Vietnam Transport Risk Analysis Documentation

Oxford Infrastructure Analytics

Sep 08, 2023

Contents

github oi-analytics/vietnam-transport[1] This documentation describes the modelling and analysis of climate-related risks to transport networks in Vietnam.

The modelling and analysis aims to support decision-making by identifying the performance of adaptation options under current and future scenarios. It comprises a network flow model, generation of failure scenarios, economic impact assessment, and cost-benefit analysis of adaptation options.

[1] https://github.com/oi-analytics/vietnam-transport/

CHAPTER 1

Contents

1.1 Installation

J.W. (2020) Addressing Climate Change in Transport: Volume 2: Pathway to Resilient Transport. World Bank, Washington DC. DOI: 10.1596/32412[2]

1.1.1 Requirements

Python and libraries

Python version 3.6 is required to run the scripts in this project. We suggest using miniconda[3] to set up an environment and manage library dependencies.

Create a conda environment from the `environment.yml` definition:

```
conda env create -f environment.yml
conda install python-igraph
```

See http://igraph.org/python/ for instructions on Windows installation of `python-igraph`.

Activate the environment:

```
conda activate vietnam-transport
```

Set up the `vtra` package (this project) for development use:

```
python setup.py develop
```

GAMS

The economic model uses GAMS[4] (General Algebraic Modeling System) via its python API. GAMS provide installation and licensing[5] instructions.

[2] http://dx.doi.org/10.1596/32412
[3] https://conda.io/miniconda.html
[4] https://www.gams.com/
[5] https://www.gams.com/latest/docs/UG_MAIN.htm

1.1.2 Configuration

1.2 Collected Data

Important:

- This section describes collected datasets that are used to create data for the Vietnam Transport Risk Analysis (VTRA)
- The datasets listed here are specfic to Vietnam and are used as inputs to data in the Processed Data Assembly steps
- To implement the VTRA pre-processing without any changes in existing codes, all data described here should be created and stored exactly as indicated below

1.2.1 Networks

1. **All pre-processed networks data are stored:**
    - In sub-folders in the file path - `/data/pre_processed_networks_data/`
    - As Shapefiles with of network nodes and edges
    - The names of files and folders are self-explanatory
    - See `/data/pre_processed_networks_data/networks_description.xlsx` for details of all shapefiles
2. **All nodes should have the following attributes:**
    - `node_id` - String Node ID
    - `geometry` - Point geometry of node with projection ESPG:4326
    - variable list of attributes depending upon sector
3. **All edges should have the following attributes:**
    - `edge_id` - String edge ID
    - `g_id` - Integer edge ID
    - `from_node` - String node ID that should be present in node_id column
    - `to_node` - String node ID that should be present in node_id column
    - `geometry` - LineString geometry of edge with projection ESPG:4326
    - variable list of attributes depending upon sector

Note: We assume that networks are provided as topologically correct connected graphs: each edge is a single LineString (may be straight line or more complex line), but must have exactly two endpoints, which are labelled as `from_node` and `to_node` (the values of these attributes must correspond to the `node_id` of a node).

Wherever two edges meet, we assume that there is a shared node, matching each of the intersecting edge endpoints. For example, at a t-junction there will be three edges meeting at one node.

1.2.2 Cost attributes

1. **Data to assign transport costs to network edges are stored:**
    - In the file in path - `/data/pre_processed_networks_data/mode_costs.xlsx`
    - As Excel sheets
2. **All cost estimtates should have the following attributes:**

- `time_cost_usd` - Float values of rate of time
- `tariff_min_vnd` - Float values minimum tariff rate in VND/ton-km (VND/ton for multi)
- `tariff_max_vnd` - Float values maximum tariff rate in VND/ton-km (VND/ton for multi)
- `tariff_min_usd` - Float values minimum tariff rate in USD/ton-km (USD/ton for multi)
- `tariff_max_usd` - Float values maximum tariff rate in USD/ton-km (USD/ton for multi)
- attributes to decide how the costs are allocated to network edges (if none then all edges have same criteria)

1.2.3 Road design attributes

1. **Data to assign characteristics to roads are stored:**
   - In the file in path - `/data/pre_processed_networks_data/road_properties.xlsx`
   - As Excel sheets
   - See `/data/pre_processed_networks_data/road_properties.xlsx` for data description

1.2.4 VITRANNS2 OD data

1. **VITRANSS2 province-level OD matrices are stored:**
   - In the path - `data/OD_data/`
   - As Excel sheets
   - `goods` sheet gives OD values by commodity
   - `modes` sheet gives OD values by mode
2. **Aggregated goods-wise province-level national OD matrices have attributes:**
   - `o` - Integer IDs of origin Provinces
   - `d` - Integer IDs of of destination Provinces
   - `name o` - String names of origin Provinces
   - `name d` - String names of destination Provinces
   - `commodity_names` - Float values of daily tonnages of commodities/industries between OD Provinces
3. **Aggregated mode-wise province-level national OD matrices have attributes:**
   - `o` - Integer IDs of origin Provinces
   - `d` - Integer IDs of of destination Provinces
   - `name o` - String names of origin Provinces
   - `name d` - String names of destination Provinces
   - `mode_names` - Float values of daily tonnages along modes between OD Provinces

1.2.5 IFPRI crop data

1. **IFPRI crop datasets are stored:**
   - In the path - `data/Agriculture_crops/`
   - As GeoTiff files
   - Only files with names `SPAM_P_crop name_ver3.tif` are used

- See Excel sheet in path `data/Agriculture_crops//crop_data/crop_unit_costs.xlsx` for costs of crops

2. **All crop GeoTiff datasets should have attributes:**
    - values greater than 0
    - raster grid geometry
    - projection systems: Default assumed = EPSG:4326

1.2.6 RiceAtlas data

1. **RiceAtlas datasets are stored:**
    - In the path - `data/rice_atlas_vietnam/`
    - As Shapefiles
    - Only the file `rice_production.shp` is used
2. **The essential attributes in the dataset are listed below. See the data for all attributes:**
    - `sub_region` - String names of Provinces in English
    - `P_Jan`, ..., `P_Dec` - Column names with float tonnage produced in each month from January to December
    - `geometry` - Polygon geometries of Provinces

1.2.7 Points of interest data

1. **Locations of populations, commune, district, province center committee points datasets are stored:**
    - In the path - `data/Points_of_interest/`
    - As Shapefiles
2. **The essential attributes in all the dataset are listed below. See the data for all attributes:**
    - `geometry` - Point geometry with projection ESPG:4326

1.3 Processed Data Assembly

Important:

- This section describes processed datasets that are used as inputs in Analysis and Results steps of the Vietnam Transport Risk Analysis (VTRA)
- The formats and attributes created in these datasets form the essential inputs for implementing the rest of the VTRA model
- To implement the VTRA without any changes in existing codes, all data described here should be created and stored exactly as indicated below

1.3.1 Networks

1. **All finalised networks data are stored:**
    - In the file path - `/data/post_processed_networks/`
    - As Excel sheets with post-processed network nodes and edges
    - As Shapefiles with post-processed network nodes and edges
2. **All nodes have the following attributes:**

- `node_id` - String Node ID
- `name` - String name in Vietnamese/English
- `tons` - Float assigned cargo freight tonnage using node
- `population` - Float assigned passenger/population number using node
- `capacity` - Float assigned capacity in tons/passenger numbers/other units
- `geometry` - Point geometry of node with projection ESPG:4326

3. **Attributes only present in inland and coastal port nodes:**
    - `port_type` - String name of type of port: inland or sea
    - `port_class` - String name of class of port: class1A (international) or class1 (domestic hub)
4. **All edges have the following attributes:**
    - `edge_id` - String edge ID
    - `g_id` - Integer edge ID
    - `from_node` - String node ID that should be present in node_id column
    - `to_node` - String node ID that should be present in node_id column
    - `geometry` - LineString geometry of edge with projection ESPG:4326
    - `terrain` - String name of terrain of edge
    - `level` - Integer number for edge level: National, Provincial, Local, etc.
    - `width` - Float width in meters of edge
    - `length` - Float estimated length in kilometers of edge
    - `min_speed` - Float estimated minimum speed in km/hr on edge
    - `max_speed` - Float estimated maximum speed in km/hr on edge
    - `min_time` - Float estimated minimum time of travel in hours on edge
    - `max_time` - Float estimated maximum time of travel in hours on edge
    - `min_time_cost` - Float estimated minimum cost of time in USD on edge
    - `max_time_cost` - Float estimated maximum cost of time in USD on edge
    - `min_tariff_cost` - Float estimated minimum tariff cost in USD on edge
    - `max_tariff_cost` - Float estimated maximum tariff cost in USD on edge
    - `vehicle_co` - Integer number of daily vehicle counts on edge
4. **Attributes only present in province and national roads edges:**
    - `surface` - String value for surface
    - `road_class` - Integer between 1 and 6
    - `road_cond` - String value: paved or unpaved
    - `asset_type` - String name of type of asset

1.3.2 OD matrices

1. **All finalised OD matrices are stored:**
    - In the path - `/results/flow_ods/`
    - As Excel sheets

The essential attributes in these OD matrices are listed below. See the data for all attributes

2. **All node-level national OD matrices contain mode-wise and total OD flows and should have attributes:**
    - `origin` - String node IDs of origin nodes
    - `destination` - String node IDs of destination nodes
    - `o_region` - String names of origin Provinces
    - `d_region` - String names of destination Provinces
    - `min_tons` - Float values of minimum daily tonnages between OD nodes
    - `max_tons` - Float values of maximum daily tonnages between OD nodes
    - `commodity_names` - Float values of daily min-max tonnages of commodities/industries between OD nodes: here based on VITRANSS2 and IFPRI data
3. **All aggregated province-level national OD matrices contain mode-wise and total OD flows and should have attributes:**
    - `o_region` - String names of origin Provinces
    - `d_region` - String names of destination Provinces
    - `min_tons` - Float values of minimum daily tonnages between OD Provinces
    - `max_tons` - Float values of maximum daily tonnages between OD Provinces
    - `commodity_names` - Float values of daily min-max tonnages of commodities/industries between OD Provinces: here based on VITRANSS2 and IFPRI data
4. **All province OD matrices contain province-wise OD flows and should have attributes:**
    - `origin` - String node IDs of origin nodes
    - `destination` - String node IDs of destination nodes
    - `min_croptons` - Float values of minimum daily tonnages of crops between OD nodes
    - `max_croptons` - Float values of maximum daily tonnages of crops between OD nodes
    - `min_netrev` - Float values of minimum daily revenue of all firms between OD nodes
    - `max_netrev` - Float values of maximum daily revenue of all firms between OD nodes

1.3.3 Hazards

1. **All hazard datasets are stored:**
    - In sub-folders in the path - `/data/Hazard_data/`
    - As GeoTiff files
    - See `/data/hazard_data/hazard_data_folder_data_info.xlsx` for details of all hazard files
2. **Single-band GeoTiff hazard raster files should have attributes:**
    - values - between 0 and 1000
    - raster grid geometry
    - projection systems: Default assumed = EPSG:32648
3. **Multi-band GeoTiff hazard raster files should have attributes:**
    - 3-bands
    - values - in each band between 0 and 255
    - raster grid geometry
    - projection systems: Default assumed = EPSG:32648

1.3.4 Administrative Areas with Statistics

1. **Vietnam boundary datasets are stored:**
    - In the path - `/data/Vietnam_boundaries/who_boundaries/`
    - In the path - `/data/Vietnam_boundaries/boundaries_stats/`
    - As Shapefiles
2. **Global boundary dataset for map plotting are stored:**
    - In the path - `/data/Global_boundaries/Natural_Earth/`

The essential attributes in the Vietnam boundary datasets are listed below. See the data for all attributes

3. **All Vietnam province boundary datasets should have the attributes:**
    - `name_eng` - String names of administrative boundary in English
    - `od_id` - Integer IDs matching ID's in VITRANSS2 OD data
    - `geometry` - Polygon geometries of boundary with projection ESPG:4326
4. **All Vietnam commune boundary datasets should have attributes:**
    - `commune_id` - Integer IDs of commune
    - `name_eng` - String names of commune in English
    - `district_i` - Integer IDs of district of commune
    - `dis_name_e` - String names of district in English
    - `province_i` - Integer IDs of province of commune
    - `pro_name_e` - String names of province in English
    - `population` - Float values of population in commune
    - `nfirms` - Float values of number of firms in commune
    - `netrevenue` - Float values of netrevenue of commune
    - `nongnghiep` - Float fractions of agriculture firms in commune
    - `geometry` - Polygon geometry of boundary with projection ESPG:4326
5. **All global boundary datasets should have attributes:**
    - `name` - String names of boundaries in English
    - `geometry` - Polygon geometry of boundary with projection ESPG:4326

1.3.5 Macroeconomic Data

1. **For the macroeconomic analysis we use the national IO table for Vietnam:**
    - In the file in path - `data/economic_IO_tables/IO Table 2012 English.xlsx`
    - We use the sheet `IO Core` in our analysis.

