
otoole Documentation

Release unknown

Will Usher

Apr 29, 2021

Contents

1	Description	3
1.1	Dependencies	4
2	Installation	5
3	Usage	7
4	Contributing	9
5	Getting Started	11
6	Contents	13
6.1	License	13
6.2	Contributors	13
6.3	Changelog	13
6.4	otoole	14
7	Indices and tables	27
	Python Module Index	29
	Index	31

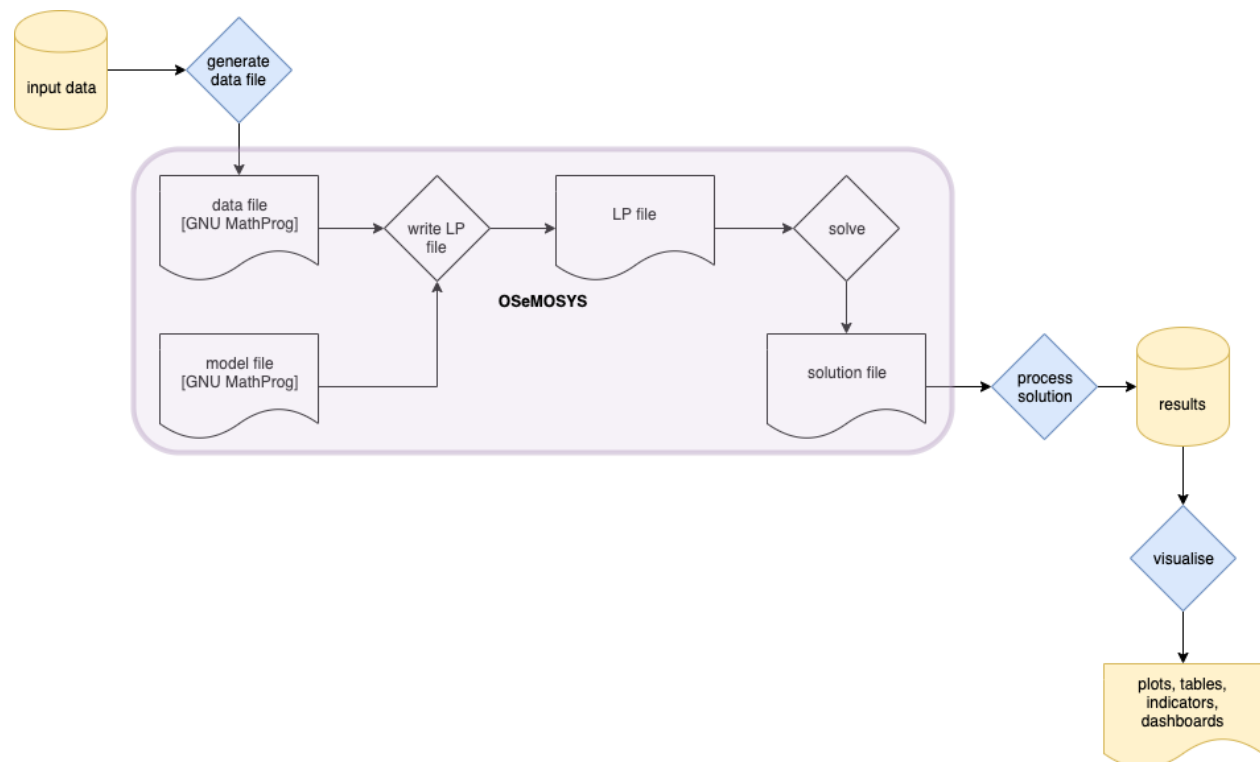
A Python toolkit to support use of OSeMOSYS

CHAPTER 1

Description

otoole is a Python package which provides a command-line interface for users of OSeMOSYS.

The aim of the package is to provide a community resource which centralises the commonly used pre- and post-processing steps around the use of OSeMOSYS.



otoole aims to support different ways of storing input data and results, including csv files, databases, datapackages and Excel workbooks, as well as different implementations of the OSeMOSYS model.

1.1 Dependencies

otoole requires a number of dependencies, including *pygraphviz*, which can be difficult to install on Windows.

The easiest way to install the dependencies is to use *miniconda*.

