Hands On Exercise 1: Installing MUSE

Short description

This hands-on exercise will allow you to install MUSE on your computer. We will then take you though an example to run and visualise a default MUSE example.

If at any point you get stuck with these hands-on exercises, feel free to post a question in the purpose-made MUSE google group, if your question hasn’t already been answered there!:

https://groups.google.com/g/muse-model

Learning objectives

* Install MUSE
* Run an example
* Visualise the results of the example

Exercise content

For Windows users only

Windows users and developers may need to install Windows Build Tools\_\_. These tools include C/C++ compilers which are needed to build some python dependencies.

MacOS includes compilers by default, hence no action is needed for Mac users.

Linux users may need to install a C compiler, whether GNU gcc or Clang, as well python development packages, depending on their distribution.

If you have MacOS or Linux you can skip this section and head to the next section below here.

1. Install Visual Studio from the following link: https://visualstudio.microsoft.com/downloads/
2. Select your preferred edition. Although, the “Community” version is free and contains what is required.
3. Install Visual Studio by selecting the default options.
4. Download the Microsoft Visual C++ Build Tools from the following link by downloading Visual Studio 2019: https://visualstudio.microsoft.com/downloads/
5. Select your preferred edition. The “Community” is free and contains what is required.
6. Run the installer
7. Select: Workloads → Desktop development with C++.
8. Install options: select only the “Windows 10 SDK” (assuming the computer is Windows 10)]. This will come up on the right-hand side of the screen.

The installation screen should look similar to the following:

Graphical user interface, application

Description automatically generated

**Figure 1.1:** Visual Studio Installer window

For further information, see this link: https://www.scivision.dev/python-windows-visual-c-14-required

Installing MUSE

MUSE is developed using python, an open-source programming language, which means that there are two steps to the installation process. First, python should be installed. Then so should MUSE.

The simplest method to install python is by downloading the Anaconda distribution. Make sure to choose the appropriate operating system (e.g. windows), python version 3.8, and the 64 bit installer. Once this has been done follow the steps for the anaconda installer, as prompted.

After python is installed we can install MUSE. MUSE can be installed via the Anaconda Prompt (or any terminal on Mac and Linux). This is a command-line interface to python and the python eco-system. In the anaconda prompt, run:

python -m pip install --user git+https://github.com/SGIModel/StarMuse

It should now be possible to run muse. Again, this can be done in the anaconda prompt as follows:

python -m muse --help

Running your first example

In this section we run an example simulation of MUSE, in the next section we will visualise the results.

First we need to download the MUSE source code. To do that navigate to the MUSE GitHub repository: https://github.com/SGIModel/StarMuse

Click on the green Code button in the top right-hand corner and then click on Download ZIP. Figure 2.1 shows how to do this, once you are on the relevant page.

Graphical user interface, text, application, email

Description automatically generated

**Figure 2.1:** How to download MUSE

Once you have downloaded the source code, unzip the folder and move it to a location that is convenient for you.

We will place ours on the desktop for simplicity, but feel free to make a folder in your documents or otherwise.

To run MUSE, we must open the anaconda prompt for Windows machines or terminal if on MacOS or Linux. Then we must navigate to the directory using the prompt or terminal to find the MUSE examples. Ours is in Desktop, so we will run the following command:

cd ~/Desktop/StarMuse/run/example/default/

But yours could be in another location, so fill in the {MUSE\_download\_location} with your path (This could be /Users/{my\_name}/Documents for instance):

cd {MUSE\_download\_location}/StarMuse/run/example/default/

Once we have navigated to the directory containing the example settings (settings.toml) we can run the simulation using the following command in the anaconda prompt or terminal:

python -m muse settings.toml

If running correctly, your prompt should output text similar to the following:

-- 2020-11-03 15:58:29 - muse.sectors.register - INFO

Sector legacy registered.

-- 2020-11-03 15:58:29 - muse.sectors.register - INFO

Sector preset registered, with alias presets.

-- 2020-11-03 15:58:29 - muse.sectors.register - INFO

Sector default registered.