1.3.6 Adaptation Options

1. **All adaptation options input datasets are stored:**
    - In the file - `/data/Adaptation_options/adaptation_costs_road_types.xlsx`
    - As Excel files
2. **Excel sheet `options` explains:**
    - `adaptation_group` - String names of the type of adaptation strategy
    - `option_code` - String codes of the option

- `item` - String descriptions of the option

3. **Excel sheets `costs_district_mountain`, `costs_district_flat`, `costs_national_mountain`, `costs_n`**

   - `adaptation_group` - String names of the type of adaptation strategy
   - `option_code` - String codes of the option
   - `item` - String descriptions of the option
   - `estimated_amount_fraction` - Float ratios of road length over which the option is implemented
   - `Estimated` length - Float assumed road lengths in example case
   - `factor` - Float factor multiplied to length based on rates and option
   - `factor_unit` - String factor units
   - `unit` - String dimension units of measurement of option
   - `rate` - Float unit rates of option in USD/unit
   - `total` cost - Float product of Estimated length*factor*rate
   - `design_assumption` - String explanations of some design assumptions
   - `comment` - String general comments

4. **Excel sheet `rehabilitation_costs` explain:**

   - `Road Type` - String type of road
   - `terrain` - String terrian of road
   - `basic_cost` - Float current costs of rehabilitation in USD/km of road
   - `design_width` - Float design width of road
   - `road_class` - Integer class for National Roads
   - `road_level` - Integer level for Province Roads
   - `unit` - String unit of cost

5. **Excels sheets `maintenance_mountain` and `maintenance_flat` explain:**

   - `adaptation_group` - String names of the type of adaptation strategy
   - `option_code` - String codes of the option
   - `item` - String descriptions of the option
   - `recurrent_cost` - Float costs of recurrent maintenance
   - `periodic_cost` - Float costs of periodic maintenance
   - `recurrent_factor` - Float factor of recurrent maintenance
   - `periodic_factor` - Float factor of periodic maintenance
   - `recurrent_maintain_time` - Float times of recurrent maintenance in years
   - `periodic_maintain_time` - Float times of periodic maintenance in years
   - `recurrent_cost_unit` - String unit of recurrent maintenance costs
   - `periodic_cost_unit` - String unit of periodic maintenance costs
   - `comment` - String general comments

1.4 Analysis and Results

Important:

- This section describes the steps Analysis and Results steps of the Vietnam Transport Risk Analysis (VTRA)
- To implement the VTRA without any changes in existing codes, all data described here should be created and stored exactly as indicated below

1.4.1 Preparing Network Data

Purpose:

- Create post-processed transport networks with attributes
- From pre-processed input Shapefiles and collected network attributes data
- For all Province road networks
- For all transport modes at national scale

Execution:

- Load data as described in Collected Data[6] Networks[7], Cost attributes[8] and Road design attributes[9]
- Run `vtra.preprocess.create_transport_networks`

Result:

- Create networks with formats and attributes described in Processed Data Assembly[10] Networks[11]
- Store outputs in `/data/post_processed_networks/`

1.4.2 Preparing Hazard Data

Purpose:

- **Convert GeoTiff raster hazard datasets to shapefiles based on masking and selecting values from**
  - Single-band raster files
  - Multi-band (3-bands) raster files

Execution:

- Load data as described in Processed Data Assembly[12] Hazards[13]
- Run `vtra.preprocess.convert_hazard_data`

Result:

- **Create hazard shapefiles with names described in excel sheet in Processed Data Assembly[14] Hazards[15] and attrib**
  - `ID` - equal to 1
  - `geometry` - Polygon outline of selected hazard

[6] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html
[7] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#networks
[8] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#cost-attributes
[9] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#road-design-attributes
[10] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html
[11] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#networks
[12] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html
[13] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#hazards
[14] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html
[15] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#hazards

- Store outputs in same paths in directory `/data/Hazard_data/`

1.4.3 Preparing OD matrix Data

Purpose:

- **Create national scale OD matrices at node and province levels from:**
  - VITRANSS2 province-scale OD data
  - IFPRI crop data at 1km resolution
- **Create province scale OD matrices between roads connecting villages to nearest communes from:**
  - Net revenue estimates of commune villages
  - IFPRI crop data at 1km resolution

Execution:

- Load data as described in Networks[16], VITRANSS2 OD data[17], IFPRI crop data[18], RiceAtlas data[19], Points of interest data[20], and Administrative Areas with Statistics[21]
- For National OD matrices run *`vtra.preprocess.national_modes_od_creation`*
- For Provinces OD matrices run *`vtra.preprocess.province_roads_access_od_creation`*

Result:

- Create OD matrices with attributes described in Processed Data Assembly[22] OD matrices[23]
- Store outputs in `/results/flow_ods/`

1.4.4 Mapping Flows onto Networks

Purpose:

- **Map the national-scale OD node level matrix values to network paths**
  - For all transport modes at national scale
  - Estimate 2 values - A MIN and a MAX value of flows between each selected OD node pair
  - Based on MIN-MAX generalised costs estimates
- **Map the commune access OD node level matrix values to road network paths in Provinces**
  - For all roads in the Provinces
  - Estimate 2 values - A MIN and a MAX value of flows between each selected OD node pair
  - Based on MIN-MAX generalised costs estimates

Execution:

- Load data as described in Networks[24] and OD matrices[25]
- For National OD matrices run *`vtra.flow_mapping.national_modes_flow_paths`*
- For Provinces OD matrices run *`vtra.flow_mapping.province_roads_access_flow_paths`*

Result:

[16] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#networks
[17] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#vitranns2-od-data
[18] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#ifpri-crop-data
[19] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#riceatlas-data
[20] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/predata.html#points-of-interest-data
[21] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#administrative-areas-with-statistics
[22] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html
[23] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#od-matrices
[24] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#networks
[25] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#od-matrices

- Store flow excel outputs in `/results/flow_mapping_paths/`
- Store flow shapefiles in `/results/flow_mapping_shapefiles/`
- Store flow csv files in `/results/flow_mapping_combined/`
- **National-scale excel sheets results of flow mapping based contain attributes:**
  - `origin` - String node ID of Origin
  - `destination` - String node ID of Destination
  - `o_region` - String name of Province of Origin node ID
  - `d_region` - String name of Province of Destination node ID
  - `min_edge_path` - List of string of edge IDs for paths with minimum generalised cost flows
  - `max_edge_path` - List of string of edge IDs for paths with maximum generalised cost flows
  - `min_distance` - Float values of estimated distance for paths with minimum generalised cost flows
  - `max_distance` - Float values of estimated distance for paths with maximum generalised cost flows
  - `min_time` - Float values of estimated time for paths with minimum generalised cost flows
  - `max_time` - Float values of estimated time for paths with maximum generalised cost flows
  - `min_gcost` - Float values of estimated generalised cost for paths with minimum generalised cost flows
  - `max_gcost` - Float values of estimated generalised cost for paths with maximum generalised cost flows
  - `min_vehicle_nums` - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
  - `max_vehicle_nums` - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
  - `industry_columns` - All daily tonnages of industry columns given in the OD matrix data
- **Province-scale excel sheets with results of flow mapping based contain attributes:**
  - `origin` - String node ID of Origin
  - `destination` - String node ID of Destination
  - `min_edge_path` - List of string of edge IDs for paths with minimum generalised cost flows
  - `max_edge_path` - List of string of edge IDs for paths with maximum generalised cost flows
  - `min_netrev` - Float values of estimated daily Net Revenue for paths with minimum generalised cost flows
  - `max_netrev` - Float values of estimated daily Net Revenue for paths with maximum generalised cost flows
  - `min_croptons` - Float values of estimated daily crop tonnage for paths with minimum generalised cost flows
  - `max_croptons` - Float values of estimated daily crop tonnage for paths with maximum generalised cost flows
  - `min_distance` - Float values of estimated distance for paths with minimum generalised cost flows
  - `max_distance` - Float values of estimated distance for paths with maximum generalised cost flows

  - `min_time` - Float values of estimated time for paths with minimum generalised cost flows
  - `max_time` - Float values of estimated time for paths with maximum generalised cost flows
  - `min_gcost` - Float values of estimated generalised cost for paths with minimum generalised cost flows
  - `max_gcost` - Float values of estimated generalised cost for paths with maximum generalised cost flows
  - `min_vehicle_nums` - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
  - `max_vehicle_nums` - Float values of estimated vehicle numbers for paths with maximum generalised cost flows

1.4.5 Hazard Exposure

Purpose:

- **Intersect hazards and network line and point geometries with hazatd polygons**
  - Write final results to Shapefiles
- **Collect network-hazard intersection attributes**
  - Combine with boundary Polygons to collect network-hazard-boundary intersection attributes
  - Write final results to an Excel sheet

Execution:

- Load shapefiles data as described in Networks[26] and Hazards[27]
- Run *`vtra.failure_scenario_selection.hazards_networks_intersections`*
- Run *`vtra.failure_scenario_selection.hazards_network_intersections_results_collect`*

Result:

- Store shapefile outputs in the directory `/results/networks_hazards_intersection_shapefiles/`
- **All hazard-edge intersection shapefiles with attributes:**
  - `edge_id` - String name of intersecting edge ID
  - `length` - Float length of intersection of edge LineString and hazard Polygon
  - `geometry` - LineString geometry of intersection of edge LineString and hazard Polygon
- **All hazard-node intersection shapefile with attributes:**
  - `node_id` - String name of intersecting node ID
  - `geometry` - Point geometry of intersecting node ID
- Store summarised results in `/results/hazard_scenarios/`
- **Generate excel sheet of network-hazard-boundary intersection with attributes:**
  - `edge_id`/node_id - String name of intersecting edge ID or node ID
  - `length` - Float length of intersection of edge LineString and hazard Polygon: Only for edges
  - `province_id` - String/Integer ID of Province
  - `province_name` - String name of Province in English
  - `district_id` - String/Integer ID of District

[26] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#networks
[27] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#hazards

  - `district_name` - String name of District in English
  - `commune_id` - String/Integer ID of Commune
  - `commune_name` - String name of Commune in English
  - `sector` - String name of transport mode
  - `hazard_type` - String name of hazard type
  - `model` - String name of hazard model
  - `year` - String name of hazard year
  - `climate_scenario` - String name of hazard scenario
  - `probability` - Float/String value of hazard probability
  - `band_num` - Integer value of hazard band
  - `min_val` - Integer value of minimum value of hazard threshold
  - `max_val` - Integer value of maximum value of hazard threshold

1.4.6 Hazard weights

Purpose

- Combine failure scenarios across probability levels into single value per hazard type, scenario, network link.

Execution

- Produce hazard scenarios as described above.
- Common functions are defined in *vtra.failure_scenario_selection.hazard_network_scenarios*
- For national networks, run *vtra.failure_scenario_selection.collect_network_hazard_scenarios_national*
- For provincial networks, run *vtra.failure_scenario_selection.collect_network_hazard_scenarios_provincial*

Result

- Combined scenarios in `results/hazard_scenarios/{national, provincial}_{mode}_hazard_intersections_risks.csv`
  - `edge_id` - string, name of failed edge
  - `hazard_type` - string, name of hazard
  - `model` - string, name of hazard model (if any)
  - `climate_scenario` - string, name of climate scenario (if any)
  - `year` - integer, year of hazard data
  - `{mode}_length` - float, length of edge (mode could be road, rail)
  - `min/max_band` - integer, hazard band (if any)
  - `min/max_height` - float, hazard height (if any)
  - `min/max_exposure_percent` - float, percentage of edge exposed to hazard
  - `min/max_duration_wt` - float, duration weight
  - `min/max_exposure_length` - float, length of edge exposed to hazard
  - `risk_wt` - float, risk weight
  - `dam_wt` - float, damage weight

1.4.7 Failure Analysis

Purpose:

- **Failure analysis of edges in invidiual national-scale networks**
  - To estimate flow isolations and rerouting effects on same network
- **Failure analysis of edges in national-scale networks with multi-modal options**
  - To estimate flow isolations and rerouting effects with multi-modal options
- **Failure analysis of edges in province-scale road networks**
  - To estimate changing accessibility to commune points

Execution:

- Load network and flow excel data as described in Networks[28], Mapping Flows onto Networks[29], and failure scenarios from Hazard exposure[30]
- For National networks failure analysis run `vtra.failure.failure_estimation_national`
- For National networks failure analysis with multi-modal options run `vtra.failure.failure_multi_modal_options`
- For Provincial roads failure analysis run `vtra.failure.failure_estimation_provinces`

Result:

- Store csv outputs in the directory `/results/failure_results/`
- Store shapefile outputs in `/results/failure_shapefiles/`
- **National-scale All failure scenarios results in `/results/failure_results/all_fail_scenarios/`**
  - `edge_id` - String name or list of failed edges
  - `origin` - String node ID of Origin of disrupted OD flow
  - `destination` - String node ID of Destination of disrupted OD flow
  - `o_region` - String name of Province of Origin node ID of disrupted OD flow
  - `d_region` - String name of Province of Destination node ID of disrupted OD flow
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_distance` - Float value of estimated distance of OD journey before disruption
  - `min/max_time` - Float value of estimated time of OD journey before disruption
  - `min/max_gcost` - Float value of estimated travel cost of OD journey before disruption
  - `min/max_vehicle_nums` - Float value of estimated vehicles of OD journey before disruption
  - `new_cost` - Float value of estimated cost of OD journey after disruption
  - `new_distance` - Float value of estimated distance of OD journey after disruption
  - `new_path` - List of string edge IDs of estimated new route of OD journey after disruption
  - `new_time` - Float value of estimated time of OD journey after disruption
  - `dist_diff` - Float value of Post disruption minus per-disruption distance
  - `time_diff` - Float value Post disruption minus per-disruption timee

[28] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#networks
[29] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/results.html#mapping-flows-onto-networks
[30] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/results.html#hazard-exposure