1. Obtain the [miniconda](#) package:
2. Add the **conda-forge** channel `conda config --add channels conda-forge`
3. Create a new Python environment `conda create -n myenv python=3.7 networkx datapackage pandas pulp graphviz xlrd`
4. Activate the new environment `conda activate myenv`
5. Use *pip* to install *otoole* `pip install otoole`

CHAPTER 2

Installation

Install **otoole** using pip:

```
pip install otoole
```

To upgrade **otoole** using pip:

```
pip install otoole --upgrade
```


CHAPTER 3

Usage

For detailed instructions of the use of the tool, run the command line help function:

```
otoole --help
```


CHAPTER 4

Contributing

New ideas and bugs are found on the repository Issue Tracker. Please do contribute by discussing and developing these ideas further, or by developing the codebase.

To contribute directly to the documentation of code development, you first need to install the package in *develop mode*:

```
git clone http://github.com/OSeMOSYS/otoole
cd otoole
git checkout <branch you wish to use>
python setup.py develop
```

Now, all changes made in the codebase will automatically be reflected in the installed Python version accessible on the command line or from importing otoole modules into other Python packages.

CHAPTER 5

Getting Started

Install otoole using pip:

```
pip install otoole
```

Check the version installed:

```
otoole -V
```

Download an OSeMOSYS datapackage and convert it to a modelfile:

```
otoole convert datapackage datafile https://zenodo.org/record/3479823/files/KTH-dESA/  
↳simplicity-v0.1a0.zip simplicity.txt
```

Visualise the Reference Energy System:

```
otoole viz res https://zenodo.org/record/3479823/files/KTH-dESA/simplicity-v0.1a0.zip  
↳res.png && open res.png
```

Alternatively, convert an OSeMOSYS datafile to a datapackage:

```
otoole convert datafile datapackage simplicity.txt simplicity
```


6.1 License

The MIT License (MIT)

Copyright (c) 2019 Will Usher

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

6.2 Contributors

- Will Usher <wusher@kth.se>

6.3 Changelog

6.3.1 Version 0.4

- Tidy up the command line interface

- Convert to/from SQLite database from/to datapackage
- Remove rotten pygraphviz dependency

6.3.2 Version 0.3

- Create a Tabular Data Package from an OSeMOSYS datafile

6.3.3 Version 0.2

- Visualise a reference energy system from a Tabular Data Package

6.3.4 Version 0.1

- Add CPLEX to csv or CBC solution file conversion script
- Create CSV files in a folder from an excel workbook
- Create a Tabular Data Package from a folder of CSVs
- Create an OSeMOSYS datafile from a Tabular Data Package
- Adds a command line interface to access these tools

6.4 otoole

6.4.1 otoole package

Subpackages

otoole.preprocess package

Submodules

otoole.preprocess.create_datapackage module

Creates a datapackage from a collection of CSV files of OSeMOSYS input data

- Uses Frictionless Data datapackage concept to build a JSON schema of the dataset
- Enforces relations between sets and indices in parameter files

```
otoole.preprocess.create_datapackage.convert_datapackage_to_sqlite(path_to_datapackage,  
                                                                    sqlite)
```

Load and save table to SQLite

```
otoole.preprocess.create_datapackage.generate_package(path_to_package)
```

Creates a datapackage in folder *path_to_package*

```
[[{'fields': 'REGION', 'reference': {'resource': 'REGION', 'fields': 'VALUE'}}]]
```

```
otoole.preprocess.create_datapackage.main(wide_folder, narrow_folder)
```

```
otoole.preprocess.create_datapackage.validate_contents(path_to_package)
```

otoole.preprocess.datafile_to_datapackage module

Converts an OSeMOSYS datafile to a Tabular Data Package

```
otoole.preprocess.datafile_to_datapackage.convert_amply_data_to_list (amply_data:
                                                                    Dict[KT,
                                                                    VT])
                                                                    →
                                                                    List[List[T]]
```

Flattens a dictionary into a list of lists

Parameters `amply_data` (*dict*) –

```
otoole.preprocess.datafile_to_datapackage.convert_amply_to_dataframe (datafile_parser,
                                                                    con-
                                                                    fig) →
                                                                    Dict[str,
                                                                    pan-
                                                                    das.core.frame.DataFrame]
```

