Rumi

An open-source energy systems modelling platform developed by Prayas (Energy Group)

Common inputs documentation

Table of Contents

1 Introduction

Rumi models are typically built to estimate energy demand and/or energy supply for some period over some geography. In order to do this, some inputs are expected which tie the demand and supply sides together, i.e. are common inputs that are relevant to both the demand and supply components of Rumi. These common inputs are provided in the `Common/Parameters` directory under the scenario root directory (or, alternatively, `Global Data/Common/Parameters` if the information is common across many scenarios). This directory is referred to as the `Root` directory henceforth.

All inputs to Rumi are in the form of comma-separated value (csv) files with a certain structure. Most input files to Rumi have header rows with defined column names, but some do not. For any input file that has a header row with named column headers, the order of the columns is immaterial. But if there is no header row, then the order of columns is important and should be exactly as documented.

It is the modeller's responsibility to ensure that the various inputs (common, demand and supply) together represent a valid abstraction of the envisaged energy system. This document describes all the common inputs that need to be provided to the Rumi platform.

2 Time granularity

Rumi allows time specification at four levels – years, seasons, day-types and day-slices. These are described below. Time specifications below the year level are optional. If specifications at a level are given, then specifications at all higher levels must be given (e.g. you can't have day-slices without day-types or seasons). If specifications are absent below a certain level, it's assumed that that level of disaggregation is not required for the model and hence cannot be used in later specifications.

2.1 ModelPeriod.csv

Directory: `Root`

Columns: `StartYear, EndYear`

This file specifies the start and end year of the model – i.e. the period which is modelled. The coarsest time granularity at which Rumi models work are annual – hence all intermediate years from the start to the end year (both inclusive) are considered part of the modelled period. The file has just one data row with values for the two columns representing the start and end years (both inclusive) of the model. The modelling period supported in Rumi is between the years 2000 and 2100.

Example[1]:

```
2020, 2050
```

[1] All examples provided in these documentation files are intended to be purely for illustrative purposes and do not necessarily present 'realistic' values. The examples may also not be mutually consistent.

2.2 Seasons.csv

Directory: `Root`

Columns: `Season, StartMonth, StartDate`

This file conveys information about the various seasons that are being modelled. It consists of a set of rows, with each row indicating the name of the season, the starting month of the season and the starting date of the season. Seasons are considered a separate category in Rumi because energy demand (and supply) is often seasonal – hence this makes modelling such seasonal variations easier. The start month is an integer between 1 and 12, representing the months from January to December, while the start date is an integer between 1 and 31, representing a valid date within the month[2].

This file is interpreted as follows. The seasons are implicitly assumed to be ordered in the sequence given. Thus, the first season of the year (and hence the year itself), begins on the first date of the first month given, and runs up to one day before the beginning of the second season. The last season is assumed to extend from the date and month in the last row, and continues up to one day before the first day of the year as specified in the first row. If there is an inconsistency in these definitions, an error will be issued.

Example:

```
Quarter1, 1, 1
Quarter2, 4, 1
Quarter3, 7, 1
Quarter4, 10, 1
```

2.3 DayTypes.csv

Directory: `Root`

Columns: `DayType, Weight`

This file lists the types of 'typical' days in a given season, since all days in a season may not be identical from a modelling point of view. The day types are given as a series of day-type names with associated weights, so that the weights add up to 1. The weight of a day-type is a value between 0 and 1, and represents the relative share of this day-type among all the day-types in the season. There is no implicit order assumed between the day types, and it is assumed that they can interleave and occur multiple times in a season[3].

[2] Note that February is assumed to have only 28 days – i.e. leap years are not currently supported.

[3] Note that if more than one day-type is specified, then the energy storage technologies defined must have their reset periodicity as DAILY. This is because any coarser reset periodicity requires each day to be modelled separately and, in the presence of multiple day-types that may interleave and occur multiple times in a season, that is not possible. This is described further in the documentation of supply inputs.

Examples:

`SINGLEDAYTYPE, 1` if just a single day type is modelled or

```
WEEKDAY, 0.71
WEEKEND, 0.29
```
if weekdays and weekends are modelled separately

2.4 DaySlices.csv

Directory: `Root`

Columns: `DaySlice, StartHour`

This file provides the number of different slices a single day can be broken into. Note that this division is common to all day-types. Like the season specification, it has a list of day-slice names along with associated starting hours, and implicitly assumes that these are given in the correct order. The start hour can be any integer between 0 to 23 (both inclusive), and represents the start hour for that slice. A day-slice lasts from its specified start hour until the start hour of the next day-slice. Since the day-slices repeat every day, the last day-slice specified lasts until the start hour of the first day-slice.

Examples:

`ENTIREDAY, 0` if the entire day is modelled as a single slice or

```
MORN, 6,
AFTERNOON, 12,
EVENING, 17,
NIGHT, 20
```
to model four different day-slices namely, 6 AM – 12 noon, 12 noon – 5 PM, 5 PM – 8 PM and 8 PM – 6 AM

3 Geographic granularity

Rumi allows geographic disaggregation at four levels. As with time specifications, only the top level geography specification is mandatory, i.e. the model must refer to some geographic area. Below that, the modeller can stop at any level of detail that's sufficient for them. All elements of a higher level must be mapped to at least one element of a lower level, if the lower level disaggregation exists. None of the geography related files have a header row.

