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

Overview

Table of Contents

1 Overview

Rumi is a generic, open-source energy systems modelling platform developed by Prayas (Energy Group) to aid policy-relevant analysis. Its design enables spatially and temporally disaggregated modelling, and modelling of energy demand in detail. This documentation pertains to version 1.0 of Rumi.

Given suitable inputs, the demand estimation component of Rumi estimates energy demand for various sectors, energy carriers, energy services etc., at the temporal and geographic granularity specified. The supply estimation component of Rumi identifies supply sources to meet the estimated demand at least cost, based on various supply-side inputs. Figure 1 depicts this architecture of Rumi.

Figure 1: Overview of Rumi
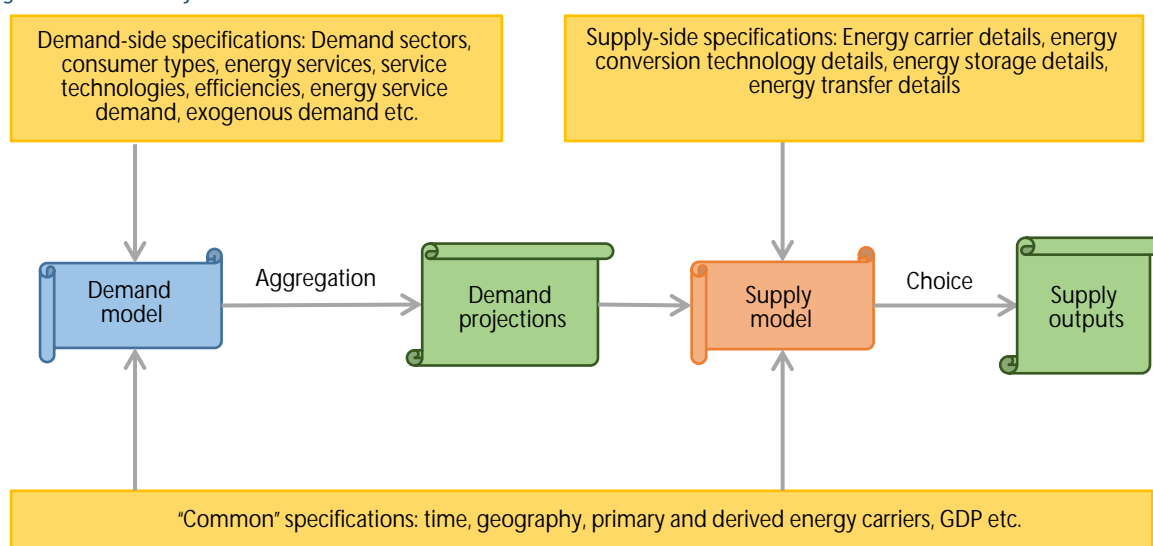
2 Installing Rumi

To support easy interoperability and open-sourcing, Rumi has been built on Python and Pyomo. Python needs to be installed on the computer on which Rumi will be installed and run. Instructions to install Rumi (and all the software it depends on) are provided in the `README.md` file available in the root directory of the Rumi platform repository. In addition to this, a solver[1] needs to be installed to run the supply component of Rumi. All documentation about Rumi is available in the `Docs.zip` file available under the latest release tag on the Rumi platform repository (https://github.com/prayas-energy/Rumi).

3 Setting up a Rumi model instance

To use Rumi and obtain results from running the model, one has to set up a *model instance*. A model instance is nothing but a collection of data files describing the kinds of things that are being modelled (e.g. the geographic areas, the time period, the energy carriers and their characteristics, the technologies used to produce some energy carriers and their characteristics etc.). Details of the input data files are given in Section 4. A complete model instance called PIER is available for free download and use from https://github.com/prayas-energy/PIER.
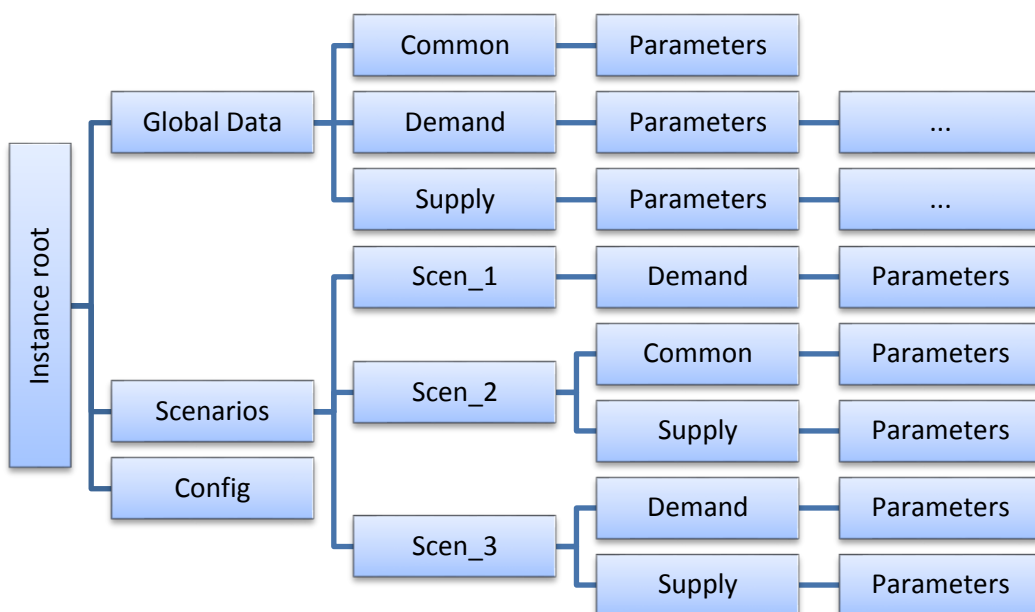
The model instance is expected to follow a certain directory (folder) structure as described below. One directory, called `Scenarios` is expected in the root directory of the model instance. As the name suggests, this directory contains the data pertaining to all the scenarios that are modelled. Under the `Scenarios` directory, there must be one directory per scenario that is modelled, where the directory name corresponds to the scenario name.