Converts an amply parser to a dict of pandas dataframes

Parameters

- `datafile_parser` (*AmPLY*) –
- `config` (*Dict*) –

Returns

Return type dict of pandas.DataFrame

```
otoole.preprocess.datafile_to_datapackage.convert_file_to_package (path_to_datafile:
                                                                    str,
                                                                    path_to_datapackage:
                                                                    str)
```

Converts an OSeMOSYS datafile to a Tabular Data Package

Parameters

- `path_to_datafile` (*str*) –
- `path_to_datapackage` (*str*) – Path to the folder in which to write the new Tabular Data Package

```
otoole.preprocess.datafile_to_datapackage.load_parameter_definitions (config:
                                                                    dict)
                                                                    → str
```

Load the set and parameter dimensions into datafile parser

Returns

Return type *str*

```
otoole.preprocess.datafile_to_datapackage.read_in_datafile (path_to_datafile:  str;
                                                                    config:      Dict[KT,
                                                                    VT])        →      am-
                                                                    ply.amply.AmPLY
```

Read in a datafile using the AmPLY parsing class

Parameters

- `path_to_datafile` (*str*) –
- `config` (*Dict*) –

otoole.preprocess.excel_to_osemosys module

Extract data from spreadsheets and write to an OSeMOSYS datafile

Reads and writes the following OSeMOSYS parameters and sets to and from files on disk.

Notes

Sets

These are the standard sets:

```
set YEAR;
set TECHNOLOGY;
set TIMESLICE;
set FUEL;
set EMISSION;
set MODE_OF_OPERATION;
set REGION;
set SEASON;
set DAYTYPE;
set DAILYTIMEBRACKET;
set FLEXIBLEDEMANDTYPE;
set STORAGE;
```

All sets are written in a CSV file in one column of values with header of VALUE. For example, the CSV file for set YEAR:

```
VALUE
2015
2016
2017
2018
2019
2020
```

Sets are written into the OSeMOSYS data file using the following syntax:

```
set YEAR := 2014 2015 2016 2017 2018 2019 2020 ;
```

param In general, parameters can be written into CSV files in narrow or wide

param formats. In narrow format, the CSV file should look as follows::

```
REGION,TIMESLICE,YEAR,VALUE      SIMPLICITY,ID,2014,0.1667      SIMPLIC-
ITY,IN,2014,0.0833  SIMPLICITY,SD,2014,0.1667  SIMPLICITY,SN,2014,0.0833  SIMPLIC-
ITY,WD,2014,0.3333 SIMPLICITY,WN,2014,0.1667
```

In wide format, the final index is transposed:

```
REGION,TIMESLICE,2014,2015,...
SIMPLICITY,ID,0.1667,0.1667
SIMPLICITY,IN,0.0833,0.0833
SIMPLICITY,SD,0.1667,0.1667
SIMPLICITY,SN,0.0833,0.0833
SIMPLICITY,WD,0.3333,0.3333
SIMPLICITY,WN,0.1667,0.1667
```

Wide format is a bit nicer to use with spreadsheets, as it allows you to more easily plot graphs of values, but narrow format is a more flexible and easily manipulated data format.

Writing parameters:

1-dimensional e.g. DiscountRate{r in REGION}

2-dimension e.g. YearSplit{l in TIMESLICE, y in YEAR}

n-dimensional e.g. DaysInDayType{ls in SEASON, ld in DAYTYPE, y in YEAR}

Global parameters:

```
param YearSplit{l in TIMESLICE, y in YEAR};
param DiscountRate{r in REGION};
param DaySplit{lh in DAILYTIMEBRACKET, y in YEAR};
param Conversionls{l in TIMESLICE, ls in SEASON};
param Conversionld{l in TIMESLICE, ld in DAYTYPE};
param Conversionlh{l in TIMESLICE, lh in DAILYTIMEBRACKET};
param DaysInDayType{ls in SEASON, ld in DAYTYPE, y in YEAR};
param TradeRoute{r in REGION, rr in REGION, f in FUEL, y in YEAR};
param DepreciationMethod{r in REGION};
```