  - `min/max_tr_loss` - Float value of estimated change in rerouting cost
  - `industry_columns` - Float values of all daily tonnages of industry columns along disrupted OD pairs
  - `min/max_tons` - Float values of total daily tonnages along disrupted OD pairs
- **National-scale Isolated OD scenarios - OD flows with no rerouting options in `/results/failure_results/`**
  - `edge_id` - String name or list of failed edges
  - `o_region` - String name of Province of Origin node ID of disrupted OD flow
  - `d_region` - String name of Province of Destination node ID of disrupted OD flow
  - `industry_columns` - Float values of all daily tonnages of industry columns along disrupted OD pairs
  - `min/max_tons` - Float values of total daily tonnages along disrupted OD pairs
- **National-scale rerouting scenarios - OD flows with rerouting options in `/results/failure_results/rero`**
  - `edge_id` - String name or list of failed edges
  - `o_region` - String name of Province of Origin node ID of disrupted OD flow
  - `d_region` - String name of Province of Destination node ID of disrupted OD flow
  - `min/max_tr_loss` - Float value of change in rerouting cost
  - `min/max_tons` - Float values of total daily tonnages along disrupted OD pairs
- **National-scale min-max combined scenarios - Combined min-max results along each edge in `/results/failu`**
  - `edge_id` - String name or list of failed edges
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_tr_loss` - Float values of change in rerouting cost
  - `min/max_tons` - Float values of total daily tonnages affected by disrupted edge
- **National-scale shapefile min-max combined scenarios**
  - `edge_id` - String name or list of failed edges
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_tr_loss` - Float values of change in rerouting cost
  - `min/max_tons` - Float values of total daily tonnages affted by disrupted edge
  - `geometry` - LineString geomtry of edges
- **Province-scale all failure scenarios results in `/results/failure_results/all_fail_scenarios/`**
  - `edge_id` - String name or list of failed edges
  - `origin` - String node ID of Origin of disrupted OD flow
  - `destination` - String node ID of Destination of disrupted OD flow
  - `o_region` - String name of Province of Origin node ID of disrupted OD flow
  - `d_region` - String name of Province of Destination node ID of disrupted OD flow
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_distance` - Float value of estimated distance of OD journey before disruption
  - `min/max_time` - Float value of estimated time of OD journey before disruption

  - `min/max_gcost` - Float value of estimated travel cost of OD journey before disruption
  - `min/max_vehicle_nums` - Float value of estimated vehicles of OD journey before disruption
  - `new_cost` - Float value of estimated cost of OD journey after disruption
  - `new_distance` - Float value of estimated distance of OD journey after disruption
  - `new_path` - List of string edge IDs of estimated new route of OD journey after disruption
  - `new_time` - Float value of estimated time of OD journey after disruption
  - `dist_diff` - Float value of Post disruption minus per-disruption distance
  - `time_diff` - Float value Post disruption minus per-disruption timee
  - `min/max_tr_loss` - Float value of estimated change in rerouting cost
  - `min/max_netrev` - Float values of total daily net revenues along disrupted OD pairs
  - `min/max_tons` - Float values of total daily crop tonnages along disrupted OD pairs
  - `min_max_econ_impact` - Float values of total daily economic impact of disrupted OD pairs
- **Province-scale min-max combined scenarios - Combined min-max results oalong each edge in `/results/fail`**
  - `edge_id` - String name or list of failed edges
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_tr_loss` - Float values of estimated change in rerouting cost
  - `min/max_tons` - Float values of total daily tonnages along edge
  - `min/max_netrev` - Float values of total daily net revenues along edge
  - `min/max_econ_impact` - Float value of total daily economic impact of edge
- **Min-max combined scenarios - Combined min-max reults of total network impacts of each edge**
  - `edge_id` - String name or list of failed edges
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_tr_loss` - Float values of estimated change in rerouting cost
  - `min/max_tons` - Float values of total daily tonnages along edge
  - `min/max_netrev` - Float values of total daily net revenues along edge
  - `min/max_econ_impact` - Float value of total daily economic impact of edge
  - `geometry` - LineString geometry of edges

1.4.8 Macroeconomic loss Analysis

Purpose:

- **Macroeconomic losses analysis due to edge failures in national-scale networks**
  - To estimate economic impacts of flow isolations/disruptions
  - To understand the wider economic impacts of these disruptions

Execution:

- Load data described in Macroeconomic Data[31] and OD matrices[32]

[31] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#macroeconomic-data
[32] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#od-matrices

- To create the multiregional input-output table for Vietnam, run *vtra.mrio.run_mrio*
- To perform the loss analysis, run *vtra.mria.run_mria*

Result:

- **Store the new multiregional input-output table in `/data/input_data/`**
  - **files starting with `IO_VIETNAM_*.xlsx` contain:**
    - Sheetname `T` with the full multiregional table
    - Sheetname `labels_T` with the column and row labels of matrix `T`
    - Sheetname `FD` with the final demand columns of the new table
    - Sheetname `labels_FD` with the column labels of matrix `FD`
    - Sheetname `ExpROW` with the export to the Rest of the World columns of the new table
    - Sheetname `labels_ExpROW` with the column labels of matrix `ExpROW`
    - Sheetname `VA` with the value added rows of the new table
    - Sheetname `labels_VA` with the row labels of matrix `VA`
- Store csv files in `/results/economic_failure_losses/summarized/`
- **All summarized files have the following attributes:**
  - `edge_id` - String edge IDs
  - `total_losses` - Value of the total economic losses due to the disruption of the corresponding edge ID
- Store csv files in `/results/economic_failure_losses/od_region_losses/`
- **All od_losses file have the following attributes:**
  - `edge_id` - String edge IDs
  - `region` - String name of the region
  - `dir_losses` - Value of the direct losses due to the diruption of the corresponding edge ID in the corresponding region
  - `total_losses` - Value of the total losses due to the diruption of the corresponding edge ID in the corresponding region
  - `ind_losses` - Value of the indirect losses due to the diruption of the corresponding edge ID in the corresponding region

1.4.9 Processing Failure Results

Purpose:

- Combine national-scale macroeconomic loss estimates with rerouting losses
- Estimate tonnage shifts from one mode onto others
- Combine economic impacts of partial multi-modal rerouting split

Execution:

- Load data described in Failure Analysis[33] and Macroeconomic loss analysis[34]
- Run *vtra.failure.economic_failure_combine_national*
- Run *vtra.failure.national_failure_transfers*
- Run *vtra.failure.transfer_costs_modes*

[33] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/results.html#failure-analysis
[34] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/results.html#macroeconomic-loss-analysis

Result:

- Store csv files in `/results/failure_results/minmax_combined_scenarios/`
- **Files with names `single_edge_failures_transfers_national_{mode}_{x}_percent_shift.csv`**
  - `edge_id` - String IDs of edges of all multi-modal options for flow transfer
  - `min_tons` - Float values of minimum tons shifted to edges
  - `max_tons` - Float values of maximum tons shifted to edges
- **Files with names `single_edge_failures_minmax_national_{mode}_{x}_percent_disrupt.csv`**
  - `edge_id` - String name or list of failed edges
  - `no_access` - Boolean 1 (no reroutng) or 0 (rerouting)
  - `min/max_tr_loss` - Float values of change in rerouting cost
  - `min/max_tons` - Float values of total daily tonnages affected by disrupted edge
  - `min/max_econ_loss` - Float values of total daily economic losses
  - `min/max_econ_impact` - Float values of sum of transport loss and macroeconomic loss

1.4.10 Adaptation

Purpose:

- Generate adaption scenarios/strategies and examine their costs, benefits, net present values and benefit-cost ratios
- For national or provincial roads, based on different types of hazards, road assets and climate-change conditions

Execution:

- Load data described in Networks[35], Processing Failure Results[36], and Adaptation Options[37]
- Common functions are in *`vtra.adaptation.adaptation_options`*
- Run *`vtra.adaptation.run_options_national`*
- Run *`vtra.adaptation.run_options_provincial`*

Result:

- Store results as excel sheets in `/results/adaptation_results/`
- **All adaptation results have the following attributes:**
  - `edge_id` - string, edge IDs
  - `hazard_type` - string, names of hazard types
  - `model` - string, names of hazard models
  - `climate_scenario` - string, names of climate scenarios
  - `year` - integer, values of year of hazard climate models
  - `level` - integer, road level
  - `terrain` - string, road terrain (flat/mountain)
  - `surface` - string, road surface
  - `road_class` - integer, road class (1-6)

[35] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#networks
[36] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/results.html#processing-failure-results
[37] https://vietnam-transport-risk-analysis.readthedocs.io/en/latest/data.html#adaptation-options

- `road_cond` - string, names of road conditions (paved/unpaved)
- `width` - float, edge widths in meters
- `road_length` - float, edge lengths in meters
- `min/max_band` - integer, hazard bands (3, 4 or 5)
- `min/max_height` - float, heights in meters of hazard exposure - if flooding
- `min/max_exposure_percent` - float, percent of edge length exposed to hazard
- `min/max_duration_wt` - float, fraction of maximum duration of disruption considered for edge failure scenario
- `min/max_exposure_length` - float, edge length in meters exposed to hazard
- `risk_wt` - float, weight given to estimating expected annual losses
- `dam_wt` - float, weight given to estimating expected annual damage costs
- `min/max_econ_impact` - float, minimum/maximum economic impact in USD/day
- `min/max_benefit` - float, minimum/maximum benefit over time in USD
- `min/max_ini_adap_cost` - float, minimum/maximum initial adaptation cost in USD
- `min/max_tot_adap_cost` - float, minimum/maximum total adaptation cost in USD
- **`min/max_ini_rel_share` - float, minimum/maximum initial relative shares (per cost** component)
- **`min/max_tot_rel_share` - float, minimum/maximum total relative shares (per cost** component)
- `min/max_bc_ratio` - float, minimum/maximum benefit cost ratio
- `min/max_bc_diff` - float, minimum/maximum benefit cost difference

1.5 vtra package

Vietnam Transport Risk Analysis

1.5.1 Subpackages

vtra.adaptation package

Submodules

vtra.adaptation.adaptation_options module

Estimate costs and benefits, either under fixed parameters or under a sensitivity analysis, varying the cost components.

`average_tuples`(*l*)

`calc_costs`(*x*, *param_values*, *mnt_dis_cost*, *mnt_nat_cost*, *cst_dis_cost*, *cst_nat_cost*, *brdg_cost*, *pavement*, *mnt_main_cost*, *cst_main_cost*, *discount_rates*, *discount_growth_rates*, *rehab_costs*, *min_main_dr*, *max_main_dr*, *duration_max=10*, *min_exp=True*, *national=False*, *min_loss=True*)

Estimate the total cost and benefits for a road segment. This function is used within a pandas apply

Parameters

- **`x`** – a row from the road segment dataframe that we are considering
- **`param_values`** – numpy array with a set of parameter combinations
- **`mnt_dis_cost`** – adaptation costs for a district road in the mountains

- **mnt_nat_cost** – adaptation costs for a national road in the mountains
- **cst_dis_cost** – adaptation costs for a district road on flat terrain
- **cst_nat_cost** – adaptation costs for a national road on flat terrain
- **pavement** – set of paving combinations. This corresponds with the cost table and the param_values
- **mnt_main_cost** – maintenance costs for roads in the mountains
- **cst_main_cost** – maintenance costs for roads on flat terrain
- **discount_rates** – discount rates to be used for the costs
- **discount_growth_rates** – discount rates to be used for the losses
- **rehab_costs** – rehabilitation costs after a disaster
- **min_main_dr** – discount rates for 4-year periodic maintenance
- **max_main_dr** – discount rates for 8-year periodic maintenance
- **min_exp** (*bool*[38], *optional*) – Specify whether we want to use the minimum or maximum exposure length. The default value is set to **True**
- **national** (*bool*[39], *optional*) – Specify whether we are looking at national roads. The default value is set to **False**
- **min_loss** (*bool*[40], *optional*) – Specify whether we want to use the minimum or maximum economic losses. The default value is set to **True**

Returns

- **uncer_output** (*list*) – outcomes for the initial adaptation costs of this road segment
- **tot_uncer_output** (*list*) – outcomes for the total adaptation costs of this road segment
- **rel_share** (*list*) – relative share of each factor in the initial adaptation cost of this road segment
- **tot_rel_share** (*list*) – relative share of each factor in the total adaptation cost of this road segment
- **bc_ratio** (*list*) – benefit cost ratios for this road segment

calculate_discounting_arrays(*discount_rate=12*, *growth_rate=6*)

Set discount rates for yearly and period maintenance costs

Parameters

- **discount_rate** – yearly discount rate
- **growth_rate** – yearly growth rate

Returns

- *discount_rate_norm* – discount rates to be used for the costs
- *discount_rate_growth* – discount rates to be used for the losses
- *min_main_dr* – discount rates for 4-year periodic maintenance
- *max_main_dr* – discount rates for 8-year periodic maintenance

max_tuples(*l*)

run_adaptation_calculation(*file_id*, *data_path*, *output_path*, *duration_max=10*, *discount_rate=12*, *growth_rate=6*, *read_from_file=False*)

sum_tuples(*l*)

[38] https://docs.python.org/3.6/library/functions.html#bool
[39] https://docs.python.org/3.6/library/functions.html#bool
[40] https://docs.python.org/3.6/library/functions.html#bool

vtra.adaptation.run_options_national module

Assess national adaptation options

vtra.adaptation.run_options_provincial module

Assess provincial adaptation options

vtra.failure package

Submodules

vtra.failure.economic_failure_combine_national module

Combine national-scale macroeconomic loss estimates with rerouting losses

main()
Process results

vtra.failure.failure_estimation_national module

Failure analysis of national-scale networks For transport modes at national scale:

- road
- rail

Input data requirements

1. Correct paths to all files and correct input parameters
2. Excel sheets with results of flow mapping based on MIN-MAX generalised costs estimates:
    - origin - String node ID of Origin
    - destination - String node ID of Destination
    - o_region - String name of Province of Origin node ID
    - d_region - String name of Province of Destination node ID
    - min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows
    - max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
    - min_distance - Float values of estimated distance for paths with minimum generalised cost flows
    - max_distance - Float values of estimated distance for paths with maximum generalised cost flows
    - min_time - Float values of estimated time for paths with minimum generalised cost flows
    - max_time - Float values of estimated time for paths with maximum generalised cost flows
    - min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
    - max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows
    - min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
    - max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
    - industry_columns - All daily tonnages of industry columns given in the OD matrix data
3. Shapefiles
    - edge_id - String/Integer/Float Edge ID
    - geometry - Shapely LineString geomtry of edges

Results

Csv sheets with results of failure analysis:

1. All failure scenarios
    - edge_id - String name or list of failed edges
    - origin - String node ID of Origin of disrupted OD flow
    - destination - String node ID of Destination of disrupted OD flow
    - o_region - String name of Province of Origin node ID of disrupted OD flow
    - d_region - String name of Province of Destination node ID of disrupted OD flow
    - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
    - min/max_distance - Float value of estimated distance of OD journey before disruption
    - min/max_time - Float value of estimated time of OD journey before disruption
    - min/max_gcost - Float value of estimated travel cost of OD journey before disruption
    - min/max_vehicle_nums - Float value of estimated vehicles of OD journey before disruption
    - new_cost - Float value of estimated cost of OD journey after disruption
    - new_distance - Float value of estimated distance of OD journey after disruption
    - new_path - List of string edge ID's of estimated new route of OD journey after disruption
    - new_time - Float value of estimated time of OD journey after disruption
    - dist_diff - Float value of Post disruption minus per-disruption distance
    - time_diff - Float value Post disruption minus per-disruption timee
    - min/max_tr_loss - Float value of estimated change in rerouting cost
    - industry_columns - Float values of all daily tonnages of industry columns along disrupted OD pairs
    - min/max_tons - Float values of total daily tonnages along disrupted OD pairs
2. Isolated OD scenarios - OD flows with no rerouting options
    - edge_id - String name or list of failed edges
    - o_region - String name of Province of Origin node ID of disrupted OD flow
    - d_region - String name of Province of Destination node ID of disrupted OD flow
    - industry_columns - Float values of all daily tonnages of industry columns along disrupted OD pairs
    - min/max_tons - Float values of total daily tonnages along disrupted OD pairs
3. Rerouting scenarios - OD flows with rerouting options
    - edge_id - String name or list of failed edges
    - o_region - String name of Province of Origin node ID of disrupted OD flow
    - d_region - String name of Province of Destination node ID of disrupted OD flow
    - min/max_tr_loss - Float value of change in rerouting cost
    - min/max_tons - Float values of total daily tonnages along disrupted OD pairs
4. Min-max combined scenarios - Combined min-max results along each edge
    - edge_id - String name or list of failed edges
    - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
    - min/max_tr_loss - Float values of change in rerouting cost
    - min/max_tons - Float values of total daily tonnages affted by disrupted edge

5. **Shapefile Min-max combined scenarios - Combined min-max reults along each edge**
    - edge_id - String name or list of failed edges
    - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
    - min/max_tr_loss - Float values of change in rerouting cost
    - min/max_tons - Float values of total daily tonnages affted by disrupted edge
    - geometry - Shapely LineString geomtry of edges

main()

Estimate failures

Specify the paths from where you want to read and write:

1. Input data
2. Intermediate calcuations data
3. Output results

Supply input data and parameters

1. **Names of modes** List of strings
2. **Unit weight of vehicle assumed for each mode** List of float types
3. **Range of usage factors for each mode to represent uncertainty in cost estimations** List of tuples of float types
4. **Min-max names of names of different types of attributes - paths, distance, time, cost, vehicles, tons** List of string types
5. **Names of commodity/industry columns for which min-max tonnage column names already exist** List of string types
6. **Percentage of OD flows that are assumed disrupted** List of float type
7. **Condition on whether analysis is single failure or multiple failure** Boolean condition True or False

Give the paths to the input data files:

1. Network edges Excel and shapefiles
2. OD flows Excel file
3. Costs of modes Excel file
4. Road properties Excel file
5. Failure scenarios Excel file

Specify the output files and paths to be created

vtra.failure.failure_estimation_provinces module

Failure analysis of province-scale road networks To estimate changing accessibility to commune points

Input data requirements

1. Correct paths to all files and correct input parameters
2. Excel sheets with results of flow mapping based on MIN-MAX generalised costs estimates
    - origin - String node ID of Origin
    - destination - String node ID of Destination
    - min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows

- max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
- min_distance - Float values of estimated distance for paths with minimum generalised cost flows
- max_distance - Float values of estimated distance for paths with maximum generalised cost flows
- min_time - Float values of estimated time for paths with minimum generalised cost flows
- max_time - Float values of estimated time for paths with maximum generalised cost flows
- min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
- max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows
- min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
- max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
- industry_columns - All daily tonnages of industry columns given in the OD matrix data

3. Shapefiles
   - edge_id - String/Integer/Float Edge ID
   - geometry - Shapely LineString geomtry of edges

Results

Csv sheets with results of failure analysis

1. All failure scenarios
   - edge_id - String name or list of failed edges
   - origin - String node ID of Origin of disrupted OD flow
   - destination - String node ID of Destination of disrupted OD flow
   - o_region - String name of Province of Origin node ID of disrupted OD flow
   - d_region - String name of Province of Destination node ID of disrupted OD flow
   - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
   - min/max_distance - Float value of estimated distance of OD journey before disruption
   - min/max_time - Float value of estimated time of OD journey before disruption
   - min/max_gcost - Float value of estimated travel cost of OD journey before disruption
   - min/max_vehicle_nums - Float value of estimated vehicles of OD journey before disruption
   - new_cost - Float value of estimated cost of OD journey after disruption
   - new_distance - Float value of estimated distance of OD journey after disruption
   - new_path - List of string edge ID's of estimated new route of OD journey after disruption
   - new_time - Float value of estimated time of OD journey after disruption
   - dist_diff - Float value of Post disruption minus per-disruption distance
   - time_diff - Float value Post disruption minus per-disruption timee
   - min/max_tr_loss - Float value of estimated change in rerouting cost
   - min/max_netrev - Float values of total daily net revenues along disrupted OD pairs
   - min/max_tons - Float values of total daily crop tonnages along disrupted OD pairs
   - min_max_econ_impact - Float values of total daily economic impact of disrupted OD pairs
2. Min-max combined scenarios - Combined min-max results of total network impacts of each edge

- edge_id - String name or list of failed edges
- no_access - Boolean 1 (no reroutng) or 0 (rerouting)
- min/max_tr_loss - Float values of estimated change in rerouting cost
- min/max_tons - Float values of total daily tonnages along edge
- min/max_netrev - Float values of total daily net revenues along edge
- min/max_econ_impact - Float value of total daily economic impact of edge

3. Shapefiles - Combined min-max reults of total network impacts of each edge
   - edge_id - String name or list of failed edges
   - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
   - min/max_tr_loss - Float values of estimated change in rerouting cost
   - min/max_tons - Float values of total daily tonnages along edge
   - min/max_netrev - Float values of total daily net revenues along edge
   - min/max_econ_impact - Float value of total daily economic impact of edge
   - geometry - Shapely LineString geomtry of edges

main()

Estimate provincial road failures

Specify the paths from where you want to read and write:

1. Input data
2. Intermediate calcuations data
3. Output results

Supply input data and parameters

1. **Names of Provinces** List of strings
2. **Unit weight of vehicle assumed for each mode** List of float types
3. **Min-max names of names of different types of attributes - paths, distance, time, cost, vehicles, tons, revenue** List of string types
4. **Percentage of OD flows that are assumed disrupted** List of float type
5. **Condition on whether analysis is single failure or multiple failure** Boolean condition True or False

Give the paths to the input data files: 1. Network edges Excel and shapefiles 2. OD flows Excel file 3. Failure scenarios Excel file

Specify the output files and paths to be created

vtra.failure.failure_multi_modal_options module

Failure analysis of national-scale networks For transport modes at national scale:

- road
- rail

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Excel sheets with results of flow mapping based on MIN-MAX generalised costs estimates:**
   - origin - String node ID of Origin

- destination - String node ID of Destination
- o_region - String name of Province of Origin node ID
- d_region - String name of Province of Destination node ID
- min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows
- max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
- min_distance - Float values of estimated distance for paths with minimum generalised cost flows
- max_distance - Float values of estimated distance for paths with maximum generalised cost flows
- min_time - Float values of estimated time for paths with minimum generalised cost flows
- max_time - Float values of estimated time for paths with maximum generalised cost flows
- min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
- max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows
- min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
- max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
- industry_columns - All daily tonnages of industry columns given in the OD matrix data

3. **Shapefiles**
    - edge_id - String/Integer/Float Edge ID
    - geometry - Shapely LineString geomtry of edges

Results

Csv sheets with results of failure analysis:

1. **All failure scenarios**
    - edge_id - String name or list of failed edges
    - origin - String node ID of Origin of disrupted OD flow
    - destination - String node ID of Destination of disrupted OD flow
    - o_region - String name of Province of Origin node ID of disrupted OD flow
    - d_region - String name of Province of Destination node ID of disrupted OD flow
    - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
    - min/max_distance - Float value of estimated distance of OD journey before disruption
    - min/max_time - Float value of estimated time of OD journey before disruption
    - min/max_gcost - Float value of estimated travel cost of OD journey before disruption
    - min/max_vehicle_nums - Float value of estimated vehicles of OD journey before disruption
    - new_cost - Float value of estimated cost of OD journey after disruption
    - new_distance - Float value of estimated distance of OD journey after disruption
    - new_path - List of string edge ID's of estimated new route of OD journey after disruption
    - new_time - Float value of estimated time of OD journey after disruption
    - dist_diff - Float value of Post disruption minus per-disruption distance
    - time_diff - Float value Post disruption minus per-disruption timee

- min/max_tr_loss - Float value of estimated change in rerouting cost
- industry_columns - Float values of all daily tonnages of industry columns along disrupted OD pairs
- min/max_tons - Float values of total daily tonnages along disrupted OD pairs

2. **Isolated OD scenarios - OD flows with no rerouting options**
   - edge_id - String name or list of failed edges
   - o_region - String name of Province of Origin node ID of disrupted OD flow
   - d_region - String name of Province of Destination node ID of disrupted OD flow
   - industry_columns - Float values of all daily tonnages of industry columns along disrupted OD pairs
   - min/max_tons - Float values of total daily tonnages along disrupted OD pairs
3. **Rerouting scenarios - OD flows with rerouting options**
   - edge_id - String name or list of failed edges
   - o_region - String name of Province of Origin node ID of disrupted OD flow
   - d_region - String name of Province of Destination node ID of disrupted OD flow
   - min/max_tr_loss - Float value of change in rerouting cost
   - min/max_tons - Float values of total daily tonnages along disrupted OD pairs
4. **Min-max combined scenarios - Combined min-max results along each edge**
   - edge_id - String name or list of failed edges
   - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
   - min/max_tr_loss - Float values of change in rerouting cost
   - min/max_tons - Float values of total daily tonnages affted by disrupted edge
5. **Shapefile Min-max combined scenarios - Combined min-max reults along each edge**
   - edge_id - String name or list of failed edges
   - no_access - Boolean 1 (no reroutng) or 0 (rerouting)
   - min/max_tr_loss - Float values of change in rerouting cost
   - min/max_tons - Float values of total daily tonnages affted by disrupted edge
   - geometry - Shapely LineString geomtry of edges

main()

Estimate multi-model failures

Specify the paths from where you want to read and write:

1. Input data
2. Intermediate calcuations data
3. Output results

Supply input data and parameters

1. **Names of modes** List of strings
2. **Unit weight of vehicle assumed for each mode** List of float types
3. **Range of usage factors for each mode to represent uncertainty in cost estimations** List of tuples of float types
4. **Min-max names of names of different types of attributes - paths, distance, time, cost, vehicles, tons** List of string types

5. **Names of commodity/industry columns for which min-max tonnage column names already exist** List of string types
6. **Percentage of OD flows that are assumed disrupted** List of float type
7. **Condition on whether analysis is single failure or multiple failure** Boolean condition True or False

Give the paths to the input data files:

1. Network edges Excel and shapefiles
2. OD flows Excel file
3. Costs of modes Excel file
4. Road properties Excel file
5. Failure scenarios Excel file

vtra.failure.national_failure_transfers module

Estimate tonnage shifts from one mode onto others

main()

Estimate modal shifts

Specify the paths from where you want to read and write:

1. Input data
2. Intermediate calcuations data
3. Output results

Supply input data and parameters

1. **Names of modes** List of strings
2. **Unit weight of vehicle assumed for each mode** List of float types
3. **Range of usage factors for each mode to represent uncertainty in cost estimations** List of tuples of float types
4. **Min-max names of names of different types of attributes - paths, distance, time, cost, vehicles, tons** List of string types
5. **Names of commodity/industry columns for which min-max tonnage column names already exist** List of string types
6. **Percentage of OD flows that are assumed disrupted** List of float type
7. **Condition on whether analysis is single failure or multiple failure** Boolean condition True or False

