Hands-on exercise 2: Running MUSE

Once MUSE have been installed, we can run an example. To start with, we will run one of the built-in MUSE examples. If you are using MUSE within a virtual environment, make sure you have it activated (refer back to exercise 1 if you need help with this).

You should be able to run the default `muse` example running the following command in the terminal:

```
python -m muse --model default
```

If running correctly, your prompt should output text similar to this (https://muse-os.readthedocs.io/en/v1.2.1/example-output.html). You can check the available built-in models, as well as information on other input arguments, with:

```
python -m muse -h
```

A common use case is to take one of the built-in models as the starting point to create your own model. This is the approach we will take in the hands-on exercises in this course. To copy the files for the default model, run:

```
python -m muse --model default --copy path/to/copy/the/model/to
```

This will create a folder called `model` in the specified path. Navigate to this folder using the `cd` command, and use `ls` to see the contents of this folder. We can then run the simulation using the following command:

```
python -m muse settings.toml
```

Results

If the default MUSE example has run successfully, you should now have a folder called `Results` in the current working directory.

This directory should contain two files:

- `MCACapacity.csv` : contains information about the capacity each agent has per technology per benchmark year. Each benchmark year is the modelled year in the `settings.toml` file. In our example, this is 2020, 2025, ..., 2050.
- `MCAPrices.csv` : has the converged price of each commodity per benchmark year and timeslice. eg. the cost of electricity at night in 2020.

Additional files can be added by modifying `settings.toml` , as will be shown in future exercises.

For the hands-on exercises in this course, we will use Python and Jupyter Notebook (https://jupyter.org) to visualise these simulation results. You can, however, visualise the results using any language or program of your choice (for example Excel, R, MATLAB), but will get the most out of these exercises if you use Jupyter Notebook.

Installing Jupyter

First, you will need to install Jupyter Notebook in your environment, which you can do with the following command:

```
python -m pip install jupyter
```

Then, once this has been installed, you can start Jupyter Notebook by running the following command:

```
python -m jupyter notebook
```

A web browser should now open up with a URL such as the following: `http://localhost:8888/tree` . If it doesn't, copy and paste the command as directed in the terminal. This will likely take the form of:

```
http://localhost:8888/?token=xxxxxxxxxx
```

With `xxxxxxxxxx` a very long collection of letters and numbers. Once you are on the page, you will be able to navigate to a location of your choice and create a new file, by clicking the `New` button in the top right, followed by `Python 3` . You should then be able to proceed and follow the tutorials in this documentation.

Missing packages

If, when running a cell, you get any errors such as:

```
ModuleNotFoundError: No module named 'pandas'
```

Then you are trying to use a package (`pandas` in the example) that is not available in the current environment. It is possible to install the missing packages by running the following in the Jupyter notebook:

```
!pip install pandas
```

The package will be installed in whatever virtual environment Jupyter is running in.

Visualisation

First, we need to load the appropriate packages required to load and visualise the results, which you can do by running the following Python commands in a notebook cell:

In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
```

Next, we will load `MCACapacity.csv` file and print the first five rows of the table using the pandas library:

In [2]:

```
mca_capacity = pd.read_csv("Results/MCACapacity.csv")
mca_capacity.head()
```

Out[2]:

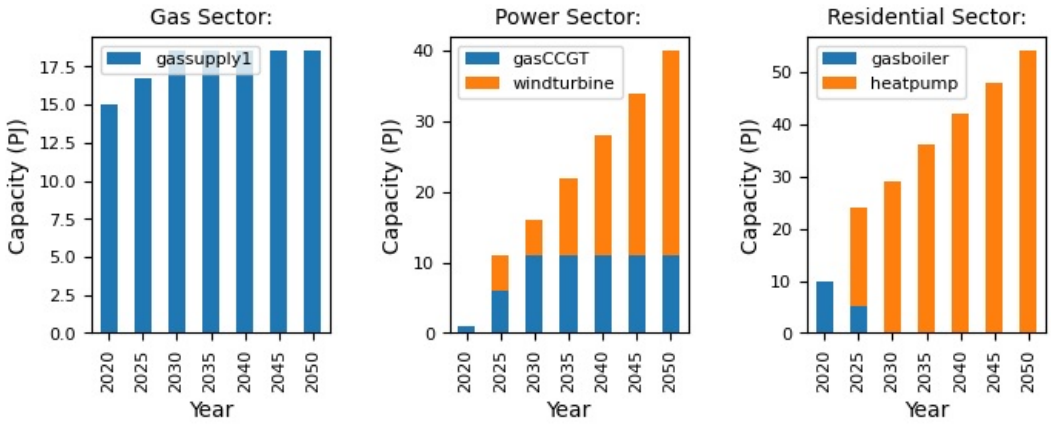
	agent	capacity	dst_region	installed	region	sector	technology	type	year
0	A1	10.0	R1	2020	R1	residential	gasboiler	newcapa	2020
1	A1	1.0	R1	2020	R1	power	gasCCGT	newcapa	2020
2	A1	15.0	R1	2020	R1	gas	gassupply1	newcapa	2020
3	A1	5.0	R1	2020	R1	residential	gasboiler	newcapa	2025
4	A1	19.0	R1	2020	R1	residential	heatpump	newcapa	2025

Finally, we will visualise the data for each of the sectors, with capacity on the y-axis and year on the x-axis. Don't worry too much about the code if some of it is unfamiliar - we effectively split the data into each sector, sum the capacity for each technology, and then create a stacked bar chart for each.

In [3]:

```
fig, axes = plt.subplots(1, 3)
for ax, (sector_name, sector_data) in zip(axes, mca_capacity.groupby("sector")):
    sector_capacity = sector_data.groupby(["year", "technology"]).sum().reset_index()
    sector_capacity.pivot(index="year", columns="technology", values="capacity").plot(
        kind="bar", stacked=True, ax=ax
    )
    ax.set_ylabel("Capacity (PJ)")
    ax.set_xlabel("Year")
    ax.set_title(f"{sector_name.capitalize()} Sector:", fontsize=10)
    ax.legend(title=None, prop={"size": 8})
    ax.tick_params(axis="both", labelsize=8)

fig.set_size_inches(8, 2.5)
fig.subplots_adjust(wspace=0.5)
```



In this toy example, we can see that the end-use technology of choice in the residential sector becomes a heatpump, which displaces the gas boiler. To account for the increase in demand for electricity, the agent invests heavily in wind turbines.

Note, that the units are in petajoules (PJ). MUSE requires consistent units across each of the sectors, and each of the input files (which we will see later). The model does not make any unit conversion internally.

Summary

In this exercise we have shown how to run the default model that comes with MUSE, and how to visualise the investment decisions made in the simulation. In future exercises we will show how to modify the input files to run different, more interesting, scenarios.