Demand parameters:

```
param SpecifiedAnnualDemand{r in REGION, f in FUEL, y in YEAR};
param SpecifiedDemandProfile{r in REGION, f in FUEL, l in TIMESLICE, y in YEAR};
param AccumulatedAnnualDemand{r in REGION, f in FUEL, y in YEAR};
```

Performance parameters:

```
param CapacityToActivityUnit{r in REGION, t in TECHNOLOGY};
param TechWithCapacityNeededToMeetPeakTS{r in REGION, t in TECHNOLOGY};
param CapacityFactor{r in REGION, t in TECHNOLOGY, l in TIMESLICE, y in YEAR};
param AvailabilityFactor{r in REGION, t in TECHNOLOGY, y in YEAR};
param OperationalLife{r in REGION, t in TECHNOLOGY};
param ResidualCapacity{r in REGION, t in TECHNOLOGY, y in YEAR};
param InputActivityRatio{r in REGION, t in TECHNOLOGY, f in FUEL, m in MODE_OF_
↳ OPERATION, y in YEAR};
param OutputActivityRatio{r in REGION, t in TECHNOLOGY, f in FUEL, m in MODE_OF_
↳ OPERATION, y in YEAR};
```

Technology Costs parameters:

```
param CapitalCost{r in REGION, t in TECHNOLOGY, y in YEAR};
param VariableCost{r in REGION, t in TECHNOLOGY, m in MODE_OF_OPERATION, y in YEAR};
param FixedCost{r in REGION, t in TECHNOLOGY, y in YEAR};
```

Storage parameters:

```
param TechnologyToStorage{r in REGION, t in TECHNOLOGY, s in STORAGE, m in MODE_OF_
↳ OPERATION};
param TechnologyFromStorage{r in REGION, t in TECHNOLOGY, s in STORAGE, m in MODE_OF_
↳ OPERATION};
param StorageLevelStart{r in REGION, s in STORAGE};
param StorageMaxChargeRate{r in REGION, s in STORAGE};
param StorageMaxDischargeRate{r in REGION, s in STORAGE};
param MinStorageCharge{r in REGION, s in STORAGE, y in YEAR};
param OperationalLifeStorage{r in REGION, s in STORAGE};
```

(continues on next page)

(continued from previous page)

```
param CapitalCostStorage{r in REGION, s in STORAGE, y in YEAR};
param ResidualStorageCapacity{r in REGION, s in STORAGE, y in YEAR};
```

Capacity Constraints parameters:

```
param CapacityOfOneTechnologyUnit{r in REGION, t in TECHNOLOGY, y in YEAR};
param TotalAnnualMaxCapacity{r in REGION, t in TECHNOLOGY, y in YEAR};
param TotalAnnualMinCapacity{r in REGION, t in TECHNOLOGY, y in YEAR};
```

Investment Constraints parameters:

```
param TotalAnnualMaxCapacityInvestment{r in REGION, t in TECHNOLOGY, y in YEAR};
param TotalAnnualMinCapacityInvestment{r in REGION, t in TECHNOLOGY, y in YEAR};
```

Activity Constraints parameters:

```
param TotalTechnologyAnnualActivityUpperLimit{r in REGION, t in TECHNOLOGY, y in YEAR}
↳;
param TotalTechnologyAnnualActivityLowerLimit{r in REGION, t in TECHNOLOGY, y in YEAR}
↳;
param TotalTechnologyModelPeriodActivityUpperLimit{r in REGION, t in TECHNOLOGY};
param TotalTechnologyModelPeriodActivityLowerLimit{r in REGION, t in TECHNOLOGY};
```

Reserve Margin parameters:

```
param ReserveMarginTagTechnology{r in REGION, t in TECHNOLOGY, y in YEAR};
param ReserveMarginTagFuel{r in REGION, f in FUEL, y in YEAR};
param ReserveMargin{r in REGION, y in YEAR};
```

RE Generation Target parameters:

```
param RETagTechnology{r in REGION, t in TECHNOLOGY, y in YEAR};
param RETagFuel{r in REGION, f in FUEL, y in YEAR};
param REMinProductionTarget{r in REGION, y in YEAR};
```

