Using the python LCOE Model

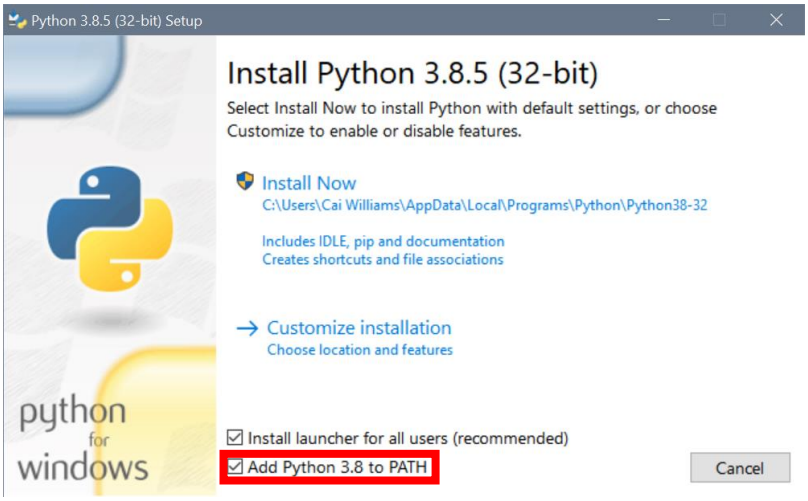
Installing Python

The program was produced with Python 3.9, the current latest version of Python. As all Libraries used are actively maintained at the time of writing, the latest version of Python at the time of reading can be installed.

The latest version of the python can be found at the following link for your operating system.

https://www.python.org/downloads/

When installing python please ensure that you have added python 3.9 to path by ensuring the tick box outlined in red is ticked. After ensuring this the install now option can be taken.



There are a 64-bit and 32-bit versions of Python, but for our purposes running the 32-bit on a 64-bit operating system is entirely adequate, there are no compatibility issues between the 32-bit and 64-bit versions.

Please restart your system before proceeding further.

Installing Visual Studio Code

Visual Studio Code is a code editor with a built-in terminal which can be used to develop with multiple languages. Visual Studio Code is used to run the model in a more user-friendly manner than simply using the windows terminal.

The latest version for Visual Studio Code can be found at the flowing link for your operating system.

https://code.visualstudio.com/download

In this case, the installation dialog can simply be followed in it default state.

Installing the Model

The model is currently hosted on GitHub and is freely available to download as a .zip then extracted to an appropriate location; this folder may then be opened in visual studio code. This folder is considered to be the working directory of the model.

The model, provided in the .zip file, may simply be extracted to the appropriate location; this folder may then be opened in visual studio code. This folder is considered to be the working directroty of the model

Open the model in Visual Studio Code, and then open one of the *.py files, Visual Studio Code will then prompt you to install the python and pylint extensions, follow these instructions.

Installing Libraries

Ensure you have the working directory of the model open in Visual Studio Code

The model makes use of several Python libraries which are not included in the python standard library and must therefore be installed before the model can be executed.

The installation of the required libraries can be completed by using the following terminal command.

```
pip3 install -r requirements.txt
```

Some libraries may have to be installed through downloading the correct *.whl for your system. If a library fails to install itself or a prerequisite take note and download the appropriate *.whl file from the link bellow. Once all libraries are installed you may now run the model.

https://www.lfd.uci.edu/~gohlke/pythonlibs/

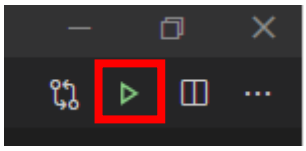
Running the Model

Ensure you have the working directory of the model open in Visual Studio Code

When first install the model is set up to replicate the results found in RandomLocsResults.csv. I would recommend first running this default setup to ensure the model is operating correctly. This may be done by comparing Results.csv against RandomLocResutls.csv.

