 'EJ/yr',
 'Final Energy|Liquids': 'EJ/yr',
 'Final Energy|Non-Energy': 'EJ/yr',
 'Final Energy|Other': 'EJ/yr',
 'Final Energy|Renewables': 'EJ/yr',
 'Final Energy|Residential': 'EJ/yr',
 'Final Energy|Residential|Electricity': 'EJ/yr',
 'Final Energy|Services|Electricity': 'EJ/yr',
 'Final Energy|Solids': 'EJ/yr',
 'Final Energy|Transportation': 'EJ/yr',
 'Final Energy|Transportation|Electricity': 'EJ/yr',
 'Primary Energy': 'EJ/yr',
 'Primary Energy|Biomass': 'EJ/yr',
 'Primary Energy|Coal': 'EJ/yr',
 'Primary Energy|Gas': 'EJ/yr',
 'Primary Energy|Geothermal|Electricity': 'EJ/yr',
 'Primary Energy|Geothermal|Heat': 'EJ/yr',
 'Primary Energy|Hydro': 'EJ/yr',
```

```
 'Primary Energy|Nuclear': 'EJ/yr',
 'Primary Energy|Oil': 'EJ/yr',
 'Primary Energy|Renewables': 'EJ/yr',
 'Primary Energy|Solar': 'EJ/yr',
 'Primary Energy|Wind': 'EJ/yr',
 'Secondary Energy|Electricity': 'EJ/yr',
 'Secondary Energy|Electricity|Coal': 'EJ/yr',
 'Secondary Energy|Electricity|Fossil': 'EJ/yr',
 'Secondary Energy|Electricity|Gas': 'EJ/yr',
 'Secondary Energy|Electricity|Gases': 'EJ/yr',
 'Secondary Energy|Electricity|Hydro': 'EJ/yr',
 'Secondary Energy|Electricity|Nuclear': 'EJ/yr',
 'Secondary Energy|Electricity|Oil': 'EJ/yr',
 'Secondary Energy|Electricity|Other': 'EJ/yr',
 'Secondary Energy|Electricity|Renewables': 'EJ/yr',
 'Secondary Energy|Electricity|Renewables|Solar': 'EJ/yr',
 'Secondary Energy|Electricity|Renewables|Wind': 'EJ/yr',
 'Secondary Energy|Electricity|Solar': 'EJ/yr',
 'Secondary Energy|Electricity|Wind': 'EJ/yr'}
```

1.6 Apply filters to the ensemble and display the timeseries data

A selection of the timeseries data of an **IamDataFrame** can be obtained by applying the filter() function, which takes keyword-arguments of criteria. The function returns a down-selected clone of the **IamDataFrame** instance.

1.6.1 Filtering by model names, scenarios and regions

The feature for filtering by **model, scenario or region** are implemented using exact string matching, where `*` can be used as a wildcard.

First, we want to display the list of all scenarios in TUEP.

> Applying the filter argument `model='MENR'` will return an empty array
> (because the model in the data is actually called **MENR (2023)**)

[10]:
```
df.filter(model='MENR').scenario
```

```
pyam.core - WARNING: Filtered IamDataFrame is empty!
```

[10]:
```
[]
```

Filtering for `model='MENR*'` will return all scenarios provided by the **Ministry of Energy and Natural Resources**.

[11]:
```
df.filter(model='MENR*').scenario
```

[11]:
```
['Baseline Scenario',
 'Cogeneration Scenario',
 'Demand Efficiency Scenario',
```

```
 'Low Demand Scenario',
 'No-Nuclear Scenario',
 'CO2 Scenario']
```

1.6.2 Inverting the selection

Using the keyword `keep=False` allows you to select the inverse of the filter arguments. We can see that our data only contains information for region *Turkey*.

[12]:
```
df.filter(region='Turkey').region
```

[12]:
```
['Turkey']
```

[13]:
```
df.filter(region='Turkey', keep=False).region
```