Emissions & Penalties parameters:

```
param EmissionActivityRatio{r in REGION, t in TECHNOLOGY, e in EMISSION, m in MODE_OF_
↳OPERATION, y in YEAR};
param EmissionsPenalty{r in REGION, e in EMISSION, y in YEAR};
param AnnualExogenousEmission{r in REGION, e in EMISSION, y in YEAR};
param AnnualEmissionLimit{r in REGION, e in EMISSION, y in YEAR};
param ModelPeriodExogenousEmission{r in REGION, e in EMISSION};
param ModelPeriodEmissionLimit{r in REGION, e in EMISSION};
```

```
otoole.preprocess.excel_to_osemosys.generate_csv_from_excel(input_workbook,
                                                             output_folder)
```

Generate a folder of CSV files from a spreadsheet

Parameters

- **input_workbook** (*str*) – Path to spreadsheet containing OSeMOSYS data
- **output_folder** (*str*) – Path of the folder containing the csv files

```
otoole.preprocess.excel_to_osemosys.read_config(path_to_user_config: str = None) →
Dict[KT, VT]
```

Reads the config file holding expected OSeMOSYS set and parameter dimensions

Parameters `path_to_user_config` (*str*, *optional*, *default=None*) – Optional path to a user defined configuration file

Returns

Return type `dict`

otoole.preprocess.longify_data module

`otoole.preprocess.longify_data.check_datatypes` (*narrow*: `pandas.core.frame.DataFrame`,
config_details: `Dict[KT, VT]`,
parameter: `str`) → `pandas.core.frame.DataFrame`

Checks a parameters datatypes

Parameters

- **narrow** (`pandas.DataFrame`) – The parameter data
- **config_details** (`dict`) – The configuration dictionary
- **parameter** (`str`) – The name of the parameter

`otoole.preprocess.longify_data.check_parameter` (*df*, *config_details*, *name*)

`otoole.preprocess.longify_data.check_set` (*df*, *config_details*, *name*)

`otoole.preprocess.longify_data.check_set_datatype` (*narrow*: `pandas.core.frame.DataFrame`,
config_details: `Dict[KT, VT]`, *set_name*: `str`) → `pandas.core.frame.DataFrame`

Checks the datatypes of a set_name dataframe

Parameters

- **narrow** (`pandas.DataFrame`) – The set data
- **config_details** (`dict`) – The configuration dictionary
- **set_name** (`str`) – The name of the set

`otoole.preprocess.longify_data.main` (*output_folder*, *narrow_folder*)

Read in a folder of irregular wide-format files and write as narrow csvs

`otoole.preprocess.longify_data.write_out_dataframe` (*folder*, *parameter*, *df*)

Writes out a dataframe as a csv into the data subfolder of a datapackage

Parameters

- **folder** (`str`) –
- **parameter** (`str`) –
- **df** (`pandas.DataFrame`) –

otoole.preprocess.narrow_to_datafile module

`otoole.preprocess.narrow_to_datafile.main` (*datapackage*, *datafilepath*, *sql=False*)

`otoole.preprocess.narrow_to_datafile.read_narrow_csv` (*filepath*)

```
otoole.preprocess.narrow_to_datafile.write_parameter(filepath: io.TextIOBase, df: pandas.core.frame.DataFrame, parameter_name, default)
```

Parameters

- **filepath** (*StreamIO*) –
- **df** (*pandas.DataFrame*) –
- **parameter_name** (*str*) –
- **default** (*int*) –

```
otoole.preprocess.narrow_to_datafile.write_set(filepath: io.TextIOBase, df: pandas.core.frame.DataFrame, set_name)
```

Parameters

- **filepath** (*StreamIO*) –
- **df** (*pandas.DataFrame*) –
- **parameter_name** (*str*) –

Module contents

```
otoole.preprocess.generate_csv_from_excel(input_workbook, output_folder)
```

Generate a folder of CSV files from a spreadsheet

Parameters

- **input_workbook** (*str*) – Path to spreadsheet containing OSeMOSYS data
- **output_folder** (*str*) – Path of the folder containing the csv files

```
otoole.preprocess.create_datapackage(wide_folder, narrow_folder)
```

```
otoole.preprocess.create_datafile(datapackage, datafilepath, sql=False)
```

```
otoole.preprocess.convert_file_to_package(path_to_datafile: str, path_to_datapackage: str)
```