In order to run the model, enter the following company it to the terminal, or alternatively press the “run in terminal” button.

```
Python3 RunTrindod.py
```



Four functions are important to the operation of the model.

```
experiment = LCOERun(experiment directory, paneldatafile)
```

This function initialises the model, and here you will declare the directory of the experiment and of the panel data file.

```
experiment.Run()
```

This function will run the experiment which the model was initialised with as described in its respective .json file, generating a new .JBS file each time it is executed therefore results may differ depending on how the experiment is configured.

```
experiment.ReRun()
```

This function will re-run the experiment as described in the .JBS file found in the experiment's directory. The results should be identical every time, regardless of how the experiment is configured.

```
experiment.ModPop(Property, New Value)
```

This function allows for the a pre-existing .JBS to be modified a property at a time. The properties available for modification are those described in the experiment file which are assigned a single value. Changes made are permanent unless, changes are undone the ModPop function. ReRun and ModPop may be used in conjunction with each other to modify the properties of an experiment utilising random locations without generating new random locations.

Setting up custom experiments

Inputs

The Inputs for the experiments are defined by the “Experiment Name”.json file. From the lists defined in the .json file, every permutation is defined and saved in “Experiment Name”.csv. From every line in the .csv a python object in generated where the associated files are loaded and saved as “Experimental Name”.JBS file.

ProjectName
Name Explanation: What is the name of the project?
Valid Inputs: Anything which is a valid file name

PanType
Name Explanation: Panel ID Number
Valid Inputs: Number e.g. 4678

InvLif
Name Explanation: Inverter Life
Valid Inputs: Number e.g. 10

PrjLif
Name Explanation: Project Life
Valid Inputs: Number e.g. 20

ModSta
Name Explanation: Model Start Date
Valid Inputs: Date e.g. 01/05/2014

PrjType
Name Explanation: Project Type
Valid Inputs: Type Name e.g. GroundmountPVArray

PrjLoc
Name Explanation: Project Location Name
Valid Inputs: Location Name e.g. MexicoCity

OprCosInf
Name Explanation: Operating Cost Inflation
Valid Inputs: Percentage e.g. 2.1

InvCosInf
Name Explanation: Inverter Cost Inflation
Valid Inputs: Percentage e.g. 2.1

RenCos
Name Explanation: Rental Cost
Valid Inputs: Number e.g. 0.5

Special Operators

Several properties may be described in using special operators. Two operators are described, these being the “X#” operator and “CA#” operator.

```
"PrjLoc": ["Random","X#50"]
```

The “X#” operator repeat the values at index 0 the number of times found after the operator. “X#50” would repeat the value at index 0 50 times. In the example a list of 50 random locations is generated.

```
"PanTyp": [4110,"CA#10"]
```

The “CA#” operator generates a list where the integer found at index 0, becomes the initial value of a list of 10 elements which incrementally increase. In the example given a list from device 4110 to 4120 will be generated.

Outputs

For quick generation of results from an experiment, Results.csv can be used. The model will examine the first line, the column headers, and append the file with the relevant results. Any variable in the program can be called as column headers. As can be seen in the tables following.

Column Header Category	Category Explanation
Job	Inputs Variables
Finance	Financial Model Variables
Panel	Panel Model Variables
EPC	EPC Model Variables