3.1 ModelGeography.csv

Directory: `Root`

This is a very simple file with a single row and column (and no header row). The single cell just contains the name of the geography that is being modelled.

Example:

INDIA or
ASIA or
WORLD

3.2 SubGeography1.csv

Directory: Root

This file lists the top level geographies into which the ModelGeography is divided. This file too does not have a header row. It consists of just one row of comma-separated values representing the various sub-geographies.

Example:

NORTH, WEST, SOUTH, EAST, NORTHEAST or
INDONESIA, MALAYSIA, PHILLIPINES, VIETNAM or
EUROPE, NORTHAMERICA, SOUTHAMERICA, ASIA, AFRICA, AUSTRALIA

3.3 SubGeography2.csv

Directory: Root

This level gives the breakup of SubGeography1 items. It consists of as many rows as the number of SubGeography1 items. The first column of each row represents the SubGeography1 item, while the subsequent columns represent the breakup of that item into its sub-geographies. This file too has no header row.

Example:

EUROPE, GERMANY, NETHERLANDS, FRANCE, UK, ESTONIA, LITHUANIA
NORTHAMERICA, USA, CANADA
SOUTHAMERICA, BRAZIL, ARGENTINA, URUGUAY, PARAGUAY, CHILE, PERU

3.4 SubGeography3.csv

Directory: Root

Similar to SubGeography2, this gives a breakup of each SubGeography2 item into further sub-geographies.

Example:

USA, NEW YORK, CALIFORNIA, TEXAS, MICHIGAN
CANADA, ALBERTA, BRITISH COLUMBIA, ONTARIO, QUEBEC

4 Energy carriers

This set of files defines the various energy carriers being modelled. There are four types of energy carriers: physical primary carriers, non-physical primary carriers, physical derived carriers and non-physical derived carriers. There is one file for each of these.

4.1 NonPhysicalPrimaryCarriers.csv

Directory: `Root`

Columns: `EnergyCarrier`

Non-physical energy carriers are those that represent some energy source but do not have a physical form, but which can be used to produce other physical or non-physical energy carriers (e.g. sunlight, wind etc.). Only the name of the carrier is required for non-physical primary energy carriers. Hence, just one column is required with each row representing a non-physical energy carrier.

Example:

```
SUNLIGHT
WIND
ATOMICENERGY
```

4.2 PhysicalPrimaryCarriers.csv

Directory: `Root`

Columns: `EnergyCarrier, BalancingArea, BalancingTime, PhysicalUnit, EnergyUnit, DomEnergyDensity, ImpEnergyDensity`

Physical primary carriers, as the name suggests, have physical properties such as physical units and energy density which need to be given. Moreover, some of these quantities may be different for 'domestic' (i.e. those that are available/produced within the ModelGeography) and 'imported' (i.e. obtained from outside the ModelGeography) variants. The balancing area and balancing time of the carrier represents the granularity at which accounting for this carrier is done – i.e. where the supply and demand are matched. These have to be one of the keywords corresponding to the geographic and temporal granularities. The keywords corresponding to geographic granularity are `MODELGEOGRAPHY`, `SUBGEOGRAPHY1`, SUBGEOGRAPHY2 and `SUBGEOGRAPHY3`. The keywords corresponding to temporal granularity are `YEAR`, `SEASON`, `DAYTYPE` and `DAYSLICE`. The physical and energy units are just names representing the units in which this carrier's quantities are given – to enable easy conversion from/to other carriers. The energy density is the amount of energy (in energy units) per quantity (in physical units) and is given separately for the domestic and imported variants.

Example:

```
COKING_COAL, SUBGEOGRAPHY1, YEAR, MT, PJ, 20.9792705, 28.0285326
```

```
STEAM_COAL, SUBGEOGRAPHY1, YEAR, MT, PJ, 17.35076799, 22.60872
NATGAS, MODELGEOGRAPHY, YEAR, BCM, PJ, 37.6812, 37.6812
```

4.3 NonPhysicalDerivedCarriers.csv

Directory: `Root`

Columns: `EnergyCarrier, BalancingArea, BalancingTime, EnergyUnit`

For non-physical derived energy carriers, one needs to give the balancing area and time, and the units in which the energy in this carrier is measured. Physical units and densities are not meaningful in this case. The current version of Rumi does not support import of derived energy carriers.

Example:

```
ELECTRICITY, SUBGEOGRAPHY2, DAYSLICE, MWh
```

4.4 PhysicalDerivedCarriers.csv

Directory: `Root`

Columns: `EnergyCarrier, BalancingArea, BalancingTime, EnergyUnit, PhysicalUnit, EnergyDensity`

For physical derived energy carriers, in addition to the information given for non-physical derived carriers, some physical attributes such as physical units and energy density are required. The values required are similar to physical primary carriers, except that there are no domestic and imported variants.