Converts an OSeMOSYS datafile to a Tabular Data Package

Parameters

- **path_to_datafile** (*str*) –
- **path_to_datapackage** (*str*) – Path to the folder in which to write the new Tabular Data Package

otoole.results package

Submodules

otoole.results.convert module

Converts an OSeMOSYS solution file from CPLEX, CBC or GLPK into CBC or CSV format


```
class otoole.results.convert.ConvertLine (data: List[T], start_year: int, end_year: int, output_format='cbc')
```

Bases: `object`

Abstract class which defines the interface to the family of converters

Inherit this class and implement the `_do_it()` method to produce the data to be written out into a new format

Example

```
>>> cplex_line = "AnnualCost          REGION  CDBACKSTOP          1.0          0.0          ↵
↵137958.8400384134"
>>> convertor = RegionTechnology()
>>> convertor.convert()
VariableName (REGION, TECHCODE01, 2015)          42.69          0\n
VariableName (REGION, TECHCODE01, 2017)          137958.84          0\n
```

convert () → List[str]

convert_cbc () → List[str]

Format the data for writing to a CBC file

convert_csv () → List[str]

Format the data for writing to a csv file

```
class otoole.results.convert.RegionTechnology (data: List[T], start_year: int, end_year: int, output_format='cbc')
```

Bases: `otoole.results.convert.ConvertLine`

```
class otoole.results.convert.RegionTimeSliceTechnologyMode (data: List[T],
start_year: int,
end_year: int, output_format='cbc')
```

Bases: `otoole.results.convert.ConvertLine`

```
otoole.results.convert.convert_cplex_file (cplex_filename: str, output_filename: str,
start_year=2015, end_year=2070, output_format='cbc')
```

Converts a CPLEX solution file into that of the CBC solution file

Parameters

- **cplex_filename** (*str*) – Path to the transformed CPLEX solution file
- **output_filename** (*str*) – Path for the processed data to be written to

```
otoole.results.convert.process_line (line: str, start_year: int, end_year: int, output_format: str) → List[str]
```

Processes an individual line in a CPLEX file

A different `ConvertLine` implementation is chosen depending upon the variable name

Parameters

- **line** (*str*) –
- **start_year** (*int*) –
- **end_year** (*int*) –
- **output_format** (*str*) – The file format required - either `csv` or `cbc`

Module contents

otoole.visualise package

Submodules

otoole.visualise.res module

Visualise the reference energy system

`otoole.visualise.res.build_graph` (*nodes: List[Tuple[str, Dict[KT, VT]]], edges: List[Tuple[str, Dict[KT, VT]]]*) → `networkx.classes.digraph.DiGraph`

Builds the graph using networkx

Parameters

- **nodes** (*list*) –
- **edges** (*list*) –

Returns A directed graph representing the reference energy system

Return type `networkx.DiGraph`

`otoole.visualise.res.create_res` (*path_to_datapackage: str, path_to_resfile: str*)

Create a reference energy system diagram from a Tabular Data Package

Parameters

- **path_to_datapackage** (*str*) – The path to the `datapackage.json`
- **path_to_resfile** (*str*) – The path to the image file to be created

`otoole.visualise.res.draw_graph` (*graph, path_to_resfile*)

Layout the graph and write it to disk

Uses pygraphviz to set some graph attributes, layout the graph and write it to disk

Parameters **path_to_resfile** (*str*) – The file path of the PNG image file that will be created

`otoole.visualise.res.load_datapackage` (*path_to_datapackage: str*) → `datapackage.Package`

Module contents

Visualise different aspects of an OSeMOSYS model and data

Provides the following commands:

```
otoole viz res <path_to_datapackage.json> <path_to_res_image>
```

`otoole viz res` generates an image of the reference energy system in a datapackage

`otoole.visualise.create_res` (*path_to_datapackage: str, path_to_resfile: str*)

Create a reference energy system diagram from a Tabular Data Package

Parameters

- **path_to_datapackage** (*str*) – The path to the `datapackage.json`
- **path_to_resfile** (*str*) – The path to the image file to be created

Submodules

otoole.cli module

Provides a command line interface to the `otoole` package

The key functions are **convert**, **cplex** and **viz**.