-- 2020-11-03 15:58:29 - muse.readers.toml - INFO

Reading MUSE settings

-- 2020-11-03 15:58:29 - muse.readers.toml - INFO

Default input values used: carbon\_budget\_control.commodities, carbon\_budget\_control.method, carbon\_budget\_control.debug, carbon\_budget\_control.control\_undershoot, carbon\_budget\_control.control\_overshoot, carbon\_budget\_control.method\_options

Results

If the default MUSE example has run successfully, you should now have a folder called Results in the same directory as settings.toml.

This directory should contain results for each sector contained within this example (Gas, Power and Residential) as well as results for the entire simulation in the form of the files MCACapacity.csv and MCAPrices.csv.

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. For example, it has the cost of electricity at night for electricity in 2020, and other similar results.

Within each of the sector result folders, there is an output for Capacity for each commodity in each year. Future years, which the simulation has not run to, refers to the capacity as it retires. Within the Residential folder there is also a folder for Supply within each year. This refers to how much end-use commodity was output.

Visualisation

There are many different ways to visualise the results of MUSE. For example, you could use a programming language such as python or R. In this course, however, we will use Excel for simplicity.

There are also many different variables and combinations of data that we can plot. In this course we will primarily explore the capacity installed over the time horizon (2020--2050). Through this, we can see which of the technologies are invested in and understand the competition between technologies.

To start the visualisation process of the default example, navigate to the folder where you run the default example. For instance:

cd {MUSE\_download\_location}/StarMuse/run/example/default/

Go into the folder called /Results/ and right click on the file called MCACapacity.csv and open it with Microsoft Excel. Once you’ve opened the file with Excel, we can begin the data visualisation process.

First, select the PivotChart button under the insert menu. You should then see a box open up such as that shown in Figure 1.3, below:

Graphical user interface, text, application

Description automatically generated

**Figure 1.3:** Insert -> Create PivotChart

Ensure that the “Select a table or range” highlights all of the data, and the “Choose where you want the PivotChart to be placed” is selected to “Existing worksheet” like in the figure above. Then within the “Table/Range” box, click the cell where you would like the figure to be placed. Click “OK” when finished.

You should then see the boxes appear as shown below in Figure 1.4 in Excel:

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 1.4:** Inserted PivotChart

Next we want to view how the capacity changes per technology over the technology horizon for the power sector.

To do this, we must drag the relevant Fields from the PivotTable to the Filters, Columns, Rows and Values boxes.

First, we will drag the capacity field to the Values boxes. These are the values that we want to plot. Then we will drag the year field to the Axis (Categories) box. This will start to populate a table and a graph. Finally, we want to filter for the residential sector. To do that we will drag the sector field to the Filters box.

We should then end up with the chart, table and PivotTable fields shown in Figure 1.5.

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 1.5:** Default visualisation example.

However, this is displaying all sectors currently, so to filter for solely the residential sector, we must click the dropdown arrow next to the “sector, (All)” cells.

This can be shown in Figure 1.6, below:

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 1.6:** Residential sector visualisation for default example.

We have unchecked all sectors apart from the residential sector to generate the results shown in Figure 1.6.

We can see that gas boilers are the main technology in 2020, but this reduces to zero by 2040. Heatpumps take over between 2020 and 2050, increasing significantly as demand increases.

Summary

In this hands-on we have installed MUSE, learnt how to run a demo example and visualised the results of this demo example. In the next lectures and hands-on we will learn in detail the fundamentals of MUSE and how to edit the demo example for a case study of our choice.

Hands On Exercise 2: Modifying a service demand

This hands-on will allow users to define their own service demand for an exogenous sector or by correlation.

Learning objectives

* Define own service demand for an exogenous sector
* Define own service demand by correlation

Adding an exogenous service demand

As a quick example, in the residential sector a service demand could be cooking. Houses require energy to cook food and a technology to service this demand, such as an electric stove.

We will start by looking at the default example. This can be found in your MUSE download at /src/muse/data/example/default/example, or you can download it at the following link:

https://zenodo.org/record/6022713#.YgOYyS-l1pQ

Next, download this and place it in a convenient location on your computer. We will now start by adding a cooking demand to this example. The default example currently only has a service demand of heat, so we will need to do some editing.