Give the paths to the input data files:

1. Network edges Excel and shapefiles
2. OD flows Excel file
3. Costs of modes Excel file
4. Road properties Excel file
5. Failure scenarios Excel file

vtra.failure.transfer_costs_modes module

Combine economic impacts of partial multi-modal rerouting split

main()

Combine economic impacts of partial multi-modal rerouting split

vtra.failure_scenario_selection package

Submodules

vtra.failure_scenario_selection.collect_network_hazard_scenarios_national module

Collect network hazard scenarios

main()

Process results

1. **Specify the paths from where you to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results
2. **Supply input data and parameters**
   - Names of the three Provinces - List of string types
   - Names of modes - List of strings
   - Names of output modes - List of strings
   - Names of hazard bands - List of integers
   - Names of hazard thresholds - List of integers
   - Condition 'Yes' or 'No' is the users wants to process results
3. **Give the paths to the input data files:**
   - Commune boundary and stats data shapefile
   - Hazard datasets description Excel file
   - String name of sheet in hazard datasets description Excel file

vtra.failure_scenario_selection.collect_network_hazard_scenarios_provincial module

Collect network hazard scenarios

main()

Process results

1. **Specify the paths from where you to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results
2. **Supply input data and parameters**
   - Names of the three Provinces - List of string types
   - Names of modes - List of strings
   - Names of output modes - List of strings
   - Names of hazard bands - List of integers

- Names of hazard thresholds - List of integers
- Condition 'Yes' or 'No' is the users wants to process results

3. **Give the paths to the input data files:**
   - Commune boundary and stats data shapefile
   - Hazard datasets description Excel file
   - String name of sheet in hazard datasets description Excel file

vtra.failure_scenario_selection.hazard_network_scenarios module

Utility functions for combined impact analysis

combine_hazards_and_network_attributes_and_impacts(*hazard_dataframe*, *network_dataframe*)

create_hazard_scenarios_for_adaptation(*all_edge_fail_scenarios*, *index_cols*, *length_thr*)

vtra.failure_scenario_selection.hazards_network_intersections_results_collect module

Summarise network-hazard intersections

Purpose

Collect network-hazard intersection attributes

- Combine with boundary Polygons to collect network-hazard-boundary intersection attributes
- Write final results to an Excel sheet

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Shapefiles of network-hazard intersections results with attributes:**
   - edge_id or node_id - String/Integer/Float Edge ID or Node ID of network
   - length - Float length of edge intersecting with hazards
   - geometry - Shapely geometry of edges as LineString or nodes as Points
3. **Shapefile of administrative boundaries of Vietnam with attributes:**
   - province_i - String/Integer ID of Province
   - pro_name_e - String name of Province in English
   - district_i - String/Integer ID of District
   - dis_name_e - String name of District in English
   - commune_id - String/Integer ID of Commune
   - name_eng - String name of Commune in English
   - geometry - Shapely geometry of boundary Polygon
4. **Excel sheet of hazard attributes with attributes:**
   - hazard_type - String name of hazard type
   - model - String name of hazard model
   - year - String name of hazard year
   - climate_scenario - String name of hazard scenario

- probability - Float/String value of hazard probability
- band_num - Integer value of hazard band
- min_val - Integer value of minimum value of hazard threshold
- max_val - Integer value of maximum value of hazard threshold

Results

1. **Excel sheet of network-hazard-boundary intersection with attributes:**
    - edge_id/node_id - String name of intersecting edge ID or node ID
    - length - Float length of intersection of edge LineString and hazard Polygon: Only for edges
    - province_id - String/Integer ID of Province
    - province_name - String name of Province in English
    - district_id - String/Integer ID of District
    - district_name - String name of District in English
    - commune_id - String/Integer ID of Commune
    - commune_name - String name of Commune in English
    - sector - String name of transport mode
    - hazard_type - String name of hazard type
    - model - String name of hazard model
    - year - String name of hazard year
    - climate_scenario - String name of hazard scenario
    - probability - Float/String value of hazard probability
    - band_num - Integer value of hazard band
    - min_val - Integer value of minimum value of hazard threshold
    - max_val - Integer value of maximum value of hazard threshold

create_hazard_attributes_for_network(*intersection_dir*, *sector*, *hazard_files*, *hazard_df*, *bands*, *thresholds*, *commune_shape*, *network_type=''*, *name_province=''*)

Extract results of network edges/nodes and hazard intersections to collect network-hazard intersection attributes

- Combine with boundary Polygons to collect network-hazard-boundary intersection attributes
- Write final results to an Excel sheet

Parameters

- **intersection_dir** (*str*[41]) – Path to Directory where the network-hazard shapefile results are stored
- **sector** (*str*[42]) – name of transport mode
- **hazard_files** (*list*[43] *[str*[44]*]*) – names of all hazard files
- **hazard_df** (*pandas.DataFrame*[45]) – hazard attributes

[41] https://docs.python.org/3.6/library/stdtypes.html#str
[42] https://docs.python.org/3.6/library/stdtypes.html#str
[43] https://docs.python.org/3.6/library/stdtypes.html#list
[44] https://docs.python.org/3.6/library/stdtypes.html#str
[45] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame

- **bands** (*list*[46] *[int*[47]*]*) – integer values of hazard bands
- **thresholds** (*list*[48] *[int*[49]*]*) – integer values of hazard thresholds
- **commune_shape** – Shapefile of commune boundaries and attributes
- **network_type** (*str*[50]*, optional*) – value -'edges' or 'nodes': Default = 'nodes'
- **name_province** (*str*[51]*, optional*) – name of province if needed: Default = ''

Returns

data_df –

network-hazard-boundary intersection attributes:

- edge_id/node_id - String name of intersecting edge ID or node ID
- length - Float length of intersection of edge LineString and hazard Polygon: Only for edges
- province_id - String/Integer ID of Province
- province_name - String name of Province in English
- district_id - String/Integer ID of District
- district_name - String name of District in English
- commune_id - String/Integer ID of Commune
- commune_name - String name of Commune in English
- sector - String name of transport mode
- hazard_type - String name of hazard type
- model - String name of hazard model
- year - String name of hazard year
- climate_scenario - String name of hazard scenario
- probability - Float/String value of hazard probability
- band_num - Integer value of hazard band
- min_val - Integer value of minimum value of hazard threshold
- max_val - Integer value of maximum value of hazard threshold
- length - Float length of intersection of edge LineString and hazard Polygon: Only for edges

Return type pandas.DataFrame[52]

main()

Collect results

1. **Specify the paths from where you to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results

[46] https://docs.python.org/3.6/library/stdtypes.html#list
[47] https://docs.python.org/3.6/library/functions.html#int
[48] https://docs.python.org/3.6/library/stdtypes.html#list
[49] https://docs.python.org/3.6/library/functions.html#int
[50] https://docs.python.org/3.6/library/stdtypes.html#str
[51] https://docs.python.org/3.6/library/stdtypes.html#str
[52] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame

2. **Supply input data and parameters**
    - Names of the three Provinces - List of string types
    - Names of modes - List of strings
    - Names of output modes - List of strings
    - Names of hazard bands - List of integers
    - Names of hazard thresholds - List of integers
    - Condition 'Yes' or 'No' is the users wants to process results
3. **Give the paths to the input data files:**
    - Commune boundary and stats data shapefile
    - Hazard datasets description Excel file
    - String name of sheet in hazard datasets description Excel file

vtra.failure_scenario_selection.hazards_networks_intersections module

Intersect networks with hazards

Purpose

Intersect hazards and network line and point geometries with hazatd polygons

Write final results to Shapefiles

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Shapefiles of network edges or nodes with attributes:**
    - edge_id or node_id - String/Integer/Float Edge ID or Node ID of network
    - geometry - Shapely geometry of edges as LineStrings or nodes as Points
3. **Shapefile of hazards with attributes:**
    - geometry - Shapely geometry of hazard Polygon

Results

1. **Edge shapefiles with attributes:**
    - edge_id - String name of intersecting edge ID
    - length - Float length of intersection of edge LineString and hazard Polygon
    - geometry - Shapely LineString geometry of intersection of edge LineString and hazard Polygon
2. **Node Shapefile with attributes:**
    - node_id - String name of intersecting node ID
    - geometry - Shapely Point geometry of intersecting node ID

intersect_networks_and_all_hazards(*hazard_dir*, *network_file_path*, *network_file_name*, *output_file_path*, *network_type=''*)

Walk through all hazard files and select network-hazard intersection criteria

Parameters

- **hazard_dir** (*str*[53]) – name of directory where all hazard shapefiles are stored

[53] https://docs.python.org/3.6/library/stdtypes.html#str

- **network_file_path** (*str*[54]) – name of directory where network shapefile is stored
- **network_file_name** (*str*[55]) – name network shapefile
- **output_file_path** (*str*[56]) – name of directory where network-hazard instersection result shapefiles will be stored
- **network_type** (*str*[57]) – values of 'edges' or 'nodes'

Edge or Node shapefiles

main()

Intersect networks with hazards

1. **Specify the paths from where you to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results
2. **Supply input data and parameters**
   - Names of the three Provinces - List of string types
   - Paths of the mode files - List of tuples of strings
   - Names of modes - List of strings
   - Names of output modes - List of strings
   - Condition 'Yes' or 'No' is the users wants to process results
3. **Give the paths to the input data files:**
   - Hazard directory

networkedge_hazard_intersection(*edge_shapefile*, *hazard_shapefile*, *output_shapefile*)

Intersect network edges and hazards and write results to shapefiles

Parameters

- **edge_shapefile** – Shapefile of network LineStrings
- **hazard_shapefile** – Shapefile of hazard Polygons
- **output_shapefile** – String name of edge-hazard shapefile for storing results

output_shapefile

- edge_id - String name of intersecting edge ID
- length - Float length of intersection of edge LineString and hazard Polygon
- geometry - Shapely LineString geometry of intersection of edge LineString and hazard Polygon

networknode_hazard_intersection(*node_shapefile*, *hazard_shapefile*, *output_shapefile*)

Intersect network nodes and hazards and write results to shapefiles

Parameters

- **node_shapefile** – Shapefile of network Points
- **hazard_shapefile** – Shapefile of hazard Polygons
- **output_shapefile** – String name of node-hazard shapefile for storing results

[54] https://docs.python.org/3.6/library/stdtypes.html#str
[55] https://docs.python.org/3.6/library/stdtypes.html#str
[56] https://docs.python.org/3.6/library/stdtypes.html#str
[57] https://docs.python.org/3.6/library/stdtypes.html#str

output_shapefile

- node_id - String name of intersecting node ID
- geometry - Shapely Point geometry of intersecting node ID

vtra.flow_mapping package

Flow mapping

Submodules

vtra.flow_mapping.national_modes_flow_paths module

Map flows on national networks

Purpose

Mapping the OD node level matrix values to network paths

For all transport modes at national scale: ['road', 'rail', 'air', 'inland', 'coastal']

The code estimates 2 values - A MIN and a MAX value of flows between each selected OD node pair

- Based on MIN-MAX generalised costs estimates

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Excel file with mode sheets containing network graph structure and attributes**
   - edge_id - String Edge ID
   - from_node - String node ID that should be present in node_id column
   - to_node - String node ID that should be present in node_id column
   - length - Float length of edge in km
   - min_time - Float minimum time of travel in hours on edge
   - max_time - Float maximum time of travel in hours on edge
   - min_time_cost - Float minimum cost of time in USD on edge
   - max_time_cost - Float maximum cost of time in USD on edge
   - min_tariff_cost - Float minimum tariff cost in USD on edge
   - max_tariff_cost - Float maximum tariff cost in USD on edge
3. **Edge shapefiles for all national-scale networks with attributes:**
   - edge_id - String Edge ID
   - geometry - Shapely LineString geometry of edges
4. **Excel file with mode sheets containing node-level OD values with attributes:**
   - origin - String node ID of Origin
   - destination - String node ID of Destination
   - min_tons - Float values of minimum daily OD in tons
   - max_tons - Float values of maximum daily OD in tons
   - Names of the industry columns specified in the inputs

Results

1. **Excel sheets with results of flow mapping based on MIN-MAX generalised costs estimates:**
    - origin - String node ID of Origin
    - destination - String node ID of Destination
    - o_region - String name of Province of Origin node ID
    - d_region - String name of Province of Destination node ID
    - min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows
    - max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
    - min_distance - Float values of estimated distance for paths with minimum generalised cost flows
    - max_distance - Float values of estimated distance for paths with maximum generalised cost flows
    - min_time - Float values of estimated time for paths with minimum generalised cost flows
    - max_time - Float values of estimated time for paths with maximum generalised cost flows
    - min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
    - max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows
    - min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
    - max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
    - industry_columns - All daily tonnages of industry columns given in the OD matrix data
2. **Shapefiles**
    - edge_id - String/Integer/Float Edge ID
    - geometry - Shapely LineString geomtry of edges
    - min_{industry} - Float values of estimated minimum daily industries/commodities/total volumes in tons on edges
    - max_{industry} - Float values of estimated maximum daily industries/commodities/total volumes in tons on edges

References

1. Pant, R., Koks, E.E., Russell, T., Schoenmakers, R. & Hall, J.W. (2018). Analysis and development of model for addressing climate change/disaster risks in multi-modal transport networks in Vietnam. Final Report, Oxford Infrastructure Analytics Ltd., Oxford, UK.

