Alliance Mission (TEMOA) Onboarding

1. General onboarding
2. What are we trying to achieve?

We are building an open-source energy systems model on the TEMOA framework so that we can model the effects of scenarios and policies on Canada’s energy landscape into the future. This will help us inform better climate policy to find, for example, optimal pathways to achieve net-zero emissions in Canada by 2050. Read the [Alliance Mission activity summary](https://utoronto-my.sharepoint.com/:w:/r/personal/daniel_posen_utoronto_ca/Documents/Academia/File%20Sharing/Alliance%20Mission%20-%20TEMOA/Full%20team%20-%20TEMOA/NSERC%20Alliance%20Mission%20Activity%20Summary.docx?d=w9bab30fa32364bc2a14bdfa2da8026d5&csf=1&web=1&e=abkOW3) to learn more.

1. What is TEMOA?

TEMOA is a bottom-up, optimised, capacity planning energy systems model written in python. It models the flow and transformation of energy in all its forms from root sources (resource extraction, domestic production, and import) through to satisfaction of end-use demands. There are two key object types in a TEMOA model: commodities and processes.

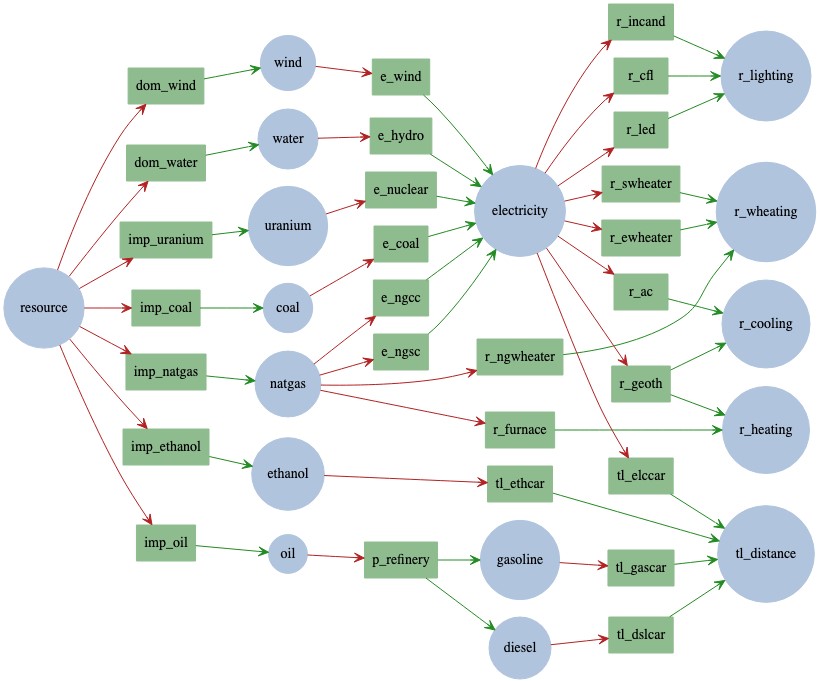
* 1. Commodities

Commodities are *stuff*, such as: kilotonnes of wood, PJ of electricity, PJ of demand for residential space heating, kilotonnes of emitted carbon dioxide. They are the things we count, measure, transform, and transport in an energy system. Commodities take three basic forms:

* Physical commodities (e.g., PJ of electricity, kilotonne of diesel). These are the physical carriers of energy that move about the energy system on trucks, in pipes, in wires.
* Demand commodities (e.g., PJ of residential space heating, person.km of car transport). These are the things we actually need from the energy system, as end-users, to go about our lives.
* Emission commodities (e.g., kilotonne of carbon dioxide). These are things that don’t necessarily carry useful energy but have meaningful impacts on the larger world.
  1. Processes

Processes are how we transform, or transport commodities from one form to another, from one place to another, from one time to another. Processes take input commodities and produce output commodities—sometimes a different commodity or sometimes the same commodity in a different place or time—and usually incur a cost for doing so. Processes typically constitute a particular technology created in a particular year (vintage). Some examples:

* An *oil refinery* is a process which takes *crude oil* as an input and produces *petrol*, *diesel*, *kerosene*, etc. as outputs.
* A *freight truck* is a process which takes *diesel* as an input and produces *transported tonne.km of freight* as an output.
* A *nuclear power plant* is a process which takes *uranium dioxide* as an input and produces *high voltage DC electricity* as an output.
* A *high voltage transmission line* is a process which takes *electricity here* and produces *electricity* *there*. A *Li-ion battery* takes *electricity* *now* and produces *electricity* *then*.



*An example of a TEMOA model network. Commodities are blue, transformed and transported by processes in green.*

* 1. Optimisation

The objective function for TEMOA is total system cost. When we run the TEMOA model, it determines all the process capacities and commodity flows that will satisfy end-use demands for the lowest total cost of the whole system. It will increase or decrease capacities and utilisations of technologies as needed to satisfy demand.

* 1. Constraints

These are the core *rules* of the model. They are how the model runs and how it represents the real world. TEMOA uses a linear solver such as CPLEX or Gurobi to try possible configurations of the system which minimise cost and satisfy all the constraint functions.

In a sense, we tell the model where it can go and where we want it to end up then it stumbles around blindly but efficiently until it gets there. We don’t give it any instructions on how to walk.

1. How do we build the model?

The TEMOA framework is designed to be flexible to different energy systems. The model isn’t programmed with rigorous behaviours matching the energy system; it’s not a simulator. Rather, we tell it the end-use demands we need to satisfy, we give it a list of technologies it can use to do that, tell it how efficient they are and how much they cost, then let it figure out the rest.