To achieve this, we will need to edit the Residential2020Consumption.csv and Residential2050Consumption.csvfiles found within the technodata/preset/ directory. The Residential2020Consumption.csv file allows us to specify the demand in 2020 for each region and technology per timeslice. The Residential2050Consumption.csv file does the same but for the year 2050. The datapoints between these are interpolated.

Firstly, we must add the new service demand cook as a column in these two files. Next, we add the demand. We can do this in Excel, or an editor of your choice. This is how it may look like for you when you open the Residential2020Consumption.csv file:

Table

Description automatically generated

**Figure 2.1:** Residential2020Consumption file opened in Excel.

We will add a new column called cook and enter some values for each timeslice. This can be seen through the addition of a positive number in the cook column.

Table

Description automatically generated

**Figure 2.2:** Modified Residential2020Consumption file opened in Excel.

The process is very similar for the Residential2050Consumption.csv file, however, for this example, we often placed larger numbers to indicate higher demand in 2050.

Next, we must edit the files within the input folder. For this, we must add the cook service demand to each of these files.

First, we will amend the BaseYearExport.csv and BaseYearImport.csv files. For this, we say that there is no import or export of the cook service demand. A brief example is outlined below for BaseYearExport.csv:

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 2.3:** Modified BaseYearImport file opened in Excel.

The same is true for the BaseYearImport.csv file:

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 2.4:** Modified BaseYearExport file opened in Excel.

Next, we must edit the GlobalCommodities.csv file. This is where we define the new commodity cook . It tells MUSE the commodity type, name, emissions factor of CO2 and heat rate, amongst other things.

The default version used for this tutorial is below:

Table

Description automatically generated

**Figure 2.5:** Non-edited GlobalCommodities file opened in Excel.

We then add a new row at the bottom to include the cook commodity:

Table

Description automatically generated

**Figure 2.6:** Edited GlobalCommodities file opened in Excel.

Finally, the Projections.csv file must be changed. This is a large file which details the expected cost of the technology in the first benchmark year of the simulation. We have highlighted in **bold** the changed column for this example.

Table, Excel

Description automatically generated

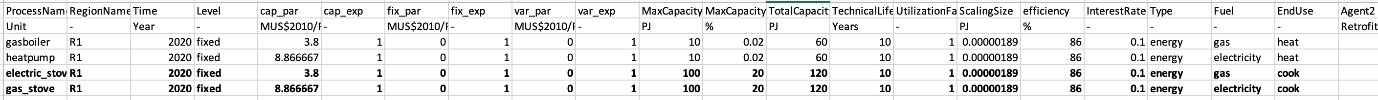
**Figure 2.7:** Edited Projections file opened in Excel.

Addition of a cooking technology

Next, we must add a technology to service this new demand. During this process we must be careful to specify the end-use of the technology as cook.

For this example, we will add two competing technologies to service the cooking demand: electric\_stove and gas\_stove to the Technodata.csv file in /technodata/residential/Technodata.csv.

For this, we copy the gasboiler row for R1 and paste it for the new electric\_stove. For gas\_stove we copy and paste the data for heatpump from region R1. In the figure below we show this, but only show the first few columns for the interest of space. We will also relax the growth constraints to ensure that the growth in technologies can meet demand.



**Figure 2.8:** Edited technodata file opened in Excel.

As can be seen we have added two technologies with different cap\_par costs to each other. We specified their respective fuels, and the enduse for both is cook.

We must also add the data for these new technologies to the following files:

* CommIn.csv
* CommOut.csv
* ExistingCapacity.csv

The CommIn.csv file details the input commodities for each technology. In this case, the inputs are gas and electricity. The CommOut file details the outputs of the technology, which will be the cook commodity.

We must add the input to each of the technologies (electricity and gas for electric\_stove and gas\_stove  respectively), outputs of cook for both and the existing capacity for each technology.

Table

Description automatically generated

**Figure 2.9:** Edited CommIn file opened in Excel.

Notice in Figure 2.9 that we had to add a column for the new cook. We must also do the same for the CommOut file, below:

Table

Description automatically generated

**Figure 2.10:** Edited CommOut file opened in Excel.

We must do this for the gas and power sector as well.