```
pyam.core - WARNING: Filtered IamDataFrame is empty!
```

[13]:
```
[]
```

1.6.3 Filtering by variables and levels

Filtering for **variable** strings works in an identical way as above, with `*` available as a wildcard.

> Filtering for `Primary Energy` will return only exactly those data
>
> Filtering for `Primary Energy|*` will return all sub-categories of primary energy (and only the sub-categories)

In addition, variables can be filtered by their **level**, i.e., the "depth" of the variable in a hierarchical reading of the string separated by `|` (*pipe*, not L or i). That is, the variable `Primary Energy` has level 0, while `Primary Energy|Coal` has level 1.

Filtering by both **variables** and **level** will search for the hierarchical depth *following the variable string* so filter arguments `variable='Primary Energy*'` and `level=1` will return all variables immediately below `Primary Energy`. Filtering by **level** only will return all variables at that depth.

[14]:
```
df.filter(variable='Primary Energy*', level=1).variable
```

[14]:
```
['Primary Energy|Biomass',
 'Primary Energy|Coal',
 'Primary Energy|Gas',
 'Primary Energy|Hydro',
 'Primary Energy|Nuclear',
 'Primary Energy|Oil',
 'Primary Energy|Solar',
 'Primary Energy|Wind',
 'Primary Energy|Renewables']
```

The next cell illustrates another use case of the **level** filter argument - filtering by `1-` (as string) instead of `1` (as integer) will return all timeseries data for variables *up to* the specified depth.

[15]:
```
df.filter(variable='Primary Energy*', level='1-').variable
```

[15]:
```
['Primary Energy',
 'Primary Energy|Biomass',
 'Primary Energy|Coal',
 'Primary Energy|Gas',
 'Primary Energy|Hydro',
 'Primary Energy|Nuclear',
 'Primary Energy|Oil',
 'Primary Energy|Solar',
 'Primary Energy|Wind',
 'Primary Energy|Renewables']
```

The last cell shows how to filter only by **level** without providing a **variable** argument. The example returns all variables that are at the second hierarchical level (i.e., not `Primary Energy`).

[16]:
```
df.filter(level=1).variable
```

[16]:
```
['Emissions|CO2',
 'Final Energy|Electricity',
 'Final Energy|Hydrogen',
 'Final Energy|Agriculture',
 'Final Energy|Gases',
 'Final Energy|Industry',
 'Final Energy|Liquids',
 'Final Energy|Non-Energy',
 'Final Energy|Other',
 'Final Energy|Renewables',
 'Final Energy|Residential',
 'Final Energy|Solids',
 'Final Energy|Transportation',
 'Primary Energy|Biomass',
 'Primary Energy|Coal',
 'Primary Energy|Gas',
 'Primary Energy|Hydro',
 'Primary Energy|Nuclear',
 'Primary Energy|Oil',
 'Primary Energy|Solar',
 'Primary Energy|Wind',
 'Secondary Energy|Electricity',
 'Final Energy|Commercial',
 'Final Energy|Heat',
 'Primary Energy|Renewables']
```

1.6.4 Displaying timeseries data

As a next step, we want to view a selection of the timeseries data.

The timeseries() function returns the data as a pandas.DataFrame in the standard IAMC template.

This is a **wide format** table where years are shown as columns.

[17]:
```python
display_df = df.filter(model='MENR*', variable='Primary Energy*', level=1,
    region='Turkey')
display_df.timeseries()
```

[17]:
```
                    2010  \
model       scenario          region variable                   unit  type
MENR (2006) Baseline Scenario Turkey Primary Energy|Biomass    EJ/yr Market
Based Simulation  0.167472
                                     Primary Energy|Coal       EJ/yr Market
Based Simulation  0.975524
                                     Primary Energy|Gas        EJ/yr Market
Based Simulation  0.008374
                                     Primary Energy|Hydro      EJ/yr Market
Based Simulation  0.209340
                                     Primary Energy|Nuclear    EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Oil        EJ/yr Market
Based Simulation  0.083736
                                     Primary Energy|Solar      EJ/yr Market
Based Simulation  0.020934
                                     Primary Energy|Wind       EJ/yr Market
Based Simulation  0.016747
MENR (2023) CO2 Scenario      Turkey Primary Energy|Coal       EJ/yr Linear
Programming            NaN
                                     Primary Energy|Gas        EJ/yr Linear
Programming            NaN
                                     Primary Energy|Nuclear    EJ/yr Linear
Programming            NaN
                                     Primary Energy|Oil        EJ/yr Linear
Programming            NaN
                                     Primary Energy|Renewables EJ/yr Linear
Programming            NaN

                    2020  \
model       scenario          region variable                   unit  type
MENR (2006) Baseline Scenario Turkey Primary Energy|Biomass    EJ/yr Market
Based Simulation  0.167472
                                     Primary Energy|Coal       EJ/yr Market
Based Simulation  1.561676
                                     Primary Energy|Gas        EJ/yr Market
Based Simulation  0.008374
                                     Primary Energy|Hydro      EJ/yr Market
Based Simulation  0.376812
                                     Primary Energy|Nuclear    EJ/yr Market
Based Simulation  0.334944
```

```
                                     Primary Energy|Oil        EJ/yr Market
Based Simulation  0.041868
                                     Primary Energy|Solar      EJ/yr Market
Based Simulation  0.041868
                                     Primary Energy|Wind       EJ/yr Market
Based Simulation  0.041868
MENR (2023) CO2 Scenario      Turkey Primary Energy|Coal       EJ/yr Linear
Programming       1.699841
                                     Primary Energy|Gas        EJ/yr Linear
Programming       1.666346
                                     Primary Energy|Nuclear    EJ/yr Linear
Programming            NaN
                                     Primary Energy|Oil        EJ/yr Linear
Programming       1.766830
                                     Primary Energy|Renewables EJ/yr Linear
Programming       1.029953

                  2030  \
model       scenario          region variable                  unit  type
MENR (2006) Baseline Scenario Turkey Primary Energy|Biomass    EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Coal       EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Gas        EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Hydro      EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Nuclear    EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Oil        EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Solar      EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Wind       EJ/yr Market
Based Simulation       NaN
MENR (2023) CO2 Scenario      Turkey Primary Energy|Coal       EJ/yr Linear
Programming       2.005477
                                     Primary Energy|Gas        EJ/yr Linear
Programming       1.997104
                                     Primary Energy|Nuclear    EJ/yr Linear
Programming       0.334944
                                     Primary Energy|Oil        EJ/yr Linear
Programming       2.294366
                                     Primary Energy|Renewables EJ/yr Linear
Programming       1.699841