So, we need data.

Building our model means finding and aggregating data on all the end-uses, technologies and commodities that form Canada’s energy systems. We fill tables with this data and feed it to the model. This data is how we tell the model what Canada’s energy landscape looks like. It tells the model:

* How much of any technology we already have.
* How long technologies will last.
* What those technologies do, how efficiently, and for what price, with what emissions.
* What demands we need to satisfy, when, and how much.
* What technologies we can build, when, and how fast.
* Which scenarios and policies to adhere to.

We build the model by collecting data on Canada’s energy technologies and energy demands—by region, by sector, by time of year, time of day.

1. Next steps to learn more
   1. Poke through some databases

Look through some existing TEMOA databases (SQLite format) to see the kind of data that these models run on. Good examples to look through are:

1. The Open Energy Outlook’s [utopia test database](https://github.com/TemoaProject/temoa/blob/energysystem/data_files/temoa_utopia.sqlite)
2. Net Zero Atlantic’s [ACES model database](https://github.com/NetZeroAtlantic/ACES-Data/blob/main/ACES-7D-6S.sqlite)
3. One of the Open Energy Outlook’s [complete databases](https://github.com/TemoaProject/oeo/blob/master/US_9R_8D.sqlite)

To read these files you’ll need to install a SQL editor like [DBeaver](https://dbeaver.io/).

* 1. Read the Open Energy Outlook’s 2022 report

[This report](https://www.cmu.edu/energy/key-initiatives/open-energy-outlook/oeo-report-2022.html) provides a good idea of what our model might be able to accomplish when it’s finished. It helps to explain what the model is saying to us with its results.

* 1. Read the TEMOA documentation

Read through the [TEMOA documentation](https://temoacloud.com/temoaproject/Documentation.html) to get a better understanding of how it works. Remember that the core logic of the model is in the objective and constraint functions. If you have experience with programming, you may find these easier to read in iterative code form in [*temoa\_rules.py*](https://github.com/TemoaProject/temoa/blob/energysystem/temoa_model/temoa_rules.py).

The TEMOA model uses sparse sets to handle data and constraints. It isn’t important to understand what that means, except that constraints are only defined for technologies we specify those constraints for. When you see something like

Ar,p,t=Br,p,t+Cr,pAr,p,t=Br,p,t+Cr,p

It is saying optimise for configurations where A equals B plus C for every region, period, and technology, *where* a constraint value is listed in the database for that combination of region, period, and technology. Constraints are only defined where we specifically define them.

* 1. Install TEMOA and run it

The Open Energy Outlook has published a good [tutorial series on YouTube](https://www.youtube.com/playlist?list=PLTxJN2lIFcQl9BhObJ7Sqgm542o2uttfp) for installing TEMOA v2 and getting it running. Clone the [GitHub repo](https://github.com/TemoaProject/temoa/tree/energysystem) and run the model with the utopia database. Play with the numbers in the utopia database and see how it changes the output results. Below is a series of videos made by Jeff Hyink (the software developer updating Temoa to v3).

* [update to v3 on Vimeo](https://vimeo.com/896067842/471970c6a2?share=copy)
* [fake vintage magic utopia on Vimeo](https://vimeo.com/908891139/1bc4c89b44?share=copy)
* [update to v3 on Vimeo](https://vimeo.com/896067842/471970c6a2?share=copy)
* [quick build on Vimeo](https://vimeo.com/896071667/31f9da3f71?share=copy)
* [new cost table on Vimeo](https://vimeo.com/918941581/da05b29673?share=copy)
* [v3 myopic walkthrough on Vimeo](https://vimeo.com/914974757/799c382f74?share=copy)
* [svmga demo on Vimeo](https://vimeo.com/969143646/04a5f4db67?share=copy)

1. Project Resources

* Access to the OneDrive shared folder: [Full team - TEMOA](https://utoronto-my.sharepoint.com/:f:/r/personal/daniel_posen_utoronto_ca/Documents/Academia/File%20Sharing/Alliance%20Mission%20-%20TEMOA/Full%20team%20-%20TEMOA?csf=1&web=1&e=th8pdn)
* Access to CANOE Trello board: <https://trello.com/b/AP4z3f7U/alliance-mission-temoa>
* Access to the Zotero library: [Zotero library.url](https://utoronto-my.sharepoint.com/:u:/r/personal/daniel_posen_utoronto_ca/Documents/Academia/File%20Sharing/Alliance%20Mission%20-%20TEMOA/Full%20team%20-%20TEMOA/Zotero/Zotero%20library.url?csf=1&web=1&e=oz330b)

## For general questions about accessing links, resources, joining teams, or any other general inquiries, please reach out to [hoda.touma@utoronto.ca](mailto:hoda.touma@utoronto.ca)

## For any technical or project-related inquiries, please reach out to [david.turnbull1@ucalgary.ca](mailto:david.turnbull1@ucalgary.ca)

## Technical Onboarding

## Technology + TechnologyType

### The different types of technology

There are 4 types of technology in the CANOE database. These are resource technologies (r), production technologies (p), baseload production (pb) and storage production technologies (ps).

* Resource technologies are technologies involved in bringing resources in. They are almost exclusively import technologies that bring in fuels/other commodities from ethos (the limitless void we use to supply limitless fuels as needed).
* Production technologies use fuels and other commodities to produce other commodities to eventually reach a demand. These are the technologies we are typically changing and experimenting with.
* Baseload production technologies are technologies that are used to achieve baseload production of certain commodities. This is mostly used in the electricity sector where we need to meet a baseload amount of electricity production. Baseload-tagged technologies are forced to operate at the same capacity in all hours.
* Storage production technologies are analogous with batteries and other energy storage types (hydrogen storage etc.), they can be used to store commodities for later use.