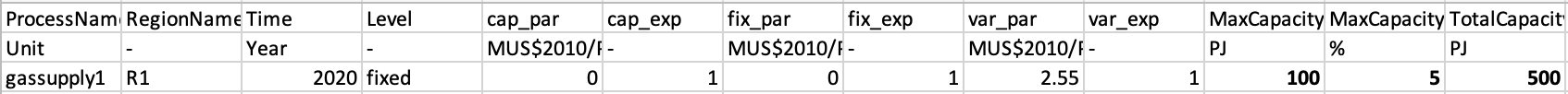
Next, we must edit the ExistingCapacity.csv file to detail how much existing capacity there is in the base year and beyond.

Table

Description automatically generated

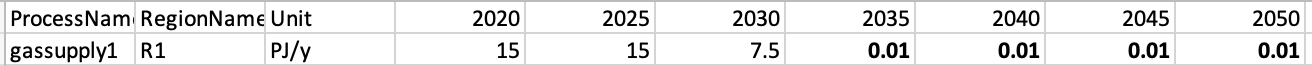
**Figure 2.11:** Edited ExistingCapacity file opened in Excel.

Due to the additional demand for gas and electricity brought on by the new cook demand, it is necessary to relax the growth constraints for gassupply1 in the technodata/gas/technodata.csv file. For this example, we set this file as follows:



**Figure 2.12:** Edited gas/technodata file opened in Excel.

We must also ensure there are no 0 in the ExistingCapacity.csv for any of the sectors. Therefore, go through the gas/ExistingCapacity.csv and power/ExistingCapacity.csv and replace them with a non-zero value, such as 0.01. Below is an example for the gas sector:



**Figure 2.13:** Edited gas/ExistingCapacity.csv file opened in Excel.

Next, we must run the simulation with our modified input files using the following command in the relevant directory:

python -m pip muse settings.toml

The figure below shows the results for this new demand in the residential sector:

Table

Description automatically generated with medium confidence

**Figure 2.14:** Capacity results for the residential sector.

We can see that electric\_stove takes over completely. This is because of the lower cap\_par value when compared to gas\_stove.

For the final example input data showed in this tutorial and results spreadsheet, please refer to the link below:

https://zenodo.org/record/6022713#.YgOYyS-l1pQ

Hands On Exercise 3: Productions constraints by timeslice

In this hands-on we explain how to add constraints to outputs of technologies at certain timeslices. This could either by a maximum constraint, for instance with the solar PV example mentioned in the previous lecture. Or, this could be a minimum constraint, where we expect a minimum amount of output by a nuclear power plant at all times.

Learning objectives

* Learn how to add min/max production constraints in MUSE

Minimum timeslice

In this tutorial we will be amending the default example, which you can find in the following zenodo link: https://zenodo.org/record/6026087#.YgPuHS-l1pQ

Firstly, we will be imposing a minimum service factor for gasCCGT in the power sector. This is the minimum that a technology can output per timeslice.

To do this, we will need to create a new csv file that specifies the minimum service factor per timeslice.

An example of the file, which also contains values for windturbine, can be seen below and in the zenodo link.

Table

Description automatically generated

**Figure 3.1:** TechnodataTimeslices.csv file for the power sector.

Notice that we have to specify the following columns: ProcessName, RegionName, Time, month, day, hour, UtilizationFactor, MinimumServiceFactor.

The majority of these columns are self explanatory, and correspond to the columns in other csv files - for instance, ProcessName, RegionName and Time. The timeslice based columns, however, are dynamic and will match the levels as defined in the toml file.

The majority of these columns are self explanatory, and correspond to the columns in other csv files - for instance, ProcessName, RegionName and Time. The timeslice based columns, however, are dynamic and will match the levels as defined in the settings.toml file in the main default folder.