                  2055
```

```
model        scenario          region variable                   unit  type
MENR (2006) Baseline Scenario Turkey Primary Energy|Biomass    EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Coal       EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Gas        EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Hydro      EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Nuclear    EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Oil        EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Solar      EJ/yr Market
Based Simulation       NaN
                                     Primary Energy|Wind       EJ/yr Market
Based Simulation       NaN
MENR (2023) CO2 Scenario      Turkey Primary Energy|Coal       EJ/yr Linear
Programming       0.376812
                                     Primary Energy|Gas        EJ/yr Linear
Programming       1.226732
                                     Primary Energy|Nuclear    EJ/yr Linear
Programming       3.068924
                                     Primary Energy|Oil        EJ/yr Linear
Programming       0.586152
                                     Primary Energy|Renewables EJ/yr Linear
Programming       5.233500
```

[18]:
```
type(display_df)
```

[18]:
```
pyam.core.IamDataFrame
```

Filtering by year Filtering for **years** can be done by one integer value, a list of integers, or the Python class range.

The last year of a range is not included, so `range(2020, 2050)` is interpreted as `[2020, 2030, 2040]`.

The next cell shows the same down-selected **IamDataFrame** as above, but further reduced to three timesteps.

[19]:
```
display_df.filter(year=range(2020,2050)).timeseries()
```

[19]:
```
                              2020  \
model        scenario          region variable                   unit  type
MENR (2006) Baseline Scenario Turkey Primary Energy|Biomass    EJ/yr Market
Based Simulation  0.167472
                                     Primary Energy|Coal       EJ/yr Market
```

```
Based Simulation  1.561676
                                       Primary Energy|Gas        EJ/yr Market
Based Simulation  0.008374
                                       Primary Energy|Hydro      EJ/yr Market
Based Simulation  0.376812
                                       Primary Energy|Nuclear    EJ/yr Market
Based Simulation  0.334944
                                       Primary Energy|Oil        EJ/yr Market
Based Simulation  0.041868
                                       Primary Energy|Solar      EJ/yr Market
Based Simulation  0.041868
                                       Primary Energy|Wind       EJ/yr Market
Based Simulation  0.041868
MENR (2023) CO2 Scenario        Turkey Primary Energy|Coal       EJ/yr Linear
Programming       1.699841
                                       Primary Energy|Gas        EJ/yr Linear
Programming       1.666346
                                       Primary Energy|Nuclear    EJ/yr Linear
Programming            NaN
                                       Primary Energy|Oil        EJ/yr Linear
Programming       1.766830
                                       Primary Energy|Renewables EJ/yr Linear
Programming       1.029953

                  2030
model       scenario          region variable                    unit  type
MENR (2006) Baseline Scenario Turkey Primary Energy|Biomass      EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Coal       EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Gas        EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Hydro      EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Nuclear    EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Oil        EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Solar      EJ/yr Market
Based Simulation       NaN
                                       Primary Energy|Wind       EJ/yr Market
Based Simulation       NaN
MENR (2023) CO2 Scenario        Turkey Primary Energy|Coal       EJ/yr Linear
Programming       2.005477
                                       Primary Energy|Gas        EJ/yr Linear
Programming       1.997104
                                       Primary Energy|Nuclear    EJ/yr Linear
```

```
Programming       0.334944
                                   Primary Energy|Oil        EJ/yr Linear
Programming       2.294366
                                   Primary Energy|Renewables EJ/yr Linear
Programming       1.699841
```

1.6.5 Parallels to the *pandas* data analysis toolkit

When developing **pyam**, we followed the syntax of the Python package **pandas** (read the docs) closely where possible. In many cases, you can use similar functions directly on the **IamDataFrame**.

In the next cell, we illustrate this parallel behaviour. The function pyam.IamDataFrame.head() is similar to pandas.DataFrame.head(): it returns the first n rows of the 'data' table in **long format** (columns are in year/value format).

Similar to the timeseries() function shown above, the returned object of head() is a pandas.DataFrame.

[20]:
```
display_df.head()
```

[20]:
```
         model           scenario  region                   variable   unit  \
0  MENR (2006)  Baseline Scenario  Turkey  Primary Energy|Biomass  EJ/yr
1  MENR (2006)  Baseline Scenario  Turkey  Primary Energy|Biomass  EJ/yr
2  MENR (2006)  Baseline Scenario  Turkey     Primary Energy|Coal  EJ/yr
3  MENR (2006)  Baseline Scenario  Turkey     Primary Energy|Coal  EJ/yr
4  MENR (2006)  Baseline Scenario  Turkey      Primary Energy|Gas  EJ/yr