### Naming Convention for Technologies

We cannot repeat names in the technology table as that is going to ensure confusion. Together as a group, we agreed upon the following naming convention:

|  |  |
| --- | --- |
| R\_IMP\_NG | I\_EAF\_SCRAP\_NEW |
| * The first letter corresponds with the sector, so R is residential. * The second phrase (IMP) refers to import. Typically, the second phrase describes the technology * The final phrase (NG) refers to natural gas. The third phrase delineates the technology from other similar technologies. * Altogether, we see this technology imports natural gas. * All uppercase | * Using the previous approach, we can determine the sector to be industry (I). * Within the industrial sector, this technology represents electroarc furnace (EAF) in the steel industry. * From the different electroarc furnaces, the next phrase details a scrap only furnace (SCRAP) * A NEW phrase allows us to force certain technologies to be implemented (more on this later) |

### Technology table binary options

The right side of the technology tables contains a number of binary options we can use to improve/simplify the technologies in the model. To use a particular feature, we use a 1 in the table under the appropriate column. A 0 or no value at all is considered as an off value for the binary option.

Unlimited Capacity – Technologies where we aren’t concerned about the specific capacity of said technology. Typically used for import technologies.

Annual – Used for when a technology is meeting a demand however it does not vary over time via the demandspecificdistribution table. In our current state, industry that aren’t being focused on (other manufacturing for example) has a no DSD and can be assumed to have demand equal throughout the year, hence tech annual is reasonable to use and would simplify the model.

Reserve – The "reserve" flag identifies technologies whose output contributes to the calculation of total electricity supply, which is required for constraints such as the planning reserve margin or renewable portfolio standards. As a result, all technologies that generate, store, or transfer electricity between regions should be marked as "reserve."

Curtail – This flag designates technologies whose output can be curtailed, typically applied to variable renewables like wind, solar, and run-of-river hydro.

Retire –  This flag enables economic retirement for technologies. If not flagged, retirement is based on the age of the technology.

Flex – Some technologies have multiple outputs, and their relative shares may be constrained by the TechOutputSplit constraint. However, Temoa only produces output commodities when there is a corresponding demand. In cases where demands for a technology's outputs are not in proportion to its output shares, infeasibilities can arise. The "flex" flag allows the technology to overproduce some outputs and discard the excess at no cost in order to meet the other demands. An example is a refinery with fixed output shares for jet fuel, diesel, propane, etc. If demand for jet fuel increases while propane demand decreases, marking the refinery as "flex" allows it to meet the increased jet fuel demand while discarding excess propane.

Variable – This is likely to be removed in the future, but currently it allows for us to use the techinputsplitaverage table, and that’s it.

Exchange – This is used for exchanging commodities between regions, it is worth keeping in mind that the transmission lines for these technologies are only 1 way, therefore we need 2 efficiencies to represent 1 technology as we can send commodities both ways. In the efficiency table, we state the 2 regions being used as AB-SK. When taking into account investment costs, we typically half the cost (unless we know the price breakdown as something different). For example, a electricity transmission line between Alberta and BC cost 20 M$, we would represent this as AB-BC 10 M$ and BC-AB 10 M$.

## Commodity + CommodityType

### The different types of Commodities

There are 4 different types of commodities in the CANOE database. These are source commodities (s), physical commodities (p), emission commodities (e) and demand commodities (d).

* Source commodities are vital commodities in CANOE, without a source commodity the model will give error warnings. We use the term ethos for source commodities but you could name them anything. Source commodities act as an endless resource where we can import as much as we need.
* Physical commodities are the commodities we are using within the model. Anything used by a technology is considered a physical commodity. Examples of physical commodities includes electricity, gasoline, coke breeze and natural gas.
* Emission commodities are the models way of defining emissions. We are able to be as detailed as we want but the minimum we want to include is carbon dioxide equivalents.
* Demand commodities are similar to source commodities as they too are vital to the model. Demand commodities can be similar to physical commodities but are what the model uses to calculate everything else. Examples of demand commodities include rolled steel, residential cooling, distance travelled using light duty vehicles.

### Naming Convention for Commodities

Similar to technologies, naming commodities is important to avoid confusion and to aid accessibility. Together as a group, we agreed upon the following naming convention:

|  |  |
| --- | --- |
| C\_ethos | T\_gsl\_elc\_phev20 |
| * The first letter corresponds with the sector, so C is commercial. * The phrase ethos is used to describe where all commodities not made endogenously are imported from. * The first letter is uppercase whilst the rest is lowercase. | * Using the previous approach, we can determine the sector to be transportation (T) * Within the transportation sector, this commodity involves the use of gasoline and electricity (gsl\_elc) * The last section details what the commodity is used for/in, in this case phev20 vehicles. |

## TimeOfDay + TimeSeason

### Testing Approach

* The minimum we need to run a model is 2 hours in TimeOfDay and 2 days in TimeSeason, however we can implement more and describe TimeSeason as seasons, or months instead if we so choose.
* We typically use this model to debug sectors and ensure a fast model run to confirm the sector is ready to run. Below we will see 2 different approaches, one we have used and the other is used in Utopia.

|  |  |  |  |
| --- | --- | --- | --- |
| TimeOfDay  1 H1  2 H2 | TimeSeason  1 D1  2D2 | TimeOfDay  1 Day  2 Night | TimeSeason  1inter  2 summers  3 winters |

### Standard Approach

* We represent all 8760 hours of the year using both TimeOfDay and TimeSeason. There will be 24 hours in TimeOfDay and 365 days in TimeSeason. If we run the model as is, it will crash and never complete.
* Instead, we use our representative days script (detailed later) which will filter the values into a set value (8-12) days and make the model more computationally efficient to run. We include all the days previously as it makes DemandSpecificDistribution (later section) easier.