[sectors.power]

type = 'default'

priority = 2

dispatch\_production = 'costed'

technodata = '{path}/technodata/power/Technodata.csv'

commodities\_in = '{path}/technodata/power/CommIn.csv'

commodities\_out = '{path}/technodata/power/CommOut.csv'

technodata\_timeslices = '{path}/technodata/power/TechnodataTimeslices.csv'

Once this has been completed, we are able to run MUSE as before, with the following command:

python -m muse settings.toml

We can then view the results as before using Excel.

Maximum timeslice constraint

Next, we want to ensure that the supply of windturbine does not exceed a certain value during the day. This may be because, for example, there is reduced wind during the day. We will, therefore, modify the TechnodataTimeslices.csv file by changing the values of UtilizationFactor. This is shown in the figure below, where we change the morning and afternoon timeslices to be 0.5, as an example.

Table

Description automatically generated

**Figure 3.2:** Edited TechnodataTimeslices file opened in Excel.

Once this has been saved, we can run the model again (python -m muse settings.toml). We can then visualise our results as before.

Summary

In this hands-on we have introduced the TimeslicesTechnodata.csv file and linked it to the settings.toml file. This has allowed us to vary the output of various energy technologies by their characteristics.

Hands On Exercise 4: Adding a technology

Learning objectives

* Learn how to add a new technology in MUSE

Addition of solar PV

In this section, we will add solar photovoltaics to the default model. To achieve this, we must modify the input files in the default example.

Technodata Input

We must note, before starting, that we require consistency in input and output units. For example, if capacity is in PJ, the same basis would be needed for the output files CommIn.csv and CommOut.csv. In addition, across sectors a commodity needs to maintain the same unit. In these examples, we use the unit petajoule (PJ).

Next, we will edit the CommIn.csv file, which specifies the commodities consumed by solar photovoltaics.

The table below shows the original CommIn.csv version in normal text, and the added column and row in bold.

Table

Description automatically generated

**Figure 4.1:** Modified CommIn.csv file for the power sector

We must first add a new row at the bottom of the file, to indicate the new solar photovoltaic technology:

* we call this technology solarPV
* place it in region R1
* the data in this row is associated to the year 2020
* the input type is fixed
* solarPV consumes solar

As the solar commodity has not been previously defined, we must define it by adding a column, which we will call solar. We fill out the entries in the solar column, ie. that neither gasCCGT nor windturbine consume solar.

We repeat this process for the file: CommOut.csv. This file specifies the output of the technology. In our case, solar photovoltaics only output electricity. This is unlike gasCCGT which also outputs CO2f, or carbon dioxide.

Table

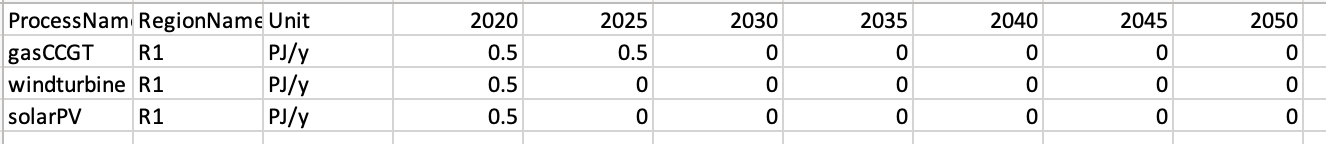
Description automatically generated

**Figure 4.2:** Modified CommOut.csv file for the power sector

Similar to the the CommIn.csv, we create a new row, and add in the solar commodity. We must ensure that we call our new commodity and technologies the same as the previous file for MUSE to successfully run. ie solar and solarPV.

Please note that we use flat forward extension of the values when only one value is defined. For example, in the CommOut.csv we only provide data for the year 2020. Therefore for the benchmark years, 2025, 2030, 2035… we assume the data remains unchanged from 2020.

The next file to modify is the ExistingCapacity.csv file. This file details the existing capacity of each technology, per benchmark year. For this example, we will set the existing capacity to be 0.5 for all technologies in the base year and 0 for the remaining years. Please note, that the model interpolates between years linearly.



**Figure 4.3:** Modified ExistingCapacity.csv file for the power sector

Finally, the technodata.csv containts parametrisation data for the technology, such as the cost, growth constraints, lifetime of the power plant and fuel used. The technodata file is too long for it all to be displayed here, so we will truncate the full version.