   year                    type     value
0  2010  Market Based Simulation  0.167472
1  2020  Market Based Simulation  0.167472
2  2010  Market Based Simulation  0.975524
3  2020  Market Based Simulation  1.561676
4  2010  Market Based Simulation  0.008374
```

[21]:
```
type(display_df.head())
```

[21]:
```
pandas.core.frame.DataFrame
```

1.6.6 Getting help

When in doubt, you can look at the help for any function by appending a `?`.

[22]:
```
df.filter?
```

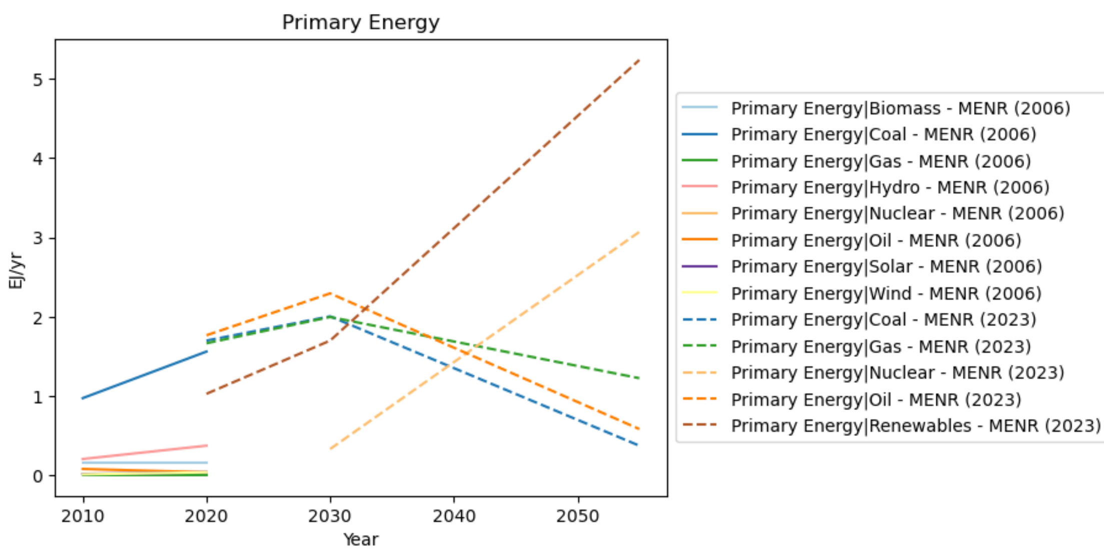
1.7 Visualize timeseries data using the plotting library

This section provides an illustrative example of the plotting features of the **pyam** package.

In the next cell, we show a simple line plot of global CO2 emissions. The colours are assigned randomly by default, and **pyam** deactivates the legend if there are too many lines.

[23]:
```
%%capture --no-display
from pyam.plotting import OUTSIDE_LEGEND
cmap = 'Paired'
display_df.filter(region='Turkey').plot(title='Primary Energy',
                                        color='variable', linestyle="model",␣
 ↪cmap=cmap, legend={"loc":"outside right"})
```

[23]: <AxesSubplot: title={'center': 'Primary Energy'}, xlabel='Year', ylabel='EJ/yr'>



The section on categorization will show more options of the plotting features, as well as a method to set specific colors for different categories. For more information, look at the other tutorials and the plotting gallery.

1.8 Visualize timeseries data using the plotting library

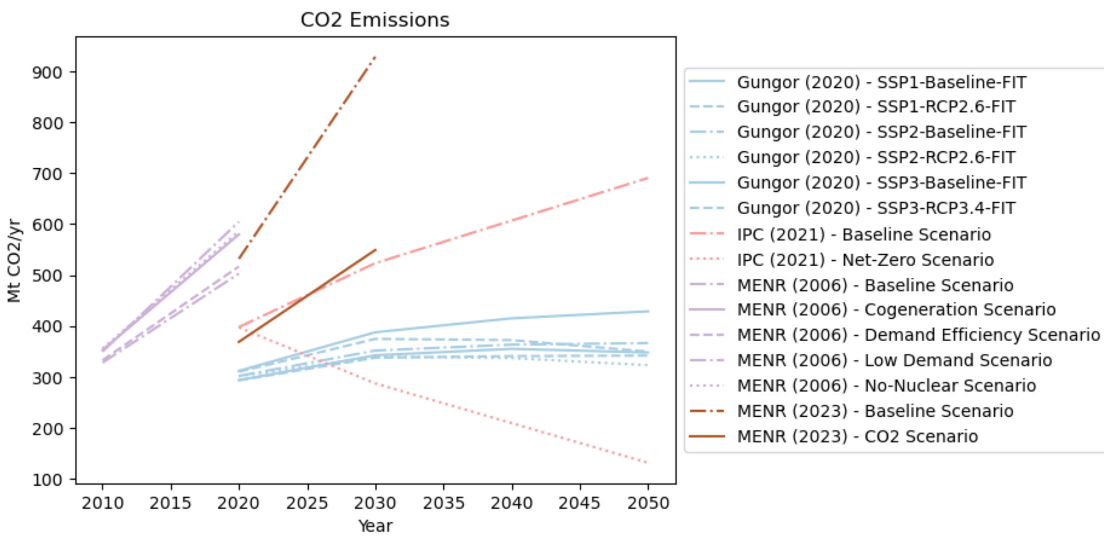
This section provides an illustrative example of the plotting features of the **pyam** package.

In the next cell, we show a simple line plot of estimated CO2 emissions. The colours are assigned randomly by default, and **pyam** deactivates the legend if there are too many lines. The **MENR (2023)** values are taken from the Updated 1st NDC of Turkey to the UNFCCC and converted from CO2eq to CO2 using the factor *0.79* calculated from the average ratio between CO2 and CO2eq (excluding LULUCF) emissions in 2022 National Inventory Report of Turkey.

[25]:
```
%%capture --no-display
cmap = 'Paired'
df.filter(variable='Emissions|CO2', region='Turkey').plot(color='model',␣
 ↪title='CO2 Emissions',
```