The **convert** command allows conversion of multiple different OSeMOSYS input formats including from/to csv, an AMPL format datafile, a Tabular Data Package, a folder of CSVs, an Excel workbook with one tab per parameter, an SQLite database

The **cplex** command provides access to scripts which transform and process a CPLEX solution file into a format which is more readily processed - either to CBC or CSV format.

The **viz** command allows you to produce a Reference Energy System diagram from a Tabular Data Package.

Example

Ask for help on the command line:

```
>>> $ otoole --help
usage: otoole [-h] [--verbose] [--version] {convert,cplex,viz} ...

otoole: Python toolkit of OSeMOSYS users

positional arguments:
{convert,cplex,viz}
  convert              Convert from one input format to another
  cplex                Process a CPLEX solution file
  viz                 Visualise the model

optional arguments:
-h, --help            show this help message and exit
--verbose, -v         Enable debug mode
--version, -V         The version of otoole
```

`otoole.cli.conversion_matrix(args)`

from o ex cs dp sq df excel - yy csv nn - yy nn nn datapackage nn ?? - yy yy sql nn - yy datafile nn ?? yy -

`otoole.cli.cplex2cbc(args)`

`otoole.cli.datapackage2res(args)`

`otoole.cli.get_parser()`

`otoole.cli.main()`

otoole.validate module

Ensures that technology and fuel names match the convention

For example, to validate the following list of names, you would use the config shown below:

```
theseUNIQUE_ENTRY1
are__UNIQUE_ENTRY2
all__UNIQUE_ENTRY1
validUNIQUE_ENTRY2
```

(continues on next page)

(continued from previous page)

```
entryUNIQUE_ENTRY1
in__UNIQUE_ENTRY2
a____UNIQUE_ENTRY1
list_UNIQUE_ENTRY2
```

Create a yaml validation config with the following format:

```
codes:
  some_valid_codes:
    UNIQUE_ENTRY1: Description of unique entry 1
    UNIQUE_ENTRY2: Description of unique entry 2
schema:
  schema_name:
  - name: first_entry_in_schema
    valid: ['these', 'are__', 'all__', 'valid', 'entry', 'in__', 'a____', 'list_']
    position: (1, 5) # a tuple representing the start and end position
  - name: second_entry_in_schema
    valid: some_valid_codes # references an entry in the codes section of the config
    position: (6, 19) # a tuple representing the start and end position
```

`otoole.validate.compose_expression (schema: List[T]) → str`

Generates a regular expression from a schema

Returns

Return type `str`

`otoole.validate.create_schema (config: Dict[KT, VT] = None)`

Populate the dict of schema with codes from the validation config

Parameters `config (dict, default=None)` – A configuration dictionary containing codes and schema keys

`otoole.validate.read_validation_config()`

`otoole.validate.validate (expression: str, name: str) → bool`

Determine if name matches the expression

Parameters

- `expression (str)` –
- `name (str)` –

Returns

Return type `bool`

`otoole.validate.validate_fuel_name (name: str) → bool`

Validate a fuel name

Parameters

- `name (str)` –
- `country_codes (list)` –
- `fuel_codes (list)` –

Returns

- True if name is valid according to `country_codes` and `fuel_codes`
- otherwise False

Module contents

`otoole.read_packaged_file` (*filename: str, module_name: str = None*)

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

O

- `otoole`, [25](#)
- `otoole.cli`, [23](#)
- `otoole.preprocess`, [20](#)
- `otoole.preprocess.create_datapackage`,
[14](#)
- `otoole.preprocess.datafile_to_datapackage`,
[15](#)
- `otoole.preprocess.excel_to_osemosys`, [16](#)
- `otoole.preprocess.longify_data`, [19](#)
- `otoole.preprocess.narrow_to_datafile`,
[19](#)
- `otoole.results`, [22](#)
- `otoole.results.convert`, [20](#)
- `otoole.validate`, [23](#)
- `otoole.visualise`, [22](#)
- `otoole.visualise.res`, [22](#)