## TimePeriod + TimePeriodType

* There are two types of time periods, existing (e) and future (f). Existing time periods are used mostly in the existing capacity table and for vintages of technologies from the existing capacity table in other tables. The future time periods are the periods we are working with and optimizing for.
* The minimum time periods we require are the future periods we are optimizing for, in our case 2021, 2025, 2030, 2035, 2040, 2045 and 2050. When we see a period column in another table, it corresponds to the future period. When we see a vintage column in a table, it can include the existing values as well as future values (indicating the technology was built in a future time period).

## TimeSegmentFraction

* Also known as SegFrac in earlier versions of the Temoa model. It is the associated weight of the TimeOfDay and TimeSeason as seen as a year. If there are only 4 values, 2 TimeOfDay and 2 TimeSeason, we can assume that they all have the same weight which would be 0.25, as the sum of the TimeSegmentFraction must equal 1.
* When using seasons and time of days similar to utopia (day, night, winter, summer, inter), some weights might be larger than other as inter represents twice the amount of days (accounting for spring and fall) and day represents more hours than night (16 to 8 hours). With this in mind we can assign the TimeSegmentFraction shown below.

|  |  |  |
| --- | --- | --- |
| TimeSeason | TimeOfDay | SegFrac |
| Winter | Day | 0.1667 |
| Winter | Night | 0.0833 |
| Summer | Day | 0.1667 |
| Summer | Night | 0.0833 |
| Inter | Day | 0.333 |
| Inter | Night | 0.1667 |

* When using our standard conditions, the SegFrac will automatically be calculated according to the representative days procedure we are using, therefore before using the rep days program, the value for each time segment is 1/8760 or 0.000114.

## MetaData

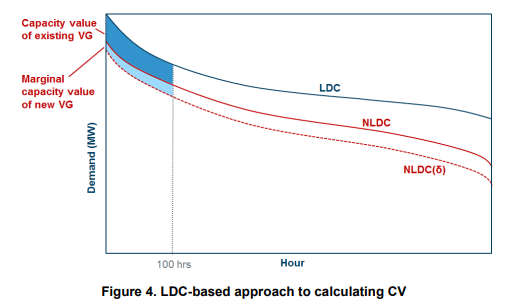
* The first entry is the myopic base year, it sets the base year of the discounted system cost when running in myopic mode, it should be set to the first planning period.
* The second two entry are the versions of the database being used for internal bookkeeping. DB major refers to the major version number, i.e Temoa 3, while DB\_MINOR refers to the minor version number, i.e Temoa 3.1 or 3.0.

## MetaDataReal

* Both of these are important, the global discount rate is key to how the model runs. The typical ranges for this value are between 2-5%, ideally 2%.
* Default loan rate, the weighted average cost of capital (WACC), is for when we don’t have the values for specific loan rates when used in the loan rate table, we can default to this loan rate.

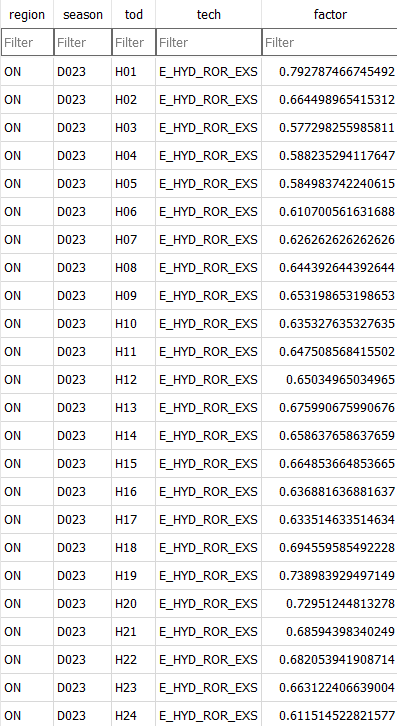
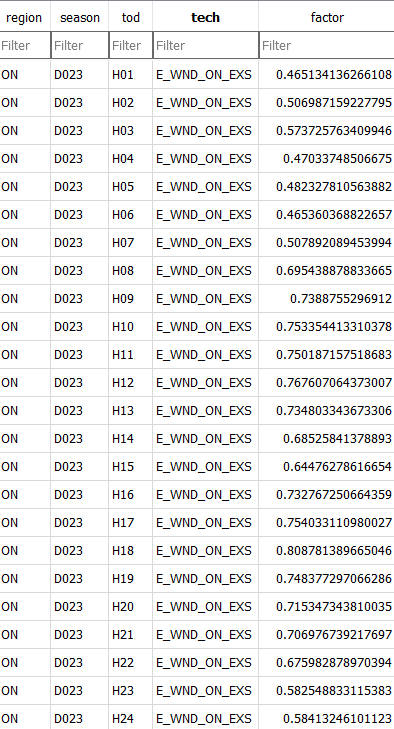
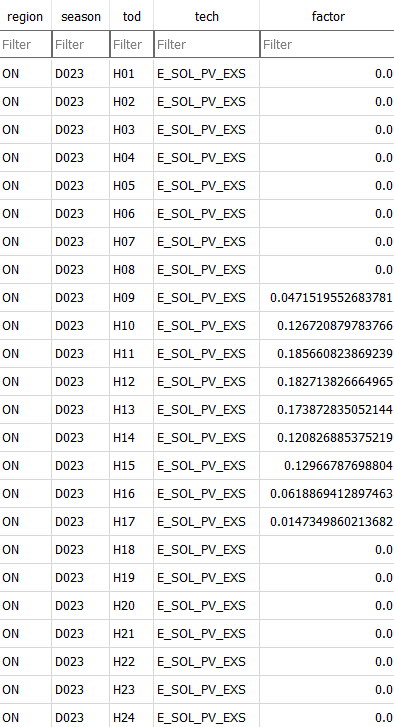
## CapacityCredit