```
                                               linestyle='scenario',␣
↪cmap=cmap, legend={"loc":"outside right"})
```

[25]: <AxesSubplot: title={'center': 'CO2 Emissions'}, xlabel='Year', ylabel='Mt CO2/yr'>



1.9 Perform scenario diagnostic and validation checks

When analyzing scenario results, it is often useful to check whether certain timeseries data exist or the values are within a specific range. For example, it may make sense to ensure that reported data for historical periods are close to established reference data or that near-term developments are reasonable.

Before diving into the diagnostics and validation features, we need to briefly introduce the 'meta' table. This attribute of an **IamDataFrame** is a pandas.DataFrame, which can be used to store categorization information and quantitative indicators of each model-scenario. Per default, a new **IamDataFrame** will contain a column `exclude`, which is set to `False` for all model-scenarios.

The next cell shows the first 10 rows of the 'meta' table.

[26]:
```
df.meta.head(10)
```

[26]:

model	scenario	exclude
Gungor (2020)	SSP1-Baseline-FIT	False
	SSP1-RCP2.6-FIT	False
	SSP2-Baseline-FIT	False
	SSP2-RCP2.6-FIT	False
	SSP3-Baseline-FIT	False
	SSP3-RCP3.4-FIT	False

```
IPC (2021)     Baseline Scenario        False
               Net-Zero Scenario        False
MENR (2006)    Baseline Scenario        False
               Cogeneration Scenario    False
```

The following section provides three illustrations of the diagnostic tools: 0. Verify that a timeseries `Primary Energy` exists in each scenario (in at least one year and, in a second step, in the last year of the horizon). 1. Validate whether scenarios deviate by more than 10% from the `Primary Energy` reference data reported in the *IEA Energy Statistics* in 2010. 2. Use the `exclude_on_fail` option of the validation function to create a sub-selection of the scenario ensemble.

1.9.1 Check for required variables

We first use the require_variable() function to assert that the scenarios contain data for the expected timeseries.

[27]:
```
df.require_variable(variable='Primary Energy', year=2020)
```

```
pyam.core - INFO: 10 scenarios do not include required variable `Primary Energy`
```

[27]:

	model	scenario
0	IPC (2021)	Baseline Scenario
1	IPC (2021)	Net-Zero Scenario
2	MENR (2006)	Cogeneration Scenario
3	MENR (2006)	Demand Efficiency Scenario
4	MENR (2006)	Low Demand Scenario
5	MENR (2006)	No-Nuclear Scenario
6	MENR (2023)	Baseline Scenario
7	TUBITAK (2012)	Baseline Scenario
8	TUBITAK (2012)	Optimistic Scenario
9	TUBITAK (2012)	Pessimistic Scenario

[28]:
```
df.require_variable(variable='Primary Energy', year=2050)
```