Here, we will only define the parameters: processName, RegionName, Time, Level,cap\_par, Fuel, EndUse, Agent2 and Agent1

We shall copy the existing parameters from the windturbine technology for the remaining parameters that can be seen in the technodata.csv file for brevity. You can see the full file at the zenodo link, below:

https://zenodo.org/record/6092287#.YgvOEy-l1pQ

Again, flat forward extension is used here. Therefore, as in this example we only provide data for the benchmark year 2020, 2025 and the following benchmark years will keep the same characteristics, e.g. costs, for each benchmark year of the simulation.

Table

Description automatically generated

**Figure 4.4:** Modified Technodata.csv file for the power sector

Notice that we have hidden the cells between F and T. These are the same as the windturbine technology, but we’ve changed the cap\_par input to 30 and the Fuel technology to solar.

Global inputs

Next, navigate to the input folder, found at:

{muse\_installation\_location}/src/muse/data/example/default/input

We must now edit each of the files found here to add the new solar commodity. Due to space constraints we will not display all of the entries contained in the input files. The edited files can be viewed in the zenodo link below, however.

https://zenodo.org/record/6092287#.YgvOEy-l1pQ

The BaseYearExport.csv file defines the exports in the base year. For our example we add a column to indicate that there is no export for solar. However, it is important that a column exists for our new commodity.

It is noted, however, that the BaseYearImport.csv as well as the BaseYearExport.csv files are optional files to define exogenous imports and exports; all values are set to zero if they are not used.

Table

Description automatically generated

**Figure 4.5:** Modified BaseYearExport.csv file for the power sector

The BaseYearImport.csv file defines the imports in the base year. Similarly to BaseYearExport.csv, we add a column for solar in the BaseYearImport.csv file. Again, we indicate that solar has no imports.

Table

Description automatically generated

**Figure 4.6:** Modified BaseYearImport.csv file for the power sector

The GlobalCommodities.csv file is the file which defines the commodities. Here we give the commodities a commodity type, CO2 emissions factor and heat rate. For this file, we will add the solar commodity, with zero CO2 emissions factor and a heat rate of 1.

Table

Description automatically generated

**Figure 4.7:** Modified GlobalCommodities.csv file for the power sector

The projections.csv file details the initial market prices for the commodities. The market clearing algorithm will update these throughout the simulation, however, an initial estimate is required to start the simulation. As solar energy is free, we will indicate this by adding a final column.

Please note that the unit row is not read by MUSE, but used as a reference for the user. The units should be consistent across all input files for MUSE; MUSE does not carry out any unit conversion.

Table, Excel

Description automatically generated

**Figure 4.8:** Modified projections.csv file for the power sector

Running our customised simulation

Now we are able to run our simulation with the new solar power technology.

To do this we run the same run command as previously in the anaconda command prompt:

python -m muse settings.toml

If the simulation has run successfully, you should now have a folder in the same location as your settings.toml file called Results. The next step is to visualise the results using Excel.

We will use the PivotChart, similar to the previous step.

Chart, funnel chart, surface chart

Description automatically generated

**Figure 4.9:** Visualisation with new technology.

The power sector now shows us the new solarPV technology.

Summary

Hands On Exercise 5: Adding a service demand by correlation

Previously, we added an exogenous service demand. That is, we explicitly specfied what the demand would be per year.

However, we may not know what the electricity demand is per year into the future. Instead, we may conclude that our electricity demand is a function of the GDP and population of a particular region.

To accommodate such scenarios, MUSE enables us to choose a regression functoin that estimates service demands from GDP and population, which may be more certain in your case. In this hands-on we find out how this can be done.

Learning objectives

* How to add a service demand by correlation

Introduction

For this work, we will use the default example, as before, from the MUSE repository.

The full scenario files for the default example can be found at the zenodo link below. https://zenodo.org/record/6092720#.YgvcMy-l1pQ

We recommend that you download these files and save them to a location convenient to you, as we will be amending these throughout this tutorial.