2. All input data folders and files referred to in the code below.

main()

Estimate flows

1. **Specify the paths from where you want to read and write:**
    - Input data
    - Intermediate calcuations data
    - Output results
2. **Supply input data and parameters**
    - Names of modes: List of strings

- Unit weight of vehicle assumed for each mode: List of float types
- Names of all industry sector and crops in VITRANSS2 and IFPRI datasets: List of string types
- Names of commodity/industry columns for which min-max tonnage column names already exist: List of string types
- Percentage of OD flow we want to send along path: FLoat type

3. **Give the paths to the input data files:**
   - Network edges Excel file
   - OD flows Excel file
   - Costs of modes Excel file
   - Road properties Excel file
4. Specify the output files and paths to be created

network_od_paths_assembly_national (*points_dataframe*, *graph*, *vehicle_wt*, *transport_mode*, *excel_writer=''*)

Assemble estimates of OD paths, distances, times, costs and tonnages on networks

Parameters

- **points_dataframe** (*pandas.DataFrame*[58]) – OD nodes and their tonnages
- **graph** – igraph network structure
- **vehicle_wt** (*float*[59]) – unit weight of vehicle
- **transport_mode** (*str*[60]) – name of modes
- **excel_writer** – Name of the excel writer to save Pandas dataframe to Excel file

Returns

save_paths_df –

- origin - String node ID of Origin
- destination - String node ID of Destination
- min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows
- max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
- min_distance - Float values of estimated distance for paths with minimum generalised cost flows
- max_distance - Float values of estimated distance for paths with maximum generalised cost flows
- min_time - Float values of estimated time for paths with minimum generalised cost flows
- max_time - Float values of estimated time for paths with maximum generalised cost flows
- min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
- max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows

[58] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame
[59] https://docs.python.org/3.6/library/functions.html#float
[60] https://docs.python.org/3.6/library/stdtypes.html#str

- min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
- max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
- industry_columns - All tonnages of industry columns given in the OD matrix data

Return type pandas.DataFrame[61]

vtra.flow_mapping.province_roads_access_flow_paths module

Map flows on provincial networks

Purpose

Mapping the commune access OD node level matrix values to road network paths in Provinces

For all roads in the Provinces: ['Lao Cai', 'Binh Dinh', 'Thanh Hoa']

The code estimates 2 values - A MIN and a MAX value of flows between each selected OD node pair

- Based on MIN-MAX generalised costs estimates

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Excel file with mode sheets containing network graph structure and attributes**
   - edge_id - String Edge ID
   - from_node - String node ID that should be present in node_id column
   - to_node - String node ID that should be present in node_id column
   - length - Float length of edge in km
   - min_time - Float minimum time of travel in hours on edge
   - max_time - Float maximum time of travel in hours on edge
   - min_time_cost - Float minimum cost of time in USD on edge
   - max_time_cost - Float maximum cost of time in USD on edge
   - min_tariff_cost - Float minimum tariff cost in USD on edge
   - max_tariff_cost - Float maximum tariff cost in USD on edge
3. **Edge shapefiles for all national-scale networks with attributes:**
   - edge_id - String Edge ID
   - geometry - Shapely LineString geometry of edges
4. **Excel file with mode sheets containing node-level OD values with attributes:**
   - origin - String node ID of Origin
   - destination - String node ID of Destination
   - min_netrev - Float values of miniimum daily OD Net Revenue in USD
   - max_netrev - Float values of maximum daily OD Net Revenue in USD
   - min_tons - Float values of minimum daily OD in tons
   - max_tons - Float values of maximum daily OD in tons

[61] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame

Results

1. **Excel sheets with results of flow mapping based on MIN-MAX generalised costs estimates:**
    - origin - String node ID of Origin
    - destination - String node ID of Destination
    - min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows
    - max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
    - min_netrev - Float values of estimated daily Net Revenue for paths with minimum generalised cost flows
    - max_netrev - Float values of estimated daily Net Revenue for paths with maximum generalised cost flows
    - min_croptons - Float values of estimated daily crop tonnage for paths with minimum generalised cost flows
    - max_croptons - Float values of estimated daily crop tonnage for paths with maximum generalised cost flows
    - min_distance - Float values of estimated distance for paths with minimum generalised cost flows
    - max_distance - Float values of estimated distance for paths with maximum generalised cost flows
    - min_time - Float values of estimated time for paths with minimum generalised cost flows
    - max_time - Float values of estimated time for paths with maximum generalised cost flows
    - min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
    - max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows
    - min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
    - max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows
2. **Shapefiles with all flows on edges mapping based on MIN-MAX generalised costs estimates:**
    - edge_id - String/Integer/Float Edge ID
    - geometry - Shapely LineString geomtry of edges
    - min_netrev - Float values of estimated daily Net Revenue in USD on edges
    - max_netrev - Float values of estimated daily Net Revenue in USD on edges
    - min_tons - Float values of estimated daily crops in tons on edges
    - max_tons - Float values of estimated daily crops in tons on edges

References

1. Pant, R., Koks, E.E., Russell, T., Schoenmakers, R. & Hall, J.W. (2018). Analysis and development of model for addressing climate change/disaster risks in multi-modal transport networks in Vietnam. Final Report, Oxford Infrastructure Analytics Ltd., Oxford, UK.

2. All input data folders and files referred to in the code below.

main()

Map flows to networks

1. **Specify the paths from where you want to read and write:**
    - Input data

- Intermediate calcuations data
- Output results

2. **Supply input data and parameters**
   - Names of the three Provinces: List of string types
   - Assumed terrains of the provinces: List of string types
   - Assumed unit weights of trucks: List of float types
3. **Give the paths to the input data files:**
   - Network edges EXcel File
   - OD flows Excel file
   - Road properties Excel file
4. Specify the output files and paths to be created

network_od_paths_assembly_provincial(*points_dataframe*, *graph*, *vehicle_wt*, *region_name*, *excel_writer=''*)

Assemble estimates of OD paths, distances, times, costs and tonnages on networks

Parameters

- **points_dataframe** (*pandas.DataFrame*[62]) – OD nodes and their tonnages
- **graph** – igraph network structure
- **vehicle_wt** (*float*[63]) – unit weight of vehicle
- **region_name** (*str*[64]) – name of Province
- **excel_writer** – Name of the excel writer to save Pandas dataframe to Excel file

Returns

save_paths_df –

- origin - String node ID of Origin
- destination - String node ID of Destination
- min_edge_path - List of string of edge ID's for paths with minimum generalised cost flows
- max_edge_path - List of string of edge ID's for paths with maximum generalised cost flows
- min_netrev - Float values of estimated netrevenue for paths with minimum generalised cost flows
- max_netrev - Float values of estimated netrevenue for paths with maximum generalised cost flows
- min_croptons - Float values of estimated crop tons for paths with minimum generalised cost flows
- max_croptons - Float values of estimated crop tons for paths with maximum generalised cost flows
- min_distance - Float values of estimated distance for paths with minimum generalised cost flows
- max_distance - Float values of estimated distance for paths with maximum generalised cost flows

[62] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame
[63] https://docs.python.org/3.6/library/functions.html#float
[64] https://docs.python.org/3.6/library/stdtypes.html#str

- min_time - Float values of estimated time for paths with minimum generalised cost flows
- max_time - Float values of estimated time for paths with maximum generalised cost flows
- min_gcost - Float values of estimated generalised cost for paths with minimum generalised cost flows
- max_gcost - Float values of estimated generalised cost for paths with maximum generalised cost flows
- min_vehicle_nums - Float values of estimated vehicle numbers for paths with minimum generalised cost flows
- max_vehicle_nums - Float values of estimated vehicle numbers for paths with maximum generalised cost flows

Return type pandas.DataFrame[65]

vtra.mria package

Submodules

vtra.mria.disruption module

Create disruption files to be used in the MRIA loss estimations.

create_disruption(*input_file*, *output_dir*, *min_rice=True*, *single_point=True*)

Create disruption file for the economic analysis.

The input for this disruption file is the outcome of the flow analysis. This function translate the failure in transport flows into impacts to the economic sectors.

Parameters

- **input_file** (*str*[66]) – name of the path to a flow failure scenario file
- **output_dir** (*str*[67]) – name for output directory for mapper file.
- **min_rice** (*bool*[68], *optional*) – determine whether you want to use the minimal rice value or the maximum rice value from the flow analysis. This MUST match the value used when creating the MRIO table. The default is **True**.
- **single_point** (*bool*[69], *optional*) – determine whether you are converting a single-point or multi-point failure analaysis to a disruption file. The default is **True**.

Returns Dictionary of all unique failure events, in terms of percentage disruption per sector in each directly affected region due to the flow failure.

Return type event_dict

df_com_to_ind(*comm_des*, *min_rice=True*)

Convert the national Origin-Destination matrix from goods to sectors.

Parameters

- **comm_dess** – national Origin-Destination matrix, showing origin and destination of goods.
- **min_rice** (*bool*[70]) – determine whether to use the minimal rice value or the maximum rice value from the flow analysis. This MUST match the value used when creating the MRIO table. The default is **True**.

[65] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame
[66] https://docs.python.org/3.6/library/stdtypes.html#str
[67] https://docs.python.org/3.6/library/stdtypes.html#str
[68] https://docs.python.org/3.6/library/functions.html#bool
[69] https://docs.python.org/3.6/library/functions.html#bool
[70] https://docs.python.org/3.6/library/functions.html#bool

Returns a Pandas dataframe of the national OD matrix on a sector level.

Return type df_od_ind

map_comm_ind(*x*)

Map the goods from the flow failure analysis to economic sectors.

Parameters **x** (*str*[71]) – row in the disruption dataframe.

Returns **x** – mapped good to sector for the specific row in the disruption dataframe

Return type str[72]

map_ind(*x*)

Map the abbreviated names for the industries to their full name.

Parameters **x** (*str*[73]) – row in the disruption dataframe.

Returns **x** – mapped abbrevation to full name for the specific row in the disruption dataframe.

Return type str[74]

vtra.mria.model module

MRIA Model

Purpose

The Multiregional Impact Assessment (MRIA) Model allows for estimating a new post-disaster economic situation in equilibrium, given a set of disruptions.

References

1) Koks, E. E., & Thissen, M. (2016). A multiregional impact assessment model for disaster analysis. Economic Systems Research, 28(4), 429-449.

class MRIA_IO(*name*, *list_regions*, *list_sectors*, *list_fd_cats=[]*)

Bases: *object*[75]

MRIA_IO sets up the modelling framework.

In this class we define the type of model, sets, set up the core variables and specify the constraints and objectives for different model setups.

name

name of the model

Type string

m

model instance

Type Pyomo.ConcreteModel

regions

model regions

Type list[76]

total_regions

number of regions

[71] https://docs.python.org/3.6/library/stdtypes.html#str
[72] https://docs.python.org/3.6/library/stdtypes.html#str
[73] https://docs.python.org/3.6/library/stdtypes.html#str
[74] https://docs.python.org/3.6/library/stdtypes.html#str
[75] https://docs.python.org/3.6/library/functions.html#object
[76] https://docs.python.org/3.6/library/stdtypes.html#list

Type int[77]

sectors
model sectors

Type list[78]

fd_cat
final demand categories

Type list[79]

baseline_data (*Table*, *disr_dict_sup*, *disr_dict_fd*)
Set up all the baseline variables for the MRIA model

Parameters

- **Table** – the **io_basic** class object
- **disr_dict_sup** – dictionary containing the reduction in production capacity
- **disr_dict_fd** – dictionary containing the disruptions in final demand
- **all required parameters and variables for the MRIA_IO class and the MRIA** (*Set*) –
- **model.** –

create_A_mat (*A_mat_in*)
Creation of the A-matrix for the optimization model.

Parameters **A_mat_in** – A-matrix dictionary from the **io_basic** class object

create_DisImp (*disr_dict*, *Regmaxcap=0.98*)
Create disaster import variable.

Parameters

- **disr_dict** – dictionary containing the reduction in production capacity
- **Regmaxcap** – maximum regional capacity. The default value is set to **0.98**
- **self.DisImp - Pyomo Variable instance for disaster imports** (*Set*) –

create_ExpImp (*ExpROW*, *ImpROW*)
Specify export and import to the rest of the world

Parameters

- **ExpROW** – Exports to the Rest of the World dictionary from the **io_basic** class object
- **ImpROW** – Imports from the Rest of the World dictionary from the **io_basic** class object
- **self.ExpROW - Pyomo Parameter instance for the Exports to the Rest of the World** (*Create*) –
- **self.ImpROW - Pyomo Parameter instance for the Imports from the Rest of the World** (*and*) –

create_FD (*FinalD*, *disr_dict_fd*)
Specify Final Demand and Local Final Demand.

