Hands-on exercise 9: Production constraints by timeslice

In some sectors it may be the case that a technology can only output a certain amount at a certain time. For instance, solar photovoltaics (PV) don't produce power in the dark, and thus their output is limited at night.

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

Minimum timeslice constraint

In this tutorial we will be amending the `default_timeslice` example.

To copy this model so you can edit the files, run:

```
python -m muse --model default_timeslice --copy PATH/TO/COPY/THE/MODEL/TO
```

You will see that, compared to the `default` example, this model has an additional `TechnodataTimeslices.csv` file in the `power` sector, which has the 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 `UtilizationFactor` column specifies the maximum utilization factor for the respective technologies in the respective timeslices, and the `MinimumServiceFactor` specifies the minimum service factor of a technology. The timeslice based columns, however, are dynamic and will match the levels as defined in the `toml` file.

We will modify the minimum service factor for `gasCCGT` in the `power` sector as follows.

ProcessName	RegionName	Time	month	day	hour	UtilizationFactor	MinimumServiceFactor
Unit	-	Year	-	-	-	-	-
gasCCGT	R1	2020	all-year	all-week	night	1	**0.2**
gasCCGT	R1	2020	all-year	all-week	morning	1	**0.4**
gasCCGT	R1	2020	all-year	all-week	afternoon	1	**0.6**
gasCCGT	R1	2020	all-year	all-week	early-peak	1	**0.4**
gasCCGT	R1	2020	all-year	all-week	late-peak	1	**0.8**
gasCCGT	R1	2020	all-year	all-week	evening	1	**1**
windturbine	R1	2020	all-year	all-week	night	1	0
windturbine	R1	2020	all-year	all-week	morning	1	0
windturbine	R1	2020	all-year	all-week	afternoon	1	0
windturbine	R1	2020	all-year	all-week	early-peak	1	0
windturbine	R1	2020	all-year	all-week	late-peak	1	0
windturbine	R1	2020	all-year	all-week	evening	1	0

Looking at the `settings.toml` file, you should see that the file has already been linked to the appropriate sector:

```
[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'
```

Notice the `technodata_timeslices` path in the bottom row.

The `default_timeslice` model also includes one additional output, which gives a detailed breakdown of commodity supply in the power sector:

```
[[sectors.power.outputs]]
filename = "{cwd}/{default_output_dir}/{Sector}_{Quantity}.csv"
sink = "aggregate"
quantity = "supply"
```

This will create a new file in the results folder called `Power_Supply.csv`, which will be important for the analysis below.

Once you've had a look at these files, run MUSE with the usual command:

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

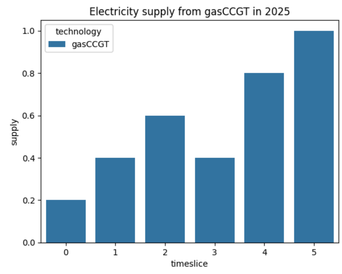
We will then visualise the output of the technologies in each timeslice:

In [1]:
```
import pandas as pd
import seaborn as sns
```

In [2]:
```
power_supply = pd.read_csv(
    "../tutorial-code/min-max-timeslice-constraints/1-min-constraint/Results/Power_Supply.csv"
)
gasCCGT_electricity_supply_2025 = (
    power_supply[
        (power_supply.technology == "gasCCGT")
        & (power_supply.commodity == "electricity")
        & (power_supply.year == 2025)
    ]
    .groupby(["timeslice", "technology"])
    .sum()
    .reset_index()
)

ax = sns.barplot(
    data=gasCCGT_electricity_supply_2025,
    x="timeslice",
    y="supply",
    hue="technology",
)
ax.set_title("Electricity supply from gasCCGT in 2025")
```

Out[2]:
```
Text(0.5, 1.0, 'Electricity supply from gasCCGT in 2025')
```



Here, we can see that the supply of electricity by `gasCCGT` in 2025 perfectly matches the values that we inputted for the `MinimumServiceFactor`. Given the capacity of `gasCCGT` in this year of 1 (see the `MCACapacity.csv` file), this suggests that this technology is operating at its minimum permitted level in all timeslices.

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 `UtilizationFactor`, as shown:

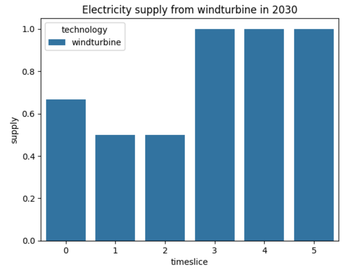
ProcessName	RegionName	Time	month	day	hour	UtilizationFactor	MinimumServiceFactor
Unit	-	Year	-	-	-	-	-
gasCCGT	R1	2020	all-year	all-week	night	1	0.2
gasCCGT	R1	2020	all-year	all-week	morning	1	0.4
gasCCGT	R1	2020	all-year	all-week	afternoon	1	0.6
gasCCGT	R1	2020	all-year	all-week	early-peak	1	0.4
gasCCGT	R1	2020	all-year	all-week	late-peak	1	0.8
gasCCGT	R1	2020	all-year	all-week	evening	1	1
windturbine	R1	2020	all-year	all-week	night	1	**0**
windturbine	R1	2020	all-year	all-week	morning	**0.5**	**0**
windturbine	R1	2020	all-year	all-week	afternoon	**0.5**	**0**
windturbine	R1	2020	all-year	all-week	early-peak	1	**0**
windturbine	R1	2020	all-year	all-week	late-peak	1	**0**
windturbine	R1	2020	all-year	all-week	evening	1	**0**

Once this has been saved, we can run the model again (`python -m muse settings.toml`), and visualise our results as before. We should hopefully see a reduction in the output of `windturbine` in the 2nd and 3rd timeslices:

In [3]:
```
power_supply = pd.read_csv(
    "../tutorial-code/min-max-timeslice-constraints/2-max-constraint/Results/Power_Supply.csv"
)
windturbine_electricity_supply_2030 = (
    power_supply[
        (power_supply.technology == "windturbine")
        & (power_supply.commodity == "electricity")
        & (power_supply.year == 2030)
    ]
    .groupby(["timeslice", "technology"])
    .sum()
    .reset_index()
)

ax = sns.barplot(
    data=windturbine_electricity_supply_2030,
    x="timeslice",
    y="supply",
    hue="technology",
)
ax.set_title("Electricity supply from windturbine in 2030")
```

Out[3]:
```
Text(0.5, 1.0, 'Electricity supply from windturbine in 2030')
```



As expected, we can see an enforced reduction in `windturbine` output in the 2nd (1) and 3rd (2) timeslices.

From the `MCACapacity.csv` file, we can see that the capacity of `windturbine` in this year is 1. Therefore, the results show that `windturbine` is operating at its maximum permitted level in all timeslices apart from the first, where it's operating below this level.

Summary

In this tutorial we've shown had to impose minimum and maximum constraints on the activity of technologies on a timeslice-basis. Not only will this impact the supply of comodities, but it may also influence investment decisions. You are encouraged to explore the implications of this on your own.