```
pyam.core - INFO: 12 scenarios do not include required variable `Primary Energy`
```

[28]:

	model	scenario
0	IPC (2021)	Baseline Scenario
1	IPC (2021)	Net-Zero Scenario
2	MENR (2006)	Baseline Scenario
3	MENR (2006)	Cogeneration Scenario
4	MENR (2006)	Demand Efficiency Scenario
5	MENR (2006)	Low Demand Scenario
6	MENR (2006)	No-Nuclear Scenario
7	MENR (2023)	Baseline Scenario
8	MENR (2023)	CO2 Scenario
9	TUBITAK (2012)	Baseline Scenario
10	TUBITAK (2012)	Optimistic Scenario
11	TUBITAK (2012)	Pessimistic Scenario

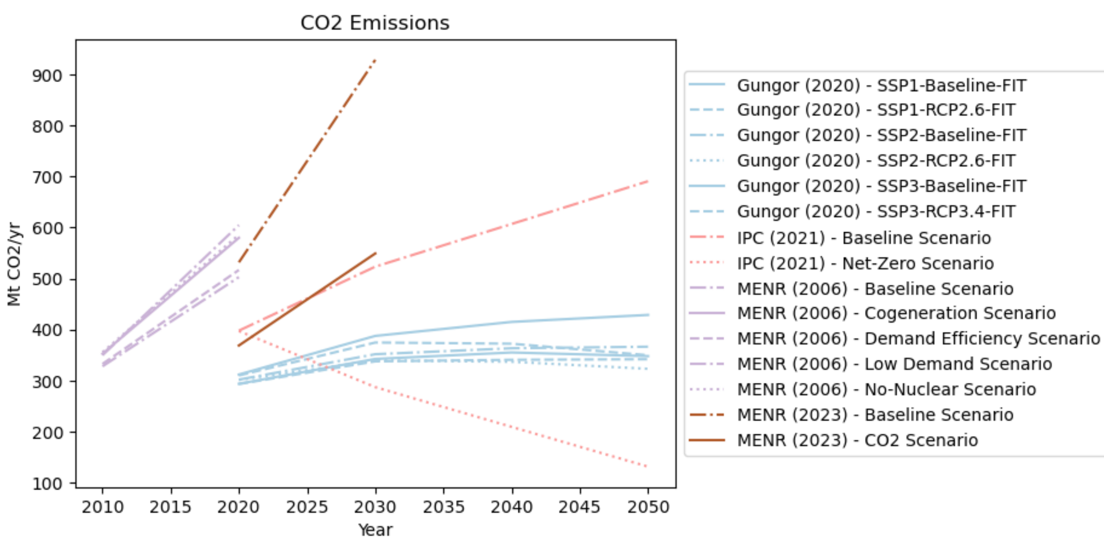
1.9.2 Use the `exclude_on_fail` feature to create a sub-selection of the scenario ensemble

Per default, the functions above only report how many scenarios or which data points do not satisfy the validation criteria above. However, they also have an option to `exclude_on_fail`, which marks all scenarios failing the validation as `exclude=True` in the 'meta' table. This feature can be particularly helpful when a user wants to perform a number of validation steps and then efficiently remove all scenarios violating any of the criteria as part of a scripted workflow.

We illustrate a simple validation workflow using the CO2 emissions. The next cell shows the trajectories of CO2 emissions across all scenarios.

[29]:
```
%%capture --no-display
df.filter(variable='Emissions|CO2').plot(color='model', title='CO2 Emissions',
                                          linestyle='scenario', cmap=cmap,␣
 ↪legend={"loc":"outside right"})
```

[29]:
```
<AxesSubplot: title={'center': 'CO2 Emissions'}, xlabel='Year', ylabel='Mt
CO2/yr'>
```



The next two cells perform validation to exclude all scenarios that have implausibly low emissions in 2020 (i.e., unrealistic near-term behaviour) as well as those that do not reduce emissions over the modeling horizon (i.e., exceed a value of 600 MT CO2 in any year).

[40]:
```
df.validate(criteria={'Emissions|CO2': {'lo': 300, 'year': 2020}},␣
 ↪exclude_on_fail=True)
```

```
pyam.core - INFO: 2 of 294 data points do not satisfy the criteria
pyam.core - INFO: 2 non-valid scenarios will be excluded
```

```
[40]:           model          scenario  region       variable       unit  year  \
      0  Gungor (2020)  SSP1-Baseline-FIT  Turkey  Emissions|CO2  Mt CO2/yr  2020
      1  Gungor (2020)    SSP1-RCP2.6-FIT  Turkey  Emissions|CO2  Mt CO2/yr  2020

                       type    value
      0  Linear Programming  293.826
      1  Linear Programming  293.363
```

```
[41]: df.validate(criteria={'Emissions|CO2': {'up': 600}}, exclude_on_fail=True)
```

```
pyam.core - INFO: 3 of 294 data points do not satisfy the criteria
pyam.core - INFO: 3 non-valid scenarios will be excluded
```

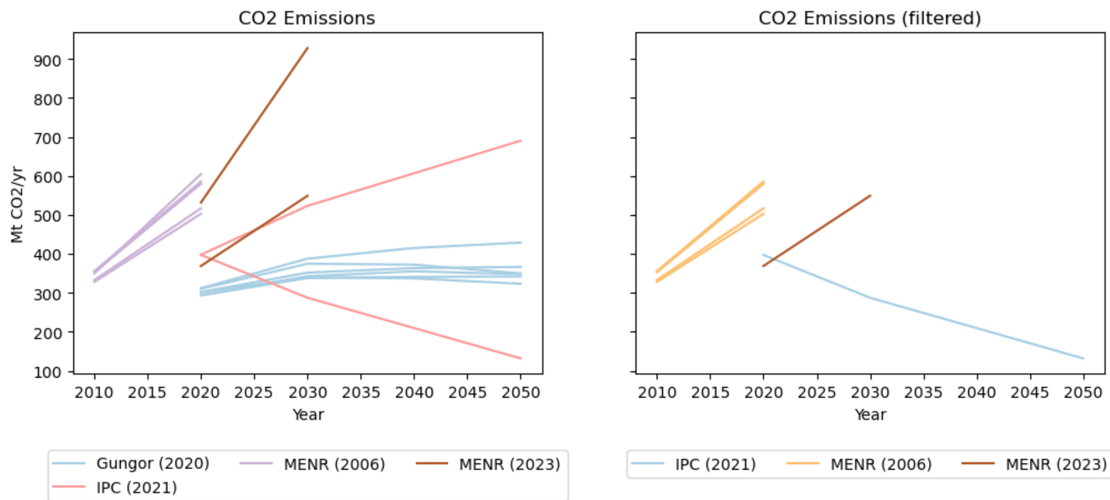
```
[41]:        model          scenario  region       variable       unit  year  \
      0   IPC (2021)  Baseline Scenario  Turkey  Emissions|CO2  Mt CO2/yr  2050
      1  MENR (2006)  Baseline Scenario  Turkey  Emissions|CO2  Mt CO2/yr  2020
      2  MENR (2023)  Baseline Scenario  Turkey  Emissions|CO2  Mt CO2/yr  2030