Parameters

- **FinalD** – Final Demand dictionary from the **io_basic** class object
- **disr_dict_fd** – dictionary containing the disruptions in final demand

[77] https://docs.python.org/3.6/library/functions.html#int
[78] https://docs.python.org/3.6/library/stdtypes.html#list
[79] https://docs.python.org/3.6/library/stdtypes.html#list

- **self.ttfd – Pyomo Parameter instance for the total final demand** (*Create*) –
- **self.fd – Pyomo Parameter instance for the final demand** (*and*) –

create_ImpShares()
Estimate Import shares and Import share DisImp

Create *self*.ImportShare - Pyomo Parameter instance for the total import shares

create_LFD(*FinalD*)
Specify local final demand

Parameters

- **FinalD** – Final Demand dictionary from the **io_basic** class object
- **self.lfd – Pyomo Parameter instance for the local final demand** (*Create*) –

create_Rat(*FinalD=None*, *Z_matrix=None*)
Create rationing variable

Parameters

- **FinalD** – Final Demand dictionary from the **io_basic** class object
- **Z_matrix** – Z-matrix dictionary from the **io_basic** class object
- **self.Rat – Pyomo Variable instance for rationing** (*Set*) –

create_Ratmarg(*Table*)
Estimate the marginal values of the rationing variable

Parameters

- **Table** – the **io_basic** class object
- **self.Ratmarg – Pyomo Parameter instance for the marginal value of rationing** (*Set*) –

create_Rdem()
Create reconstruction demand variable.

Create *self*.Rdem - Pyomo Parameter instance for the total reconstruction demand

create_TotExp()
Estimate Total Export

Create *self*.TotExp - Pyomo Parameter instance for the total export

create_TotImp()
Estimate Total Import

Create *self*.TotExp - Pyomo Parameter instance for the total import

create_Trade(*FinalD*, *Z_matrix=None*)
Create Trade Matrix

Parameters

- **FinalD** – Final Demand dictionary from the **io_basic** class object
- **Z_matrix** – Z-matrix dictionary from the **io_basic** class object
- **self.trade – Pyomo Parameter instance for the trade matrix between regions** (*Create*) –

create_VA(*ValueA*)
Specify Value Added

Parameters

- **ValueA** – Value Added dictionary from the **io_basic** class object
- **self.ValueA - Pyomo Parameter instance for the total Value Added** (*Create*) –

create_X(*disr_dict*, *Regmaxcap=0.98*, *A_matrix_ini=None*, *Z_matrix=None*, *FinalD=None*, *Xbase=None*, *fd=None*, *ExpROW=None*)

Create the total production **X** variable

Parameters

- **disr_dict** – dictionary containing the reduction in production capacity
- **Regmaxcap** – maximum regional capacity. The default value is set to **0.98**
- **A_matrix_ini** – A-matrix dictionary from the **io_basic** class object
- **Z_matrix** – Z-matrix dictionary from the **io_basic** class object
- **FinalD** – Final Demand dictionary from the **io_basic** class object
- **Xbase** – Total Production **X** parameter from the **MRIA** class object
- **fd** – Final Demand parameter from the **MRIA** class object
- **ExpROW** – Export to the Rest of the World parameter from the **MRIA** class object
- **self.X_up - Pyomo Variable instance of total production X** (*Create*) –

create_X_up(*disr_dict*, *Regmaxcap=0.98*)

Specify upper bound of total production **X**.

Parameters

- **disr_dict** (*dict*[80]) – dictionary containing the reduction in production capacity
- **Regmaxcap** (*float*[81], *optional*) – maximum regional capacity. The default value is set to **0.98**
- **self.X_up - Pyomo Parameter instance for the upper bound of total production X** (*Create*) –

create_Xbase(*Z_matrix*, *disr_dict*, *FinalD=None*)

Specify Baseline value of total production **X**

Parameters

- **Z_matrix** – Z-matrix dictionary from the **io_basic** class object
- **disr_dict** – dictionary containing the reduction in production capacity
- **FinalD** – Final Demand dictionary from the **io_basic** class object
- **self.X_up - Pyomo Parameter instance for the Baseline value of total production X** (*Create*) –

create_Z_mat()

Specify Trade between regions

Create *self*.Z_matrix - Pyomo Parameter instance for the total trade matrix

create_alias()

Set aliases.

Parameters

- **list_regions** (*list*[82] [*str*[83]]) – regions to include in the model calculations

[80] https://docs.python.org/3.6/library/stdtypes.html#dict
[81] https://docs.python.org/3.6/library/functions.html#float
[82] https://docs.python.org/3.6/library/stdtypes.html#list
[83] https://docs.python.org/3.6/library/stdtypes.html#str

- **list_sectors** (*list*[84] *[str*[85]*]*) – sectors to include in the model calculations
- **list_fd_cats** (*list*[86] *[str*[87]*]*) – the final demand categories in the table. This will be aggregated to one column.

create_demand()

Specify demand function

Create *self*.Demand - Pyomo Variable instance for total demand

create_sets(*FD_SET=None*, *VA_SET=None*)

Create of the various sets, allowing for specification of set inputs.

Parameters

- **FD_SET** (*list*[88] *or None*[89]*, optional*) – final demand categories, default to an **empty list**
- **VA_SET** (*list*[90] *or None*[91]*, optional*) – value added categories, default to an **empty list**

impact_data(*Table*, *disr_dict_sup*, *disr_dict_fd*, *Regmaxcap=0.98*)

Create additional parameters and variables required for impact analysis

Parameters

- **Table** – the **io_basic** class object
- **disr_dict_sup** – dictionary containing the reduction in production capacity
- **disr_dict_fd** – dictionary containing the disruptions in final demand
- **Regmaxcap** – maximum regional capacity. The default value is set to **0.98**
- **all additional parameters and variables required for the MRIA_IO class and the** (*Set*) –
- **model to do an impact analysis.** (***MRIA***) –

run_basemodel(*solver=None*)

Run the baseline model of the MRIA model.

This should return the baseline situation (i.e. no changes between input matrix and output matrix).

Parameters

- **solver** (*str*[92]) – Specify the solver to be used with Pyomo. The Default value is set to **None**. If set to **None**, the ipopt solver will be used
- **out the output of an optimized MRIA_IO class and the MRIA model** (*Write*) –

run_impactmodel(*solver=None*, *output=None*, *tol=1e-06*, *DisWeight=1.75*, *RatWeight=2*)

Run the **MRIA** model with disruptions. This will return an economy with a new equilibrium, based on the new production and demand values.

Parameters

- **solver** – Specify the solver to be used with Pyomo. The Default value is set to **None**. If set to **None**, the ipopt solver will be used

[84] https://docs.python.org/3.6/library/stdtypes.html#list
[85] https://docs.python.org/3.6/library/stdtypes.html#str
[86] https://docs.python.org/3.6/library/stdtypes.html#list
[87] https://docs.python.org/3.6/library/stdtypes.html#str
[88] https://docs.python.org/3.6/library/stdtypes.html#list
[89] https://docs.python.org/3.6/library/constants.html#None
[90] https://docs.python.org/3.6/library/stdtypes.html#list
[91] https://docs.python.org/3.6/library/constants.html#None
[92] https://docs.python.org/3.6/library/stdtypes.html#str

- **output** – Specify whether you want the solver to print its progress.The default value is set to **None**
- **tol** – the tolerance value that determines whether the outcome of the model is feasible. The default value is set to **1e-6**
- **DisWeight** – the weight that determines the penalty set to let the model allow for additional imports. A higher penalty value will result in less imports. The default value is set to **1.75**
- **RatWeight** – the weight that determines the penalty set to let the model allow to ration goods. A higher penalty value will result in less rationing. The default value is set to **2**
- **write out the output of an optimized MRIA_IO class and MRIA model** (*Optionally*) –

vtra.mria.ratmarg module

Create output file for gams and run a quick gams module to estimate marginal values of rationing demand

load_db_IO(*table_in*)

Load the Input-Output data from the **io_basic** Class object and converts it to a GAMS .gdx file.

Write out .gdx file of the IO data

Parameters **table_in** – **io_basic** class object, containing all IO data

ratmarg_IO(*table_in*)

Estimate marginal values of the rationing variable in GAMS. GAMS is required, as the marginal values of a variable are not returned in the free python solvers.

Parameters

- table_in - **io_basic** class object, containing all IO data

Outputs

- pandas DataFrame with the marginal values of the rationing variable

vtra.mria.run_mria module

Run the MRIA Model for a given set of disruptions.

estimate_losses(*input_file*)

Estimate the economic losses for a given set of failure scenarios

Parameters

- input_file - String name of input file to failure scenarios

Outputs

- .csv file with total losses per failure scenario

vtra.mria.table module

Create the economic tables required to run the MRIA model.

class io_basic(*name*, *filepath*, *list_regions*)

Bases: object[93]

io_basic is used to set up the table.

load_all_data()

Load all data for the **io_basic** class.

[93] https://docs.python.org/3.6/library/functions.html#object

Parameters

- *self* - **io_basic** class object

Output

- *self*.FD_data - pandas Dataframe of Final Demand in the **io_basic** class
- *self*.T_data - pandas Dataframe of Z matrix in the **io_basic** class
- *self*.VA_data - pandas Dataframe of Value Added in the **io_basic** class
- *self*.ImpROW_data - pandas Dataframe of import from the Rest of the World in the **io_basic** class
- *self*.ExpROW_data - pandas Dataframe of exports to the Rest of The World in the **io_basic** class

load_labels()

Load all labels for the **io_basic** class.

Parameters

- *self* - **io_basic** class object

Output

- *self*.FD_labels - labels for Final Demand columns in the **io_basic** class
- *self*.FD_cat - labels for Final Demand categories in the **io_basic** class
- *self*.Exp_labels - labels for Export columns in the **io_basic** class
- *self*.T_labels - region and sector labels for Z-matrix in the **io_basic** class
- *self*.VA_labels - labels for Value Added in the **io_basic** class
- *self*.sectors - labels for the sectors in the **io_basic** class

prep_data()

Transform the dataframes into dictionaries, ready to be used in the **MRIA_IO** class instance.

Parameters

- *self* - **io_basic** class object

Output

- *self*.FinalD - dictionary of Final Demand in the **io_basic** class
- *self*.A_matrix - dictionary of A matrix in the **io_basic** class
- *self*.Z_matrix - dictionary of Z matrix in the **io_basic** class
- *self*.ValueA - dictionary of Value Added in the **io_basic** class
- *self*.ImpROW - dictionary of import from the Rest of the World in the **io_basic** class
- *self*.ExpROW - dictionary of exports to the Rest of The World in the **io_basic** class

vtra.mrio package

Submodules

vtra.mrio.functions module

MRIO utility functions

aggregate_table(*vnm_IO*, *vnm_IO_rowcol*, *in_million=True*)

Aggregate national Input-Output table to the amount of sectors used in the multiregional Input-Output table.

Parameters

- vnm_IO - pandas dataframe of national Input-Output table

- vnm_IO_rowcol - pandas dataframe with all sectors in the national Input-Output table
- in_million - Specify whether we want to divide the table by 1000000, to have values in millions. The default value is set to **True**

Outputs

- pandas Dataframe with aggregated national Input-Output table

create_indices(*data_path*, *provinces*, *write_to_csv=True*)
Create list of indices required to disaggregate the national table to the different regions.

Parameters

- data_path - String name of data path
- provinces - pandas DataFrame with provincial/regional data
- write_to_csv - Boolean to specify whether you want to save output to .csv files. The default value is set to **True**

Outputs

- set of .csv files with indices proxy data

create_level14_proxies(*data_path*, *od_table*, *own_production_ratio=0.8*, *write_to_csv=True*)
Function to create the level14 proxies, required to disaggregate the national table.

Parameters

- data_path - String name of data path
- od_table - pandas DataFrame with the Origin-Destination matrix
- own_production_ratio - Specify how much supply and demand is locally supplied and used, and how much is imported/exported. The default is set to **0.8**
- write_to_csv - Boolean to specify whether you want to save output to .csv files. The default value is set to **True**

Outputs

- set of .csv files with level 14 proxy data

create_proxies(*data_path*, *notrade=False*, *own_production_ratio=0.9*, *min_rice=True*)
Create all proxies required in the disaggregation process.

Parameters

- data_path - String name of data path
- notrade - Boolean to specify whether we should include trade in the disaggregation. This should be set to **True** in the first step of the disaggregation. The default is set to **False**
- min_rice - Boolean to determine whether you want to use the minimal rice value or the maximum rice value from the flow analysis. The default is set to **True**
- own_production_ratio - Specify how much supply and demand is locally supplied and used, and how much is imported/exported. The default is set to **0.8**

Outputs

- all proxy level .csv files.

create_regional_proxy(*data_path*, *regions*, *write_to_csv=True*)
Function to create the proxy to disaggregate the national table to the different regions.

Parameters

- data_path - String name of data path
- regions - pandas DataFrame with provincial/regional data

- write_to_csv - Boolean to specify whether you want to save output to .csv files. The default value is set to **True**

Outputs

- set of .csv files with regional proxy data

create_sector_proxies (*data_path*, *regions*, *write_to_csv=True*)

Create sector proxies required to disaggregate the national table to the different sectors in each region.

Parameters

- data_path - String name of data path
- regions - pandas DataFrame with provincial/regional data
- write_to_csv - Boolean to specify whether you want to save output to .csv files. The default value is set to **True**

Outputs

- set of .csv files with sector proxy data

create_zero_proxies (*data_path*, *od_table*, *notrade=False*, *write_to_csv=True*)

Function to create the trade proxies, required to disaggregate the national table.

Parameters

- data_path - String name of data path
- od_table - pandas DataFrame with the Origin-Destination matrix
- notrade - Boolean to specify whether we should include trade in the disaggregation. This should be set to **True** in the first step of the disaggregation. The default is set to **False**
- write_to_csv - Boolean to specify whether you want to save output to .csv files. The default value is set to **True**

Outputs

- set of .csv files with level 14 proxy data

estimate_gva (*regions*, *in_million=True*)

Functions to estimate the Gross Value Added for each sector in each province.