* CapacityCredit represents the fraction of total installed capacity of a process that can be relied upon during the time slice in which peak electricity demand occurs. There are a couple of methods we use to estimate this value. The first method involves getting the forecasted capability at normal summer peak divided by total installed capacity. The second method is the NREL ReEDS method.
* The NREL ReEDS method considers the reduction of net electricity load at the 100 highest load hours. In the simplest instance, adding a single renewable generator to a system, the hourly generation of that new generator is subtracted from hourly load. This gives two time series, the original hourly load and the new hourly net load. Both time series are sorted high to load (a duration curve) and the average hourly difference between the two at the highest hour is the capacity value of that generator. The capacity credit is then the capacity value divided by the nameplate capacity of the generator. This process can be iteratively performed again for each new generator added.

****

## CapacityFactorTech

* Parameter that specifies how much activity much be given out per its capacity. It is the percentage of its maximum possible output for a particular timestep, i.e., the specific season and tod timestep. This is activity is split between useful activity and curtailment (lost energy). Most, if not all, technologies in the capacity factor tech table should have a 1 in the curtailment field of the technology table.



* Above are three examples of capacity factor tech for 3 different technologies. For solar systems, we can see 0 values which makes sense as they don’t operate at night. For the wind system, we see high variability throughout the day which corresponds with wind rates. Hydro is slightly different as while there is variability, it is less than that of wind.

## CapacityFactorProcess

* Capacity factor process is almost identical to capacity factor tech, however it is also indexed by vintage. This allows us to change capacity factors for technologies per vintage as we can expect some technologies to become more effective over time.
* We can include both values for the same technology in both CF process and technology, however doing so is redundant as only one value can be used in the model, the value that supersedes the other is CF process.

## CapacityToActivity

* This table is in essence a conversion table allowing us to convert capacity to activity in different units, for example if we have thousands of cars as capacity and billions of passenger kilometres as activity, there is a unit mismatch. Therefore, we have capacity to activity to alleviate this issue.

## CostEmission

* This table enables us to place a value associated with emitting pollutants. Inside Canada we are instituting a carbon tax and this table allows us to implement it into the model easier. To ensure the values align, check the EmissionActivity table and check the units to make sure they are on the same scale, i.e., kTonne/PJ and kTonne.
* The utility in this table is the fact we can change the carbon pricing per time slice so we can use current and potential future policy to guide carbon pricing to determine the effect on the model. Below is an example of several records in the table. They illustrate that we can alter the carbon tax as we proceed through the time slices in the model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| region | period | emis\_comm | cost | unit | notes | reference |
| ON | 2025 | co2eq | 0.16 | M$/kTonne | Scaled from $/Tonne to M$/kTonne | example |
| ON | 2030 | co2eq | 0.17 | M$/kTonne | Scaled from $/Tonne to M$/kTonne | example |

## CostFixed + CostVariable + CostInvest

* In Temoa, OPEX are split into the costfixed and costvariable. The costfixed table is used to report maintenance costs for technologies. The costvariable table refers to the variable cost of fuels being used by the technology. Typically, the values in the costfixed table do not change throughout the periods (this has potential to change in the future with a better implementation regarding learning rate). As expected though, the values in the costvariable table are expected to change throughout the periods (unless it is an uncommon commodity with little data).
* Costvariable is highly important to the model, we can remove all other costs if necessary but costvariable is pivotal to model performance.
* The values used inside these tables are cost per a unit; for costfixed, the value is dollar per unit capacity while costvariable is cost per unit (such as PJ, MWatts, MTonnes).
* CAPEX is represented in the costinvest table. We don’t typically change the investment costs over different periods but it is doable. Similar to the costfixed table, the cost value is in cost per unit capacity

## Demand

* Demand is the most crucial input into the model, without demands nothing will happen. Demands as specified in the commodities table as a demand commodity and they are assigned a specific quantity that must be made by the model.
* Demand can have multiple different units (PJ, MTonnes etc.), as long as the unit interconversions throughout the model are correct, the model will be correct.
* Currently, we forecast general demands, say for a black box interpretation for industry, by multiplying the known value by the increase in population change. We do this until we have values for all future time periods.
* When possible, we use demand forecasts from official sources (i.e., CER)
* An alternative approach intended to be used is to use time series forecasting with historical data to anticipate the changes going forward.