Similarly to before, we must amend the preset folder for this. However, we no longer require the Residential2020Consumption.csv and Residential2050Consumption.csv files. These files set the exogenous service demand for the residential sector.

We must replace these files, with the following files:

* A macrodrivers file. This contains the drivers of the service demand that we want to model. For this example, these will include GDP based on purchasing power parity (GDP PPP) and the population that we expect from 2010 to 2110.
* A regression parameters file. This file will set the function type we would like to use to predict the service demand and the respective parameters of this regression file per region.
* A timeslice share file. This file sets how the demand is shared between timeslice.

The example files for each of those just mentioned can be found in the zenodo link below. https://zenodo.org/record/6092720#.YgvcMy-l1pQ

Download these files and save them within the preset folder.

Next, we must amend our toml file to include our new way of calculating the preset service demand.

TOML file

Editting the TOML file to include this can be done relatively quickly if we know the variable names.

In the second bottom section of the toml file, you will see the following section:

[sectors.residential\_presets]

type = 'presets'

priority = 0

consumption\_path= "{path}/technodata/preset/\*Consumption.csv"

This enables us to run the model in exogenous mode, but now we would like to run the model from the files previously mentioned. This can be done by linking new variables to the new files, as follows:

[sectors.residential\_presets]

type = 'presets'

priority = 0

timeslice\_shares\_path = '{path}/technodata/preset/TimesliceSharepreset.csv'

macrodrivers\_path = '{path}/technodata/preset/Macrodrivers.csv'

regression\_path = '{path}/technodata/preset/regressionparameters.csv'

We’ve just linked the new files to MUSE.

Running and visualising our new results

Figure 5.1, below, shows the power sector over the future horizon. We can see a significantly higher installed capacity, as the demand has increased due to the correlation of GDP PPP and population.

Chart, bar chart

Description automatically generated

**Figure 5.1:** Visualisation of the power sector

Summary

In this hands-on we added a service demand by correlation. Specifically, GDP purchasing power parity and population. We saw that we could make inferences on how the demand will grow based on these using seperate files in MUSE.

Hands On Exercise 6: Adding an agent

Now we will learn how to add a new agent to our example.

Learning objectives

* How to add a new agent

Introduction

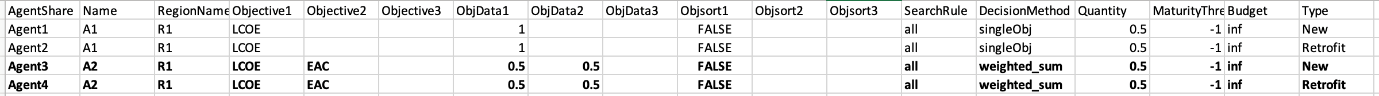
In this hands-on, we will add a new agent called A2. This agent will be slightly different to the other agents in the default example, in that it will make investments based upon a mixture of levelised cost of electricity (LCOE) and equivalent annual cost (EAC). These two objectives will be combined by calculating a weighted sum of the two when comparing potential investment options. We will give the LCOE a relative weight value of 1 and the EAC a relative weight value of 0.25.

We will edit the default example to add a new agent, which can be found from the following zenodo link: https://zenodo.org/record/6323453#.Yh-QWi-l1pQ

To achieve this, first, we must modify the Agents.csv file in the directory:

{muse\_install\_location}/src/muse/data/example/default/technodata/Agents.csv

To do this, we will add two new rows to the file. To simplify the process, we copy the data from the first two rows of agent A1, changing only the rows: AgentShare, Name, Objective1, Objective2, ObjData1, ObjData2, DecisionMethod and Quantity. The values we changed can be seen below. Notice how we edit the AgentShare column. This variable allows us to split the existing capacity between the two different agents. We will also need to edit the technodata file to define these new AgentShares.

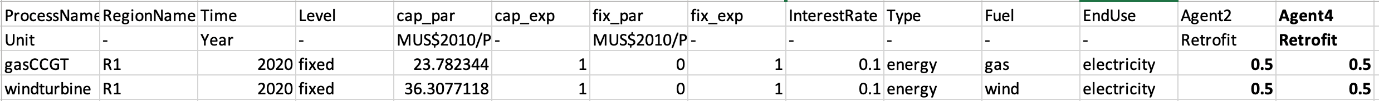


**Figure 6.1:** Updated technodata.

Also notice that we amend the Quantity column. The reason for this is that we want to specify that Agent A1 makes up 50% of the population, and A2 makes up the remaining 50% of the population.