Job Inputs	Properties Explained
Job.ProjectName	Projects Name
Job.PanTyp	Panel Type used in project
Job.InvLif	Lifetime of the inverter
Job.PrjLif	Lifetime of the project
Job.PrjTyp	Project Type used for financial calculations
Job.PrjLoc	Location of the project
Job.OprCosInf	Operational costs inflation
Job.InvCosInf	Inverter costs inflation
Job.RenCos	Rental Costs
Job.TimStp	Timestep used by the model
Job.Tech	Low light enhancment profile used
Job.Tilt	Tilt of the panels
Job.Latitude	Latitude of the project
Job.Longitude	Longitude of the project
Job.IRR	Internal Rate of Return
Job.Elevation	Height of lowest edge of panel
Job.Spacing	Spacing between the panels
Job.PanelID	ID of panel in panel data
Job.Type	Name of panel in panel data
Job.Life	Lifetime of panel
Job.Burn-in	Burn-in coefficient of the panel
Job.Long-termDegradation	Long term degradation coefficient of the panel
Job.Burn-inPeakSunHours	Burn-in PSH, energy to of overcome burn-in
Job.Cost	Panel cost
Job.PowerDensity	Power density of panel
Job.EnergyEfficiency	Power Conversion Efficiency of panel
Job.0	Low light enhancment coefficient 0
Job.1	“ ” 1
Job.2	“ ” 2
Job.3	“ ” 3
Job.4	“ ” 4
Job.5	“ ” 5
Job.6	“ ” 6
Job.ProjectNameT	Name of project used as baseline for calculations

Job.PVSize	Initial capacity of baseline project
Job.SystemArea	Area of baseline project
Job.ProjectYield	Yield of baseline project
Job.PanelSize	Size of Panels used in baseline project
Job.Location	Name of Location of baseline project
Job.Design	Design cost of baseline project
Job.Construction	Construction cost of baseline project
Job.Framing	Framing cost of baseline project
Job.DCcabling	DC cabling cost of baseline project
Job.Accabling	AC cabling cost of baseline project
Job.CivilWork(Panels)	Civil Work (Panel) cost of baseline project
Job.CivilWork(general)	Civil Work (general) cost of baseline project
Job.PVPanels	PV Panels cost of baseline project
Job.FixedProjectCosts	Fixed project costs of baseline project
Job.Freight(Panels)	Freight costs of Panels of baseline project
Job.Freight(other)	Freight costs of other of baseline project
Job.Inverters	Inverters cost of baseline project
Job.Controls	Controls cost of baseline project

Finance Model	**Properties Explained**
Finance.DCR	Discount Rate
Finance.InitialCost	Initial Costs
Finance.InstallationCostExcPanels	Installation Cost, Excluding Panels
Finance.InterestDevisor	Changes time scale of Interest
Finannce.InverterCostAsPercentageofCiepPrice	Inverter Cost as Percentage Installation exp Panels
Finance.InverterCostInflation	Inverter Cost Inflation
Finance.LCOE	Levelised Cost of Energy
Finance.NewArea	New Area
Finance.NewPrice	New Price
Finance.OperationCostInflation	Operational Cost Inflation
Finance.PVSize	Project Capacity
Finance.PanelCost	Panel Cost

EPC Model	**Properties Explained**
EPC.EqRatingPanels	Equivalent Rating of Panels
EPC.InstallCostPerPanel	Installation Cost per Panel
EPC.InstallationCostExcPanels	Installation Cost, Excluding Panels
EPC.InverterCost	Inverter Cost
EPC.InverterCostAsPercentofCiepPrice	Inverter Cost as Percentage Installation exp Panels
EPC.NewArea	New Area
EPC.NewPrice	New Price
EPC.NumberOfPanels	Number of Panels
EPC.OldArea	Old Area
EPC.OriginalCost	Original Cost
EPC.PanelCost	Panel Cost
EPC.PanelCost2	Total Panel Costs

EPC.PanelSize	Panel Size
EPC.PriceExcludingPanels	Price Excluding Panels
EPC.RequiredNumberPanels	Required Number of Panels

Panel Model	**Properties Explained**
Panel.CPSH	Cumulative peak sun hours experienced by panels
Panel.Capacity	Maximum capacity of farm
Panel.Dates	Time steps used by model
Panel.EM	Effective Multiplier experienced by panels
Panel.EffectiveCapacity	Effective Capacity of farm
Panel.Irradiance	Irradiance experienced by panels
Panel.Lifetime	Remaining Lifetime of panels
Panel.PSH	Peak sun hours experienced by panels
Panel.PVGen	Energy generated by panels
Panel.StateOfHealth	Stated of Health of panels
Panel.Yield	KWh/KWp of farm