## DemandSpecificDistribution

* The demandspecificdistribution (also known as dsd) table, enables us to distribute the demand for commodities throughout the year at uneven pacing. Without using the dsd table, it is assumed the demand is required constantly throughout the year via the same distribution as the timesegmentfragment table (discussed later)
* Rashid or Davey can explain how they got their values

## Efficiency

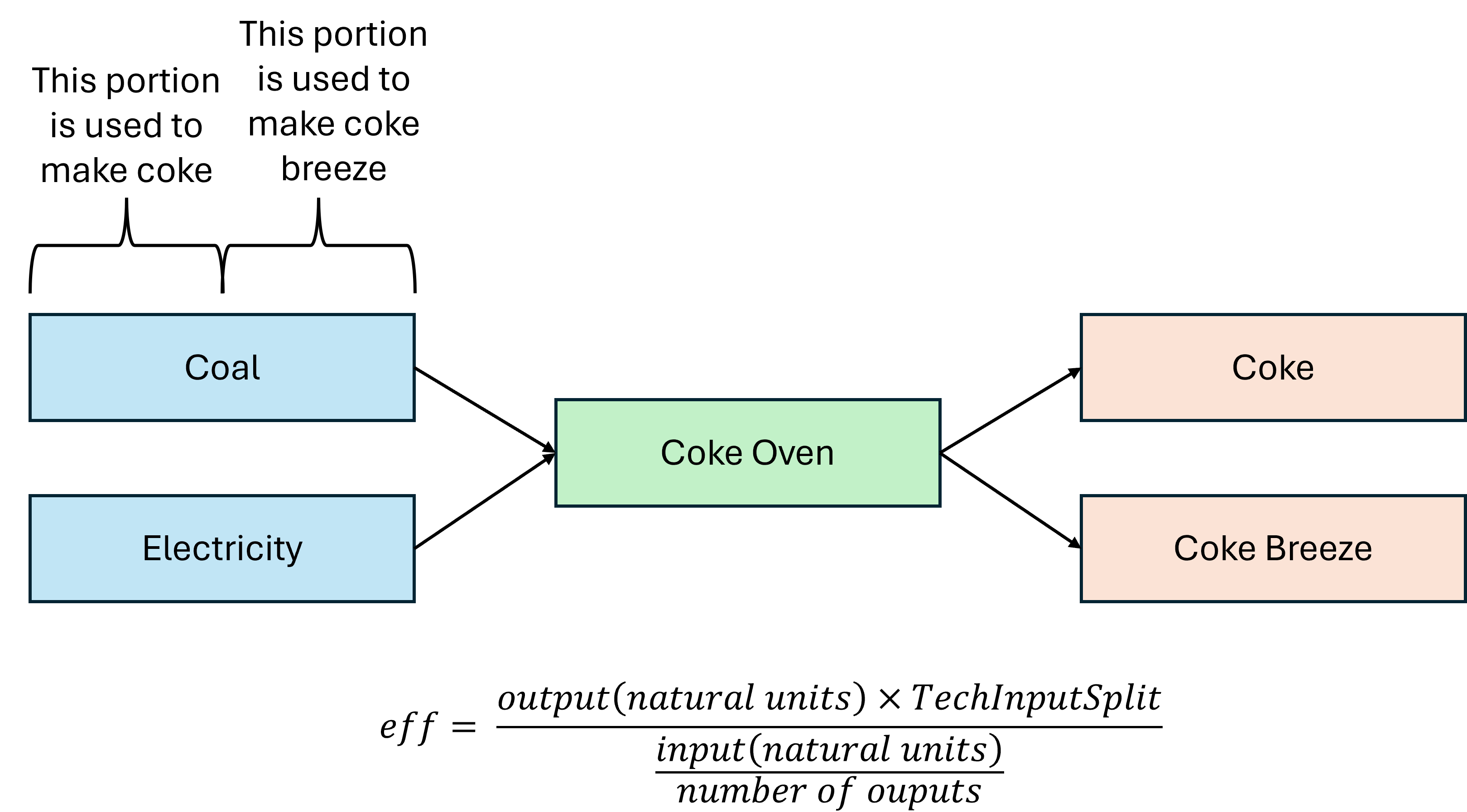
* This table is similarly as important as the demand table. It is required for technologies to convert commodities into other commodities, not including a value for a technology, results in that technology not being considered in the model.
* The efficiency calculation can be perceived differently depending on if there are multiple inputs/outputs. The simplest efficiency calculation is when there is only one input and one output, the calculation is just output divided input. We use mass and energy balanced to know these values.



* The second equation is when we have 2 inputs and one output, we adapt the equation to include the techinputsplit (discussed later), this results in the efficiency for both inputs being the same. This only works when the techinputsplit equals one for the technology.



* The final equation, and the most complicated is when we have multiple inputs and multiple outputs. The equation is similar to when we have multiple inputs, however we have to take into account we making more outputs therefore we divide the input by the number of outputs.
* When we don’t do this, our inputs are off by a factor of the number of outputs since the model is importing more commodity to meet demand. The diagram below shows the rationale, the equation in that diagram is specific for each output, but not for each input. If we had 3 different outputs, we would expect to see 3 different efficiencies.
* The equation below can be considered the master equation for efficiency as it works for all instances, when we only have 1 input and 1 output, the techinputsplit and number of outputs equals 1. When we have multiple inputs and 1 output, the techinputsplit is used however the number of outputs is 1 meaning the result would be the same as equation 2 in the figure above.



## EmissionActivity

* The emissionactivity table allows us to determine how much emissions from different processes/technologies. The standard units for this table is kilotonnes per PJ of energy input.
* There are two approaches that can be used in this table, either we assign emissions to the import of fuels under the assumption that any fuel imported with be completely consumed, or we can assign emissions to a technology.
* The columns of interest in this table are the emis\_comm, input\_comm and technology. In approach 1, the input\_comm will be ethos and the technology will involve IMP (import) of some kind. In approach 2, the input\_comm will be a fuel of some kind and the technology is a process that consumes the fuel. The emis\_comm can be whatever emission we deem as worthwhile to study, such as carbon dioxide equivalent.

