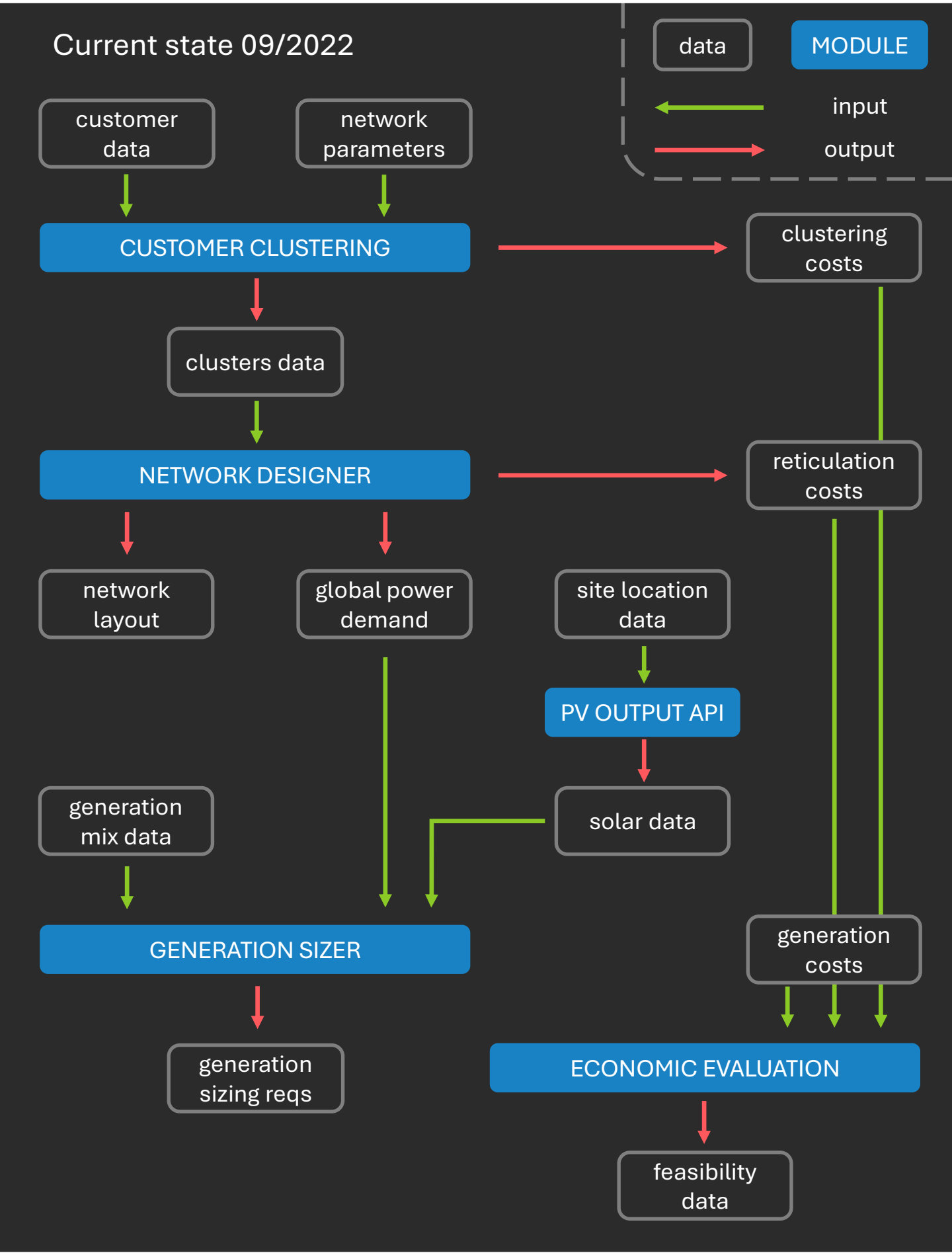
POWERGEN-PY

by Energy for Development

Documentation Manual

CONTENTS


Current state 09/2022
data
MODULE
input
output
customer data
network parameters
CUSTOMER CLUSTERING
clustering costs
clusters data
NETWORK DESIGNER
reticulation costs
network layout
global power demand
site location data
PV OUTPUT API
generation mix data
solar data
GENERATION SIZER
generation costs
generation sizing reqs
ECONOMIC EVALUATION
feasibility data

CustomerClustering class

from customer_clustering.py

class customer_clustering.**CustomerClustering**(*init_cluster, network_voltage, pole_cost, pole_spacing, resistance_per_km, current_rating, cost_per_km, max_voltage_drop=None, max_distance=None*)

Parameters

init_cluster : *InitCluster*
InitCluster object which initially pools all customers together.
max_connections : *int*
Maximum customers allowed per cluster.
network_voltage : *float*
Voltage at which network operates.
pole_cost : *float*
Cost of electrical pole which will be placed at centroid location of cluster and to support line.
pole_spacing : *flaot*
Space between each electrical pole in meters.
resistance_per_km : *float*
Resistance per kilometer of cable used in ohm/km.
current_rating : *float*
Cable's max current rating.
cost_per_km : *float*
Cable's cost per kilometer.
max_voltage_drop : *float, optional*
Maximum voltage drop allowed between pole and customer. If None then maximum voltage drop is dictated by voltage regulation. The default is *None*.
max_distance : *float, optional*
Maximum distance allowed between pole and customer in meters.
The default is *None*.

Attributes

clusters : *list*
List of cluster objects.
total_cable_length : *float*
Total cable length of all clusters combined.
total_cost : *float*
Combined capital cost of cables and poles.