Each scenario directory consists of three sub-directories called `Common`, `Demand` and `Supply`, each of which have of a further sub-directory called `Parameters`. The `Parameters` directory contains a set of `.csv` files which contain the actual data used by Rumi. There is also a directory structure that is expected within the `Demand` and `Supply` directories, which is explained in their respective documentation. Typically, much of the input data is common across the various scenarios. To facilitate this, Rumi supports a directory called "`Global Data`" (without the double-quotes) in the root directory which contains the data inputs to be used by default in any scenario. This directory has a structure similar to the scenario directories. That is, `Global Data` too has three sub-directories called `Common`, `Demand` and `Supply`, each of which consists of a further sub-directory called `Parameters` containing the actual `.csv` data files that are to be used by default across scenarios. Each individual scenario directory can then only have the data files that are specific to that scenario. If a scenario does not contain some input data file, that file is taken from the `Global Data` directory. Thus, typically, the directory structure of a model instance would be as shown in Figure 2[2].

---

[1] `cbc`, `glpk`, `cplex` and `gurobi` are some of the commonly known and used solvers. Not all solvers are available freely, i.e., they may have to be purchased. Please see Section 3 of this document to understand how to tell Rumi which solver you are using.

[2] Strictly speaking, the `Global Data` directory need not be present, if all data is provided for each scenario.

The `Config` directory in the Rumi installation contains a (default) configuration file named `Config.yml`. The modeller should specify configuration parameters for Rumi in this file. In this release, configuration parameters are supported only for the supply component of Rumi. The only mandatory configuration parameter in this configuration file is `solver_name` which specifies the name of the solver to be used by the supply component. All the other configuration parameters that can be specified in the configuration file viz. `solver_executable`, `solver_options_file`, `symbolic_solver_labels` are optional. The supplied `Config.yml` file in Rumi contains explanatory documentation for all these parameters. Note that if a line in the configuration file starts with the # character, it is treated as a comment line and ignored. Section 5 explains how to run the Rumi components.

Rumi also provides the facility to optionally have a `Config.yml` configuration file for each model instance. In that case, the file should be present in the `Config` directory of the model, and will override the default `Config.yml`.

The `Config` directory in Rumi also contains an `EnergyUnitConversion.csv` file. This file contains the numerical constants required to convert one energy unit to another – specified in a matrix format. In this release, the supplied file in Rumi specifies conversions from and to 24 different energy units. This file is used by the supply component to perform the required energy unit conversions. The modeller can choose to modify or extend this file in Rumi's `Config` directory as required. Rumi also provides the facility to have an `EnergyUnitConversion.csv` file with the model instance (in its

`Config` directory) – in which case, the file present in the model overrides the default file available in the `Config` directory of Rumi.

4   Inputs to Rumi

Rumi accepts three kinds of inputs corresponding to the three sub-directories that are expected under any scenario, namely:

- Inputs for demand processing in the `Demand` directory
- Inputs for supply processing in the `Supply` directory
- Inputs that are common to both supply and demand processing in the `Common` directory

The three specification document files available in the Docs directory of the Rumi installation, namely `common-inputs-documentation.pdf`, `demand-documentation.pdf` and `supply-documentation.pdf` describe the various input files that are expected, where they need to be located, and what they should contain.

All data input files to Rumi are `.csv` (comma-separated value) files. Most of these input files also contain a header row with pre-defined names for columns in the file[3]. For all such files with a header row, the order of the columns does not matter, since the column heading indicates what each column represents. Moreover, such files can also have extra columns with headings not recognized by Rumi. Such columns are simply ignored and can be useful to document some features of the input data being provided.

---

[3] The few exceptions where column headers are not required are described in the respective inputs description files. For such files, the order of columns matters.

5   Running Rumi

Once Rumi has been installed, and a model instance has been set-up, it can be run as follows. This document only provides a broad overview of the commands to run the model and does not provide all the options available to the modeller. Options are generally in three categories: mandatory parameters that must be supplied, optional parameters to change some defaults, and optional parameters to define the scope of the command. Details of all the options available for the commands are provided in the `README.md` file available in the root directory of the Rumi platform repository. Here, we briefly mention the mandatory parameters that must be supplied with the Rumi commands.

To run the demand component of Rumi, use the following command on the command line:

```
rumi_demand -m MODEL_INSTANCE_PATH -s SCENARIO
```

To run the supply component of Rumi, use the following command on the command line:

```
rumi_supply -m MODEL_INSTANCE_PATH -s SCENARIO
```

In both the above examples:

```
MODEL_INSTANCE_PATH   ➔        Root directory of the model instance
SCENARIO              ➔        Name of the scenario to run in the model
```

Note that all the parameters are case sensitive.

Contact Prayas (Energy Group) at energy.model@prayaspune.org for any queries regarding Rumi.