Example:

```
MS, MODELGEOGRAPHY, YEAR, PJ, MT, 44.79876
HSD, MODELGEOGRAPHY, YEAR, PJ, MT, 43.33338
```

5 Other elements

Some additional information – not all of which is directly related to estimating demand or matching supply and demand – is provided through these files. Some of the inputs are currently placeholders, which can be used by the modeller to compute some interesting values after the model has run and produced outputs. The model itself does not use these values. Many of these inputs are also optional.

5.1 EmissionTypes.csv

[This input is currently accepted but not processed by Rumi]

Directory: `Root`

Columns: `EmissionType, EmissionUnit`

This file lists the various kinds of emissions that are modelled. This input is optional. Currently, Rumi does not process this file. For each emission type, its name and the unit (both just strings) in which the emissions are measured are given in the respective columns. The unit is used only for reporting.

Example:

```
CO2, tonne
PM2.5, kg
```

5.2 PhysicalCarrierEmissions.csv

[This input is currently accepted but not processed by Rumi]

Directory: `Root`

Columns: `EnergyCarrier, EmissionType, DomEmissionFactor, ImpEmissionFactor`

This input specifies the emissions of each emission type from a physical carrier. Non-physical carriers are assumed not to have any emissions. The physical carriers for which emissions are specified, may be primary or derived. This input is optional and is required only if there is at least one emission type. However, this input is not processed in the current version of Rumi. For each physical carrier and each applicable emission type, the default[4] emission units per physical unit of the carrier has to be specified. If a carrier's emissions for an emission type is not given, it's assumed to be zero. For a primary physical energy carrier, two emission values can be specified for each applicable emission type – one for the domestic variant and one for the imported variant. For derived energy carriers, currently imports are not supported and hence the emission factor for the imported variant is not meaningful, but for technical reasons it still needs to be supplied.

Example:

```
COAL, CO2, 1500, 2000
COAL, PM2.5, 15000, 10000
```

5.3 CurrencyUnit.csv

Directory: `Root`

This is a simple file without a header row and with just one row and one column. The single cell just represents the name of the default current unit used for prices and costs in the model. It is not used anywhere in the model processing but can be useful for consistency checking of model inputs and for subsequent reporting.

Example

[4] 'Default' because this can be overridden in the demand / supply inputs if desired.

`Million Rupees`

5.4 UnmetDemandValue.csv

Directory: `Root`

Columns: `EnergyCarrier, Year, UnmetDemandValue`

This file represents the 'value of unserved energy' for each carrier in each year. This value represents the per-unit cost of the carrier beyond which it is cheaper to not supply this energy rather than source it from somewhere. In other words, this is the cost to be paid per unit of unserved energy, if the model chooses to not supply it. This value is assumed to be the same across the entire ModelGeography, and is given for each year in the model period. The unmet demand value itself is given in currency units per unit of energy carrier, where the unit is the physical unit for physical carriers and energy unit for non-physical carriers. Each row represents the unmet demand value for one carrier and year combination. If the value is not given for any carrier-year combination, it's assumed that the demand of that energy carrier in that year (in all balancing geographies) must necessarily be fully met regardless of the cost incurred.

Example:

```
ELECTRICITY, 2021, 100
MS, 2028, 300000
```

5.5 Demographics.csv

[This input is currently accepted but not processed by Rumi]

Directory: `Root`

Columns: `Year, Geography levels, Population, NumHouseholds`

This file specifies the demographics of the modelled geography. In particular, it captures information about the population and number of households in the various geographic areas modelled. It is not directly relevant to the model's computation but could be useful to report items such as per-household or per-capita values. The parameter is optional.

If the parameter is given, the first column represents the year for which the population and household count is given. The subsequent few columns correspond to the geographic granularity at which this information is being provided, and the column headings should correspondingly be from `ModelGeography, SubGeography1, SubGeography2 and SubGeography3` – to the extent required. The geographic granularity must be the same in all rows of the file. The final two columns are positive integers representing the population and number of households of the chosen geographic unit.

Example

```
2023, INDIA, WEST, MH, 12331334, 2612344
```

5.6 GDP.csv

Directory: `Root`

Columns: `Year, Geography Levels, GDP`

This file is to specify the GDP as an input, where the GDP column is a positive real number in the currency units. As with demographics, the temporal granularity is annual and the geographic granularity can be any chosen granularity. This is also an optional parameter. But if it is not given, GDP-elasticity based demand estimation cannot be done. If this input is provided at a finer (or equal) geographic granularity than what is required for GDP-elasticity based calculations, then the GDP would be aggregated up to the required level to calculate the GDP growth rate. If it is provided at a coarser geographic granularity than required for the elasticity calculations, then the GDP growth rate calculated at the coarser granularity would be used for all corresponding finer granularities.

If this input is provided, it must be provided for all model years and all combinations of geographic units at the chosen granularity. It should also be provided for the year before the model's first year, if GDP-elasticity based calculation is to be done, since that forms the 'base value' for GDP elasticity calculations.

Example

```
2020, INDIA, 242342342
```