CustomerClustering class

from customer_clustering.py

class customer_clustering.**CustomerClustering**(*init_cluster, network_voltage, pole_cost, pole_spacing, resistance_per_km, current_rating, cost_per_km, max_voltage_drop=None, max_distance=None*)

Methods

import_from_csv(*filename, network_voltage, pole_cost, pole_spacing, resistance_per_km, current_rating, cost_per_km, scale_factor=1, max_voltage_drop=None, max_distance=None*)

Description	Imports customer data from CSV file (must use specific template, see template generator)
Parameters	**filename** : *str* CSV file location. [Remaining are same as initialisation parameters. See Page 2]
Returns	**self** : *object* Returns CustomerClustering object with specified parameters.

cluster(*max_customers=6*)

Description	Compute customer clustering using k-means clustering. Calculates total cost as sum of pole cost and cable cost.
Parameters	**max_customers** : *int* Maximum number of customers per cluster.
Returns	**self** : *object* Returns CustomerClustering object with clustered customers.

InitCluster & Customer classes

from customer_clustering.py

class customer_cluster.**InitCluster**(*customers)*

Description	Special cluster object used for first cluster created. Centroid is automatically calculated at creation.
Parameters	**customers** : *array-like* 1D array of Customer objects.
Attributes	**customers** : *list* 1D array of Customer objects. **position** : *tuple* X and Y coordinates of Cluster. **distance** : *ndarray of shape (customers,)* 1D array containing distance between customers and centroid.

class customer_cluster.**Customer**(*customer_id,position,power_demand)*

Description	Customer object for clustering algorithm.
Parameters	**customer_id** : *str* Customer Identifier. **position** : *array-like of shape (1,2)* X and Y coordinates of customer. **power-demand** : *array-like of shape (1,)* 1D array of customer's hourly power demand.

NetworkDesigner class

from network_designer.py

class network_designer.**NetworkDesigner**(*self, source_location, nodes_locations, nodes_power_dem, network_voltage, pole_cost, pole_spacing, res_per_km, max_current, cost_per_km, scl=1, max_V_drop=None, node_ids=None, V_reg=6*)

Parameters

source_location : *array-like*
1x2 array containing X and Y coordinates of source.

nodes_locations : *array-like*
Array of 1x2 arrays containing X and Y coordinates of nodes.

nodes_power_dem : *array-like*
Array of arrays containing power demands (W) of each node in network.

network_voltage : *float*
Operating voltage of network.

pole_cost : *float*
Cost of single electrical pole.

pole_spacing : *float*
Spacing between poles in meters.

res_per_km : *float*
Cable's resistance per kilometer (ohm/km).

max_current : *float*
Cable's maximum current rating (A).

cost_per_km : *float*
Cable's cost per km.

scl : *float, optional*
Scales coordinates of source and nodes by chosen amount. The default is *1*.

max_V_drop : *flaot, optional*
Maximum voltage drop. If not specified value dictated by voltage regulation. The default is *None*.

node_ids : *array-like, optional*
Array containing node identifiers. The default is None.

V_reg : *float, optional*
Maximum voltage drop as percentage of network voltage. The default is *6*.

NetworkDesigner class

from network_designer.py

class network_designer.**NetworkDesigner**(*self, source_location, nodes_locations, nodes_power_dem, network_voltage, pole_cost, pole_spacing, res_per_km, max_current, cost_per_km, scl=1, max_V_drop=None, node_ids=None, V_reg=6*)

Attributes

nodes : *list*
List of nodes in network.
total_length : *float*
Total length of cables connecting nodes.
line_cost : *float*
Cables total cost.
num_poles : *float*
Number of required electrical poles.
poles_cost : *float*
Poles total cost.
total_cost : *float*
Total network cost (poles and cables).
total_Pdem : *ndarray of shape (1,)*
Network's total hourly power demand.
distances : *ndarray of size (nodes,nodes)*
Distances matrix. Distances between each node
connections : *ndarray of size (nodes,nodes)*
Connection matrix. Populated with distances between nodes where connection made.
Vdrop_max : *float*
Maximum voltage drop allowed in network.
max_V_drop : *flaot, optional*
Maximum voltage drop. If not specified value dictated by voltage regulation. The default is *None*.
node_ids : *array-like, optional*
Array containing node identifiers. The default is None.
V_reg : *float, optional*
Maximum voltage drop as percentage of network voltage. The default is *6*.

NetworkDesigner class

from network_designer.py

class network_designer.**NetworkDesigner**(*self, source_location, nodes_locations, nodes_power_dem, network_voltage, pole_cost, pole_spacing, res_per_km, max_current, cost_per_km, scl=1, max_V_drop=None, node_ids=None, V_reg=6*)

Methods

import_from_csv(*filename, network_voltage, pole_cost, pole_spacing, res_per_km, max_current, cost_per_km, scl=1, max_V_drop=None, V_reg=6***)**

Description	Imports source location and nodes locations and power demands from specified CSV file.
Parameters	**filename** : *str* CSV file location. [Remaining are same as initialisation parameters. See Page 5]
Returns	**self** : *object* Returns NetworkDesigner object with specified parameters and data extracted from CSV file.

build_network()

Description	Build network based on parameters assigned when initialised.
Returns	**self** : *object* Returns NetworkDesigner with completed connections.

draw_graph(*save=False*)

Description	Imports source location and nodes locations and power demands from specified CSV file.
Parameters	**save** : *bool* If true saves graph image. The default is False.
Returns	**self** : *object* Returns NetworkDesigner object with specified parameters and data extracted from CSV file.

pv_output function

from pvoutput.py

function pvoutput.**pv_output**(*lati, long, capacity, year=2019, auto_dataset=True, dataset="merra2", system_loss=0, auto_tilt=True, tilt=40, azim=180*)

Description	Retrieves PV panel output in Watts from renewables.ninja API.