We then edit all of the technodata files to split the existing capacity between the two agents by the proportions we like. As we now have two agents which take up 50% of the population each, we will split the existing capacity by 50% for each of the agents. Notice that we only require the columns Agent2 and Agent4 to define the retrofit agents.

The new technodata file for the power sector will look like the following (we have hidden the middle columns as they remain the same):



**Figure 6.2:** Edited power technodata file.

However, remember you will have to make the same changes for the residential and gas sectors!

We will now save this file and run the new simulation model using the following command in Anaconda prompt:

python -m muse settings.toml

Figure 6.3 shows us the results of these two agents. We can see a divergence between technologies invested in by the agents dependent on their objectives

Chart

Description automatically generated

**Figure 6.3:** Visualisation of the two different agents - a) agent = A1, b) agent = A2.

For all the files explored in this hands-on, please refer to the following link: https://zenodo.org/record/6323453#.Yh-QWi-l1pQ

Summary

In this hands-on we added a new agent which had different characteristics to the original agent and saw that this lead to a dramatic change in technologies invested in.

Hands On Exercise 7: Adding a region

Now we will learn how to add a new region to our example.

Learning objectives

* Learn how to add a new region

Introduction

The next step is to add a region which we will call R2, however, this could equally be called USA or India. These regions do not have any energy trade. This requires us to undertake a similar process as before of modifying the input simulation data. However, this time we will also have to change the settings.toml file to achieve this.

The process to change the settings.toml file is relatively simple. We just have to add our new region to the regions variable, in the 4th line of the settings.toml file, like so:

regions = ["R1", "R2"]

The process to change the input files, however, takes a bit more time. To achieve this, there must be data for each of the sectors for the new region. This, therefore, requires the modification of every input file.

Due to space constraints, we will not show you how to edit all of the files. However, you can access the modified files at the zenodo link below:

Effectively, for this example, we will copy and paste the results for each of the input files from region R1, and change the name of the region for the new rows to R2.

However, as we are increasing the demand by adding a region, as well as modifying the costs of technologies, it may be the case that a higher growth in technology is required. For example, there may be no possible solution to meet demand without increasing the windturbine maximum allowed limit. We will therefore increase the allowed limits for windturbine in region R2.

We have placed two examples as to how to edit the power sector below. Again, the edited data are highlighted in **bold**, with the original data in normal text.

The following file is the modified /technodata/power/CommIn.csv file:

Table

Description automatically generated

**Figure 7.1:** Updated CommIn.csv.

Whereas the following file is the modified /technodata/power/ExistingCapacity.csv file:

Table

Description automatically generated

**Figure 7.2:** Updated ExistingCapacity.csv.

Below is the reduced /technodata/power/technodata.csv file, showing the increased capacity for windturbine in R2. For this, we highlight only the elements we changed from the rows in R1. The rest of the elements are the same for R1 as they are for R2.

Again, we don’t show the entries for 2040, apart from the edited windturbine row, for the sake of brevity.

Graphical user interface, application, table

Description automatically generated

**Figure 7.3:** Updated Technodata.csv.

Now, go ahead and amend all of the other input files for each of the sectors, the Agents file and the input files BaseYearExport, BaseYearImport and Projections.csv by copying and pasting the rows from R1 and replacing the RegionName to R2 for the new rows. All of the edited input files can be seen at the zenodo link:

Again, we will run the results using the python -m pip muse settings.toml in anaconda prompt, and analyse the data using excel as follows:

Graphical user interface, application, table

Description automatically generated

**Figure 7.3:** Capacity visualisation for both regions in the power sector - a) Region = R1, b) Region = R2.

Summary

In this hands-on we added a new fictional region with the same characteristics for both of these regions. We see that the output of the two regions in the power sector are the same. This is because the characteristics in both regions are identical.