Parameters

- regions - pandas DataFrame with provincial/regional data

Outputs

- list with GVA values per sector in each province

get_final_sector_classification ()

Return the list of sectors to be used in the new multiregional Input-Output table.

Outputs:

- list of sectors

get_trade_value (*x*, *sum_use*, *sector*, *own_production_ratio=0.8*)

Function to get the trade value between a certain origin and destination.

Parameters

- x - row in Origin-Destination dataframe
- sum_use - total use in a certain destination
- own_production_ratio - Specify how much supply and demand is locally supplied and used, and how much is imported/exported. The default is set to **0.8**

Outputs

- returns trade value

is_balanced(*io_table*)
Function to check if Input-Output table is balanced.

Parameters

- io_table - Input-Output table.

Outputs

- return print statement if table is balanced.

load_od(*data_path*, *min_rice=True*)
Load national Origin-Destination matrix as pandas DataFrame.

Parameters

- data_path - String name of data path
- min_rice - Boolean to determine whether you want to use the minimal rice value or the maximum rice value from the flow analysis. The default is set to **True**

Outputs

- pandas DataFrame with national Origin-Destination matrix

load_output(*data_path*, *provinces*, *notrade=True*)
Read output from disaggregation process and translate to usable pandas DataFrame

Parameters

- data_path - String name of data path
- provinces - pandas DataFrame with provincial/regional data
- notrade - Boolean to specify whether we should include trade in the disaggregation. This should be set to **True** in the first step of the disaggregation. The default is set to **False**

Outputs

- pandas DataFrame with disaggregated Input-Output table

load_provincial_stats(*data_path*)
Load shapefile with provincial-level data.

Parameters

- data_path - String name of data path

Outputs

- geopandas GeoDataFrame with provincial data.

load_sectors(*data_path*)
Load national Input-Output table and extracted all sectors

Parameters

- data_path - String name of data path

Outputs

- pandas Dataframe with all sectors in national Input-Output table

load_table(*data_path*)
Load national Input-Output table as pandas dataframe.

Parameters

- file_path - String name of data path

Outputs

- pandas Dataframe with Input-Output table that is going to be used

map_regions()
Create dictionary to map regions to consistent format.

Outputs

- dictionary to map regions to consistent format

map_sect_vnm_to_eng()
Convert vietnamese sector names to simple sector classification.

Outputs

- dictionary to map vietnamese sectors to simple sector names.

map_sectors(*vnm_IO_rowcol*)
Map the sectors of the loaded national Input-Output table to the sectors which are going to used in the multiregional Input-Output table.

Parameters

- vnm_IO_rowcol - pandas dataframe with all sectors in the national Input-Output table.

Outputs

- dictionary to map row sectors
- dictionary to map column sectors

map_sectors_to_od(*od_table*)
Create dictionary to map products from national Origin-Destination matrix to sector classification for the Input-Output table.

Parameters

- od_table - pandas DataFrame with the Origin-Destination matrix

Outputs

- dictionary to map goods to sectors.

vtra.mrio.ras_method module

RAS Method

Purpose

Estimate a new matrix X with exogenously given row and column totals that is a close as possible to a given original matrix X0 using the Generalized RAS (GRAS) approach

Usage

X = gras(X0, u, v) OR [X, r, s] = gras(X0, u, v) with or without eps included as the fourth argument, where

Input

- X0 = benchmark (base) matrix, not necessarily square
- u = column vector of (new) row totals
- v = column vector of (new) column totals
- eps = convergence tolerance level; if empty, the default threshold is 0.1e-5 (=0.000001)

Output

- X = estimated/adjusted/updated matrix
- r = substitution effects (row multipliers)
- s = fabrication effects (column multipliers)

References

1) Junius T. and J. Oosterhaven (2003), The solution of updating or regionalizing a matrix with both positive and negative entries, Economic Systems Research, 15, pp. 87-96.
2) Lenzen M., R. Wood and B. Gallego (2007), Some comments on the GRAS method, Economic Systems Research, 19, pp. 461-465.
3) Temurshoev, U., R.E. Miller and M.C. Bouwmeester (2013), A note on the GRAS method, Economic Systems Research, 25, pp. 361-367.

invd(*x*)

Extract projection, data bands numbers and valuees from raster

Parameters

- file_path - String name of input GeoTff file path

Outputs

- counts - Number of bans in raster
- crs - Projection system of raster
- data_vals - Numpy array of raster values

ras_method(*X0*, *u*, *v*, *eps=1e-05*)

Extract projection, data bands numbers and valuees from raster

Parameters

- file_path - String name of input GeoTff file path

Outputs

- counts - Number of bans in raster
- crs - Projection system of raster
- data_vals - Numpy array of raster values

vtra.mrio.run_mrio module

Run MRIO

main()

mrio_to_excel(*Xin*, *min_rice=True*)

Save the newly created multiregional Input-Output table to Excel, in the format required for the MRIA calculation.

Parameters

- Xin - pandas DataFrame of the new multiregional Input-Output table
- min_rice - Boolean to determine whether you want to use the minimal rice value or the maximum rice value from the flow analysis. The default is set to **True**

Outputs

- .xlsx file with the multiregional Input-Output table

run_mrio_disaggregate(*notrade=False*, *min_rice=True*, *own_production_ratio=0.8*)

This function will disaggregate the (single-region) national Input-Output table to a provincial multiregional Input-Output table

Parameters

- notrade - Boolean to specify whether we should include trade in the disaggregation. This should be set to **True** in the first step of the disaggregation. The default is set to **False**
- min_rice - Boolean to determine whether you want to use the minimal rice value or the maximum rice value from the flow analysis. The default is set to **True**
- own_production_ratio - Specify how much supply and demand is locally supplied and used, and how much is imported/exported. The default is set to **0.8**

Outputs

- .csv file containing the new multiregional Input-Output table.
- pandas DataFrame with a multiregional Input-Output table

vtra.plot package

Submodules

vtra.plot.adaptation_box_plots module

Plot adaptation boxplots

main()

plot_boxplots(*input_data*, *input_labels*, *input_colors*, *x_label*, *y_label*, *plot_title*, *plot_file_path*)

plot_boxplots_subplots(*scenario_groups*, *input_data*, *x_label*, *y_label*, *plot_title*, *plot_file_path*)

vtra.plot.admin_map module

Maps of admin boundaries in Vietnam

plot_admin_map()

plot_admin_map_with_regions_highlighted()

vtra.plot.air_network_flows module

Air network flows map

main()

vtra.plot.air_network_flows_max_scales module

Air network flows map

main()

vtra.plot.air_network_map module

Air network map

main()

vtra.plot.coastal_network_flows module

Coastal network flows map

main()

vtra.plot.coastal_network_flows_max_scales module

Coastal network flows map

main()

vtra.plot.coastal_network_map module

Coastal network map

main()

vtra.plot.cost_benefits module

Road network flows

main()

plot_cumsum_ranges(*input_data_list*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_file_path*)

plot_many_ranges(*input_dfs*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_labels*, *plot_file_path*)

vtra.plot.create_crop_maps module

Crop maps

main()

vtra.plot.district_center_heatmap module

Road network (flows to commune centres) maps

main()

vtra.plot.inland_network_flows module

Inland network flows map

main()

vtra.plot.inland_network_flows_max_scales module

Inland network flows map

main()

vtra.plot.inland_network_map module

Inland network map

main()

vtra.plot.multimodal_network_map module

Multimodal network map

main()

vtra.plot.national_hazard_exposure_plots module

National hazard exposure maps

main()

vtra.plot.national_rail_risks module

Rail hazard exposure maps

main()

vtra.plot.national_roads_risks_and_adaptation module

Road network risks and adaptation maps

main()

vtra.plot.network_rerouting_losses module

Network rerouting loss maps

main(*mode*)

vtra.plot.plot_range_provinces module

Plot adaptation cost ranges (provincial results)

main()

plot_many_ranges(*input_dfs*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_labels*, *plot_file_path*)

plot_many_ranges_subplots(*input_dfs*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_labels*, *plot_file_path*)

plot_ranges(*input_data*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_label*, *plot_file_path*)

vtra.plot.plot_ranges module

Plot adaptation cost ranges (national results)

main()

plot_many_ranges(*input_dfs*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_labels*, *plot_file_path*)

plot_many_ranges_subplots(*input_dfs*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_labels*, *plot_file_path*)

plot_ranges(*input_data*, *division_factor*, *x_label*, *y_label*, *plot_title*, *plot_color*, *plot_label*, *plot_file_path*)

vtra.plot.province_hazard_exposure_plots module

Provincial road hazard exposure maps

main()

vtra.plot.province_roads_failures module

Provincial road network loss maps

main()

vtra.plot.province_roads_maps module

Provincial road network maps

main()

vtra.plot.province_roads_risks_and_adaptation module

Provincial road network risks and adaptation maps

main()

vtra.plot.rail_network_failures module

Rail network loss maps

main()

vtra.plot.rail_network_failures_multi_modal module

Rail network loss maps

main()

vtra.plot.rail_network_flows module

Rail network flows map

main()

vtra.plot.rail_network_flows_max_scales module

Rail network flows map

main()

vtra.plot.rail_network_map module

Rail network map

main()

vtra.plot.rail_network_routes module

Rail network routes map

main()

vtra.plot.rail_transfers module

Rail network transer map

main()

vtra.plot.rice_atlas_maps module

Rice atlas maps

main()

vtra.plot.road_network_failures module

Road network failure maps

main()

vtra.plot.road_network_failures_multi_modal module

Road network failure maps (multi-modal)

main()

vtra.plot.road_network_flows module

Road network flow maps

main()

vtra.plot.road_network_flows_max_scales module

Road network flow maps

main()

vtra.plot.road_network_map module

Road network map

main()

vtra.plot.road_transfers module

Road network transfer maps

main()

vtra.plot.water_network_flows module

Water network flows

main()

vtra.plot.water_network_map module

Water network map

main()

vtra.plot.water_transfers module

Water network transfers maps

main(*mode*)

vtra.preprocess package

Submodules

vtra.preprocess.convert_hazard_data module

Pre-process hazard data

Purpose

Convert GeoTiff raster hazard datasets to shapefiles based on masking and selecting values from

- Single-band raster files
- Multi-band (3-bands) raster files

Input data requirements

1. Correct paths to all hazard datasets
2. **Single-band GeoTiff hazard raster files with:**
    - values - between 0 and 1000
    - raster grid geometry
    - projection systems: Default assumed = EPSG:32648
3. **Multi-band GeoTiff hazard raster files with:**
    - 3-bands
    - values - in each band between 0 and 255
    - raster grid geometry
    - projection systems: Default assumed = EPSG:32648

Results

1. **Shapefiles whose names show the hazard models and their selected range of values**
    - ID - equal to 1
    - geometry - Shapely Polygon outline of selected hazard

convert(*threshold*, *infile*, *tmpfile_1*, *outfile*)

Convert GeoTiff raster file to Shapefile with geometries based on raster threshold less that 999

Parameters

- threshold - Float value of lower bound of GeoTiff threshold value to be selected
- infile - String name of input GeoTff file path
- tmpfile_1 - Stirng name of tmp file 1
- outfile - Stirng name of output shapefile

Outputs Shapefile with Polygon geometries of rasters based on raster values above a threshold

convert_geotiff_to_vector_with_multibands(*band_colors*, *infile*, *infile_epsg*, *tmpfile_1*, *tmpfile_2*, *outfile*)

Convert multi-band GeoTiff raster file to Shapefile with geometries based on raster band color values

Parameters

- band_colors - Tuple with 3-values each corresponding to the values in raster bands
- infile - String name of input GeoTff file path
- infile_epsg - Integer value of EPSG Projection number of raster
- tmpfile_1 - Stirng name of tmp file 1
- tmpfile_2 - Stirng name of tmp file 2
- outfile - Stirng name of output shapefile

Outputs Shapefile with Polygon geometries of rasters based on raster band values

convert_geotiff_to_vector_with_threshold(*from_threshold*, *to_threshold*, *infile*, *infile_epsg*, *tmpfile_1*, *tmpfile_2*, *outfile*)

Convert GeoTiff raster file to Shapefile with geometries based on raster threshold ranges

Parameters

- from_threshold - Float value of lower bound of GeoTiff threshold value to be selected
- to_threshold - Float value of upper bound of GeoTiff threshold value to be selected

- infile - String name of input GeoTff file path
- infile_epsg - Integer value of EPSG Projection number of raster
- tmpfile_1 - Stirng name of tmp file 1
- tmpfile_2 - Stirng name of tmp file 2
- outfile - Stirng name of output shapefile

Outputs Shapefile with Polygon geometries of rasters based on raster threshold ranges

glofris_data_details(*file_name*, *root_dir*)

Read names of GLOFRIS files and create attributes

Parameters

- file_name - String name of GeoTff file
- root_dir - String path to directory of file

Outputs

df - Pandas DataFrame written to csv file with columns:

- file_name - String
- hazard_type - String
- year - Integer: 2016 or 2030
- climate_scenario - String: RCP4.5 or RCP8.5 or none
- probability - Float: 1/(return period)
- banded - Boolean: True or False
- bands - Integer

main()

Process hazard data

1. **Specify the paths from where to read and write:**
   - Input data
   - Hazard data
2. **Supply input data and parameters**
   - Thresholds of flood hazards
   - Values of bands to be selected
   - Color code of multi-band rasters
   - Specific file names that might require some specific operations

raster_projections_and_databands(