## EmissionLimit

* This sets an emission limit of specified emission commodities for a particular region for a particular time period. It is worth noting that we can group regions together by writing region + region + region in the region section of the table which simplifies this table considerably. An example of this would be ‘ON+AB+MB’.

## ExistingCapacity

* The existingcapacity table is used to describe preexisting capacity of technologies. Without including any existing capacity, the model is required to build everything in the first time period, which is obviously unrealistic.
* In this table, we specify the amount of capacity we can already meet using this particular technology. We can have multiples of the same technology to represent different sites. We can also specify vintage which works in conjunction with lifetimetech (discussed later) to retire existing capacity if needed.

## GrowthRateMax + GrowthRateSeed

* GrowthRateMax is when we want to limit the growth of a new technology, we can set a growth rate that is aligned with reality. The values are percentage based therefore we should use 0.1 for 10% growth rate.
* GrowthRateSeed is a starting point for when we introduce a new technology that doesn’t have any existing capacity. If we don’t use growthrateseed, we apply a growth rate of 10% to 0 as there is no existing capacity, 0+10% of 0 is still 0 so we get a problem. We can set a seed to initial create some amount of the technology and then use the growth rate we determined before.

## LifetimeTech + LifetimeProcess

* The lifetimetech table define the lifetime of technologies in years. There are 2 ways to approach assigning lifetimes, either use a median lifetime based on existing data or use an artificial lifetime that enables certain technologies to be retired at certain time slices.
* Without a value being inputted the default value is 50 years.
* Lifetime process is again similar to lifetime tech but with particular vintages.

## LoanLifetimeTech + LoanRate

* LoanLifetimeTech is the lifetime for the loan needed for capital costs (costinvest) for technologies. If we imagine the investment to be akin to a house, the loanlifetimetech would be 25-30 years.
* Loan rate is the loan rate for specified technologies, if we don’t use a specified loan rate, the default loan rate used in metadatareal will be used instead.

## MaxActivity + MaxActivityGroup + MaxActivityShare

* This table sets a constraint on the max activity for a specific technology or technology group. When running the model, sometimes the decisions the model makes does not reflect the real world, maybe by choosing to maximize certain technologies early then we would consider. This is when we can use MaxActivity to limit the unit of certain technologies.
* This often works in conjunction with MaxCapacity (discussed later), as we can limit the amount of capacity and then lower the activity when compared to the activity. For example, we can limit capacity of a natural gas combined cycle plant to 1 PJ of electricity out but constrain the activity to 0.5 PJ.

## MaxAnnualCapacityFactor + MinAnnualCapacityFactor

* These tables set an upper and lower limit of the amount of activity a specified technology can have based on its capacity for an entire time period. It should be known that this is a percentage value.
* An example for this could be a baseload nuclear powerplant. Nuclear powerplants are on all the time, therefore there Maxannualcapacityfactor is high, around 80% or 0.8 in the table. Natural gas CC plants are mostly used to offset the demand peak, they are on only a few hours of the day. This leads to a low max annual capacity factor, around 15% or 0.15.

## MaxCapacity + MaxCapacityShare

* Similar to MaxActivity, MaxCapacity sets a limit for the maximum amount of capacity that can be built for specified tachnologies. This allows us to shape the model with decisions we know will occur.
* For example, the model might build a large amount of one technology when we have reports that the industry being studied will actually utilize a different technology at a set capacity. We can force the model to build a set capacity of said technology and allow it to make decisions for the remaining demand.

## MaxNewCapacity + MaxNewCapacityGroup + MaxNewCapacityShare + Min versions

* MaxNewCapacityGroup and MaxNewCapacityShare work with technology groups (more on that later), what we can do with these tables is assign a max new capacity for these technology groups we made as well as the share of the new capacity for each technology in the group. For example, we could make a group for light duty vehicles containing gasoline, diesel and electric vehicles, assigning a new capacity for the group ensures that the sum of all new vehicles in this class won’t go over that value.
* MaxNewCapacity is similar however specifies only a single technology.
* The min tables for these are the inverse where we can choose the minimum new capacity.

## MaxResource

* This sets a limit on the amount of a particular resource we can use. For example, we can set a limit on something like coal, we can estimate how much we have left, and set that limit so once we reach that amount used, technologies that use that commodity will be unable to continue without an alternative.

## MaxSeasonalActivity + MinSeasonalActivity

* Sets an upper and lower bound for the amount of activity a specified technology can have based on its capacity for an season, in our model these are representative days. This allows us to get a daily energy budget from resources such as hydro.

## MinActivity + MinActivityGroup + MinActivityShare

* This table is the inverse of the MaxActivity table, by using both together, we can force the model to use a set amount of certain technologies or a set range of certain technologies.

## MinCapacity + MinCapacityGroup + MinCapacityShare

* This table is the inverse of the MaxCapacity table, by using both together, we can force the model to build a set amount of certain technologies or a set range of certain technologies.

## PlanningReserveMargin

* Assigns an amount of extra electricity needed to be generated to meet planning reserve margin for each region. An example being that Alberta has a PRM of 25%, using that in the model results in an increase in electricity generated at all times to ensure demand can always be met in case of demand shock.

## RampDown + RampUp

* Rampdown and rampup sets limits for variability of production. Some technologies like a nuclear powerplant cannot change output rapidly so they may have a ramp rate of 0.05 meaning it can only fluctuate 5% production activity over 2 hours. While a technology like a natural gas CC plant can be turned off and on and whim leading to a fluctuation of 100% over an hour

## RPSRequirement

* RPSrequirement uses a renewable energy tech group and sets an amount of electricity that must be generated using this group. This is in line with several provinces, such as Newfoundland, having requirements on how much electricity must be made through renewables.