As written of when a Panel.* output is called the average value with zeros excluded is returned this may be modified on line 796 of Trindod.py

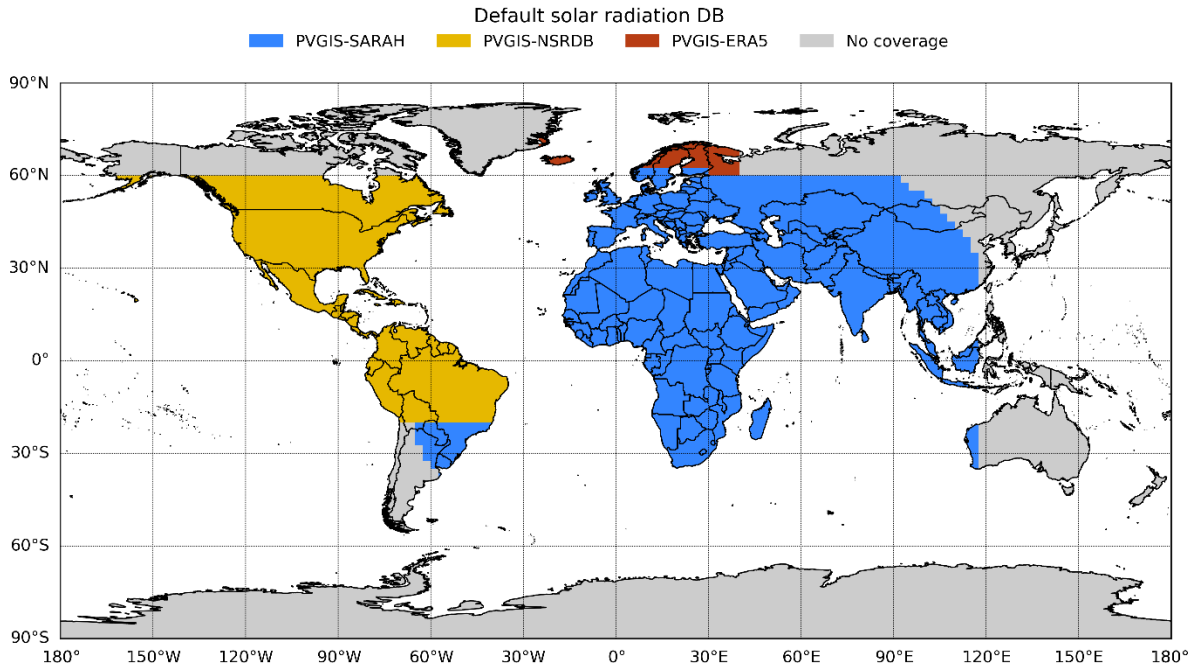
Adding Locations

Locations yield and sun hours, as well as longitude and latitude, are stored under ~/Data/Location Each .json file represents a location. An empty template has been included, named Template.json. For correct formatting inspect other locations for insight, but anywhere an “x” in placed in the templated a float must be entered.

Yield values can be found from: https://globalsolaratlas.info/map Generating a .XLSX report for a Ground-mounted large-scale project will yield the appropriate yield values.

Peak Sun hours can be found form: https://power.larc.nasa.gov/data-access-viewer/ Selecting the “Monthly & Annual” Temporal Average, and then a single location and the appropriate year and output format (CSV is suggested). In parameters select “Clear Sky Surface Shortwave Downward Irradiance”

Locations outside the range covered by PVGIS may not use the hourly timescale, a monthly timescale should be used in this case, the range covered by PVGIS can be seen in the figure bellow.



Modifying Range of Random Locations

The range of latitudes and longitudes which may be generated by random may be changed but require a change to the code which can be found on lines 196 and 197. By default, the range is 60^{o}N to 35^{o}S and 20^{o}E to 30^{o}E.