                             type   value
      0                       CGE  690.50
      1  Market Based Simulation  604.63
      2       Linear Programming  928.25
```

We can select all scenarios that have *not* been marked to be excluded by adding `exclude=False` to the filter() statement.

To highlight the difference between the full scenario set and the reduced scenario set based on the validation exclusions, the next cell puts the two plots side by side with a shared y-axis.

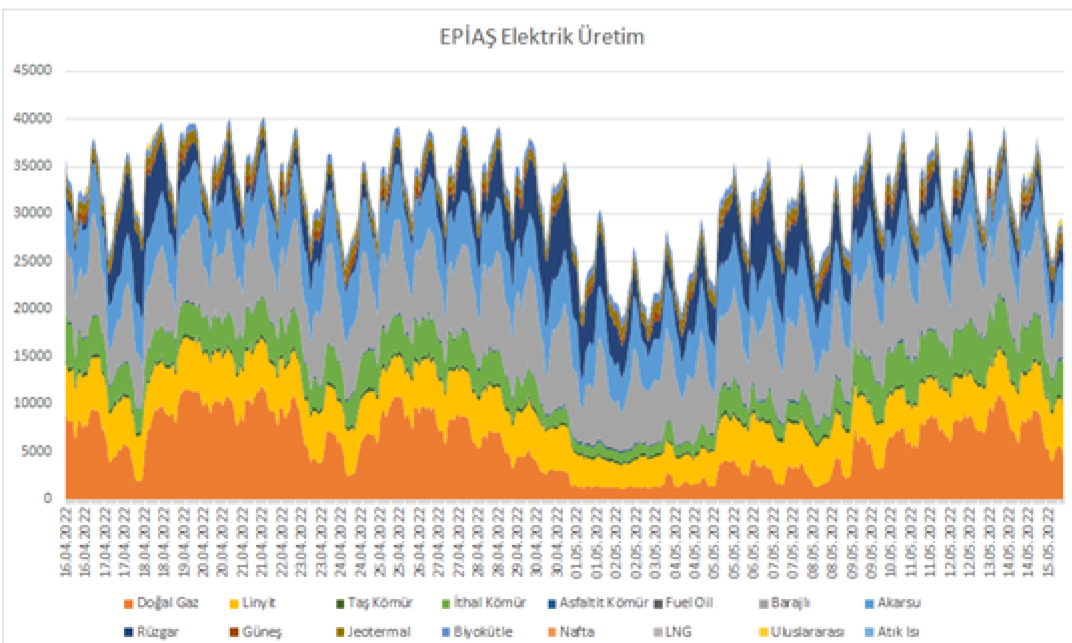
```
[54]: %%capture --no-display
fig, ax = plt.subplots(1, 2, figsize=(12, 4), sharey=True)
df_co2 = df.filter(variable='Emissions|CO2')
df_co2.plot(ax=ax[0],title='CO2 Emissions',color='model',
            cmap=cmap, legend={"loc":"outside bottom"})
df_co2.filter(exclude=False).plot(ax=ax[1],title='CO2 Emissions␣
 ↪(filtered)',color='model',
                                  cmap=cmap, legend={"loc":"outside bottom"})
```