## StorageDuration

* This table sets how long storage technologies can stored their commodities for. Some technologies only require short-term storage, such as batteries, however other are given a years worth of storage, typically chemical fuels such as hydrogen.
* Short-term storage allows the model to utilize them as a capacity credit to reduce the peak demand of electricity on the model.
* Long-term storage allows the model to keep certain fuels on hand meaning they are not needed to be made as and when they are needed which does not resemble the real world.

## StorageInit

* This table sets an initial storage amount for specified technologies to be used in the model. This parameter is not currently supported. A value can be set in the table but it will not be built into the model and so the StorageInit value will be optimised instead.
* **BE WARNED:** Using this constraint, will result in a non-optimal model as the beginning and end-state of the storage technology must be the same so the model will force the initial amount to be present at the end of the model, which results in a non-optimal solution

## TechGroup + TechGroupMember

* In techgroup, we create new arbitrary groups we wish to use. Techgroupmember allows us to assign particular technologies to that group.
* An example would be to create a group for light duty vehicles, then we could assign LDVs using gasoline, diesel and electricity into this group. This would allow us to use specific tables that simplifies the model.

## TechInputSplit

* This tables relates to the split in commodities being used by a technology. The sum of commodity split cannot be more than 1 (100%) however it can be lower and the model will make up for that in a cost-effective way.
* We can change techinputsplits over time periods to change more general technologies, an example is shown below. What we can see is that the generic smelting technology becomes more electrified as time goes on with a reduction in carbon emitting fuels.

|  |  |
| --- | --- |
| I\_SMELTING  2025  Coal 0.25  Electricity 0.3  Natural Gas 0.15  Oil 0.3 | I\_SMELTING  2030  Coal 0.15  Electricity 0.5  Natural Gas 0.25  Oil 0.1 |

## TechInputSplitAverage

* This is a more flexible version of tech input split where it sets the amount of commodities to be used throughout the year. In some timesteps it might use entirely one commodity and in other it might use a mix of the others. By the end of the time period, it will match the percentages used in the table. This table is not used as much now as RPSRequirement table has superseded its use.

## TechOutputSplit

1. Similar to techinputsplit, this table determines the percentage of the output from a technology. Unlike techinputsplit, it is unwise to assume the values will change over time.

# Standard operating procedure for getting started (Windows)

1. Create a git clone of the canoe temoa repository ([CANOE-main/temoa: Tools for Energy Model Optimization and Analysis (github.com)](https://github.com/CANOE-main/temoa)). If you don’t know how to do that look at bullet points a
   1. First, ensure that you have git installed on your computer, you do this by going to Powershell in your apps and using the code [winget install --id Git.Git]
   2. Once git is installed, close powershell and use the command line also known as CMD. In there copy this code [https://github.com/CANOE-main/temoa.git]. It will create a folder containing all the needed files to run the model.
   3. To update the CANOE-temoa version, use the CMD app, type [cd temoa] to change the folder you are in to the needed folder and then type [git pull] which will auto-update the needed files.
2. Install anaconda [[Download Anaconda Distribution | Anaconda](https://www.anaconda.com/download/)] and make sure it is working. If you already have anaconda/python, ensure that you have at least python 3.11 otherwise there will be issues. Contact David Turnbull ([David.turnbull1@ucalgary.ca](mailto:David.turnbull1@ucalgary.ca)) and he will help.
3. Once anaconda is installed, open anaconda and open powershell prompt. In their, use the following code to create a virtual environment [python3.12 -m venv venv], then use this code to activate the environment [venv\Scripts\activate].
4. Now you are in the virtual environment, use the code [pip install -r requirements.txt] to download the appropriate packages you need for temoa to run.
   1. There is an inherited issue in one of the packages that only affects windows, contact [David.turnbull1@ucalgary.ca](mailto:David.turnbull1@ucalgary.ca) for the solution, it takes 5 minutes to fix but hard to explain via text.
5. You are now ready to use Temoa!

# Standard operating procedure for running Temoa (Windows)

1. First thing is to make sure you have a useable database to work with, as a standard databases are saved .sql files as they are more memory efficient however the model only works with .sqlite files. If you have a sqlite file then continue to step 2, otherwise continue with the steps below.
   1. Using the powershell prompt on anaconda, use the cd command to navigate to the data files folder, this would look like [cd temoa], [cd data\_files], [cd example\_dbs].
   2. Once there, use this code [sqlite3 utopia.sqlite < utopia.sql] to create a new sqlite file from an existing sql file.
2. Open the my\_configs folder and ensure that settings you are choosing are appropriate. For a beginners run with utopia (the OEO test model), select the solver to be appsi\_highs (a free, available solver). In future runs, it is recommended to use gurobi as the solver but this requires installing it separately from [The Leader in Decision Intelligence Technology - Gurobi Optimization](https://www.gurobi.com/#:~:text=Gurobi%20supports%20the%20teaching%20and%20use)
3. Once we have a .sqlite file and the config file is set up, we can run the model. Using the powershell prompt from anaconda, make sure the folder you are in is the temoa folder at the highest level, (note, if you need to go backwards in folders, use [cd ..]. Now we use the following code to run the model [python main.py --config data\_files/my\_configs/config\_sample.toml], this should run the utopia model and give commodity maps that can be found in the output files folders.
4. Congratulations you have just completed your first run!