B

`build_graph()` (in module `otoole.visualise.res`), 22

C

`check_datatypes()` (in module `otoole.preprocess.longify_data`), 19

`check_parameter()` (in module `otoole.preprocess.longify_data`), 19

`check_set()` (in module `otoole.preprocess.longify_data`), 19

`check_set_datatype()` (in module `otoole.preprocess.longify_data`), 19

`compose_expression()` (in module `otoole.validate`), 24

`conversion_matrix()` (in module `otoole.cli`), 23

`convert()` (`otoole.results.convert.ConvertLine` method), 21

`convert_amply_data_to_list()` (in module `otoole.preprocess.datafile_to_datapackage`), 15

`convert_amply_to_dataframe()` (in module `otoole.preprocess.datafile_to_datapackage`), 15

`convert_cbc()` (`otoole.results.convert.ConvertLine` method), 21

`convert_cplex_file()` (in module `otoole.results.convert`), 21

`convert_csv()` (`otoole.results.convert.ConvertLine` method), 21

`convert_datapackage_to_sqlite()` (in module `otoole.preprocess.create_datapackage`), 14

`convert_file_to_package()` (in module `otoole.preprocess`), 20

`convert_file_to_package()` (in module `otoole.preprocess.datafile_to_datapackage`), 15

`ConvertLine` (class in `otoole.results.convert`), 20

`cplex2cbc()` (in module `otoole.cli`), 23

`create_datafile()` (in module `otoole.preprocess`), 20

`create_datapackage()` (in module `otoole.preprocess`), 20

`create_res()` (in module `otoole.visualise`), 22

`create_res()` (in module `otoole.visualise.res`), 22

`create_schema()` (in module `otoole.validate`), 24

D

`datapackage2res()` (in module `otoole.cli`), 23

`draw_graph()` (in module `otoole.visualise.res`), 22

G

`generate_csv_from_excel()` (in module `otoole.preprocess`), 20

`generate_csv_from_excel()` (in module `otoole.preprocess.excel_to_osemosys`), 18

`generate_package()` (in module `otoole.preprocess.create_datapackage`), 14

`get_parser()` (in module `otoole.cli`), 23

L

`load_datapackage()` (in module `otoole.visualise.res`), 22

`load_parameter_definitions()` (in module `otoole.preprocess.datafile_to_datapackage`), 15

M

`main()` (in module `otoole.cli`), 23

`main()` (in module `otoole.preprocess.create_datapackage`), 14

`main()` (in module `otoole.preprocess.longify_data`), 19

`main()` (in module `otoole.preprocess.narrow_to_datafile`), 19

O

`otoole` (module), 25

`otoole.cli` (module), 23

`otoole.preprocess` (module), 20

`otoole.preprocess.create_datapackage` (module), 14

`otoole.preprocess.datafile_to_datapackage` (module), 15

`otoole.preprocess.excel_to_osemosys`
 (*module*), 16
`otoole.preprocess.longify_data` (*module*),
 19
`otoole.preprocess.narrow_to_datafile`
 (*module*), 19
`otoole.results` (*module*), 22
`otoole.results.convert` (*module*), 20
`otoole.validate` (*module*), 23
`otoole.visualise` (*module*), 22
`otoole.visualise.res` (*module*), 22

P

`process_line()` (*in module otoole.results.convert*),
 21

R

`read_config()` (*in module*
 otoole.preprocess.excel_to_osemosys), 18
`read_in_datafile()` (*in module*
 otoole.preprocess.datafile_to_datapackage), 15
`read_narrow_csv()` (*in module*
 otoole.preprocess.narrow_to_datafile), 19
`read_packaged_file()` (*in module otoole*), 25
`read_validation_config()` (*in module*
 otoole.validate), 24
`RegionTechnology` (*class in otoole.results.convert*),
 21
`RegionTimeSliceTechnologyMode` (*class in*
 otoole.results.convert), 21

V

`validate()` (*in module otoole.validate*), 24
`validate_contents()` (*in module*
 otoole.preprocess.create_datapackage), 14
`validate_fuel_name()` (*in module*
 otoole.validate), 24

W

`write_out_dataframe()` (*in module*
 otoole.preprocess.longify_data), 19
`write_parameter()` (*in module*
 otoole.preprocess.narrow_to_datafile), 19
`write_set()` (*in module*
 otoole.preprocess.narrow_to_datafile), 20