```
[54]: <AxesSubplot: title={'center': 'CO2 Emissions (filtered)'}, xlabel='Year',
      ylabel='Mt CO2/yr'>
```

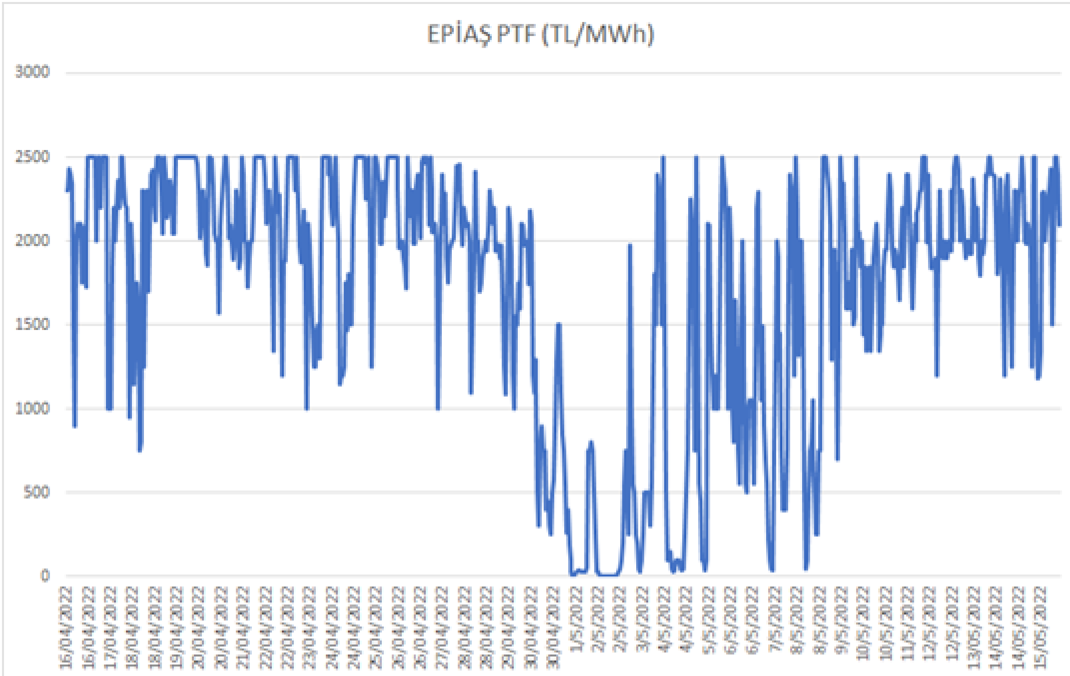


2 Energy Market

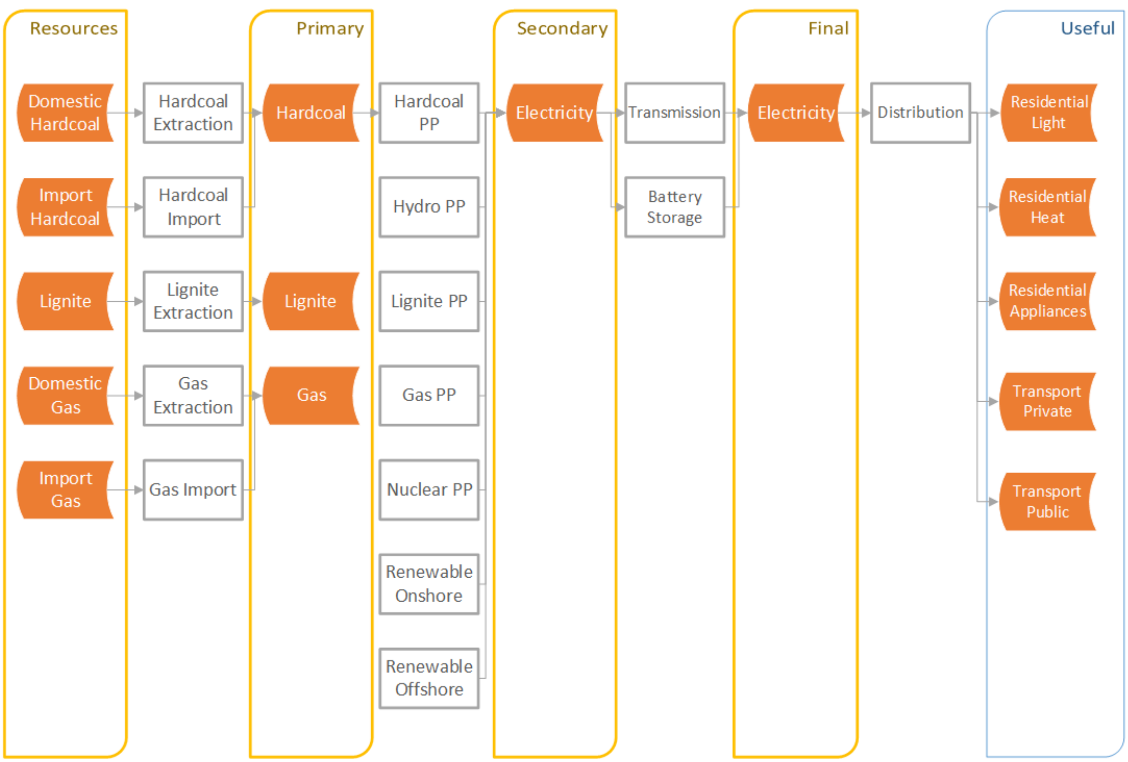
The energy market exchange amounts and prices are continuously published by the energy market operator EPİAŞ Transparency Platform.



The one-month period from 16th of April to 16th of May 2022 includes Ramadan holiday where electricity demand is reduced. The market exchange price, which is around the cap during workdays, drops during the holiday period.



2.1 Energy flows for electricity generation with storage



2.2 Further steps

- Include data from recent academic (peer-reviewed) studies based on the net-zero target of Turkey

- Extract meta-data for emissions and related temperature increase using **MAGICC** emulator
- Develop a model for the low carbon transition of the electricity sector
- Test the hypothesis for utilizing hydrogen and battery storage as a market solution for low carbon transition

2.3 Questions?

Take a look at our GitHub repository!

[55]:
```
df.to_excel('data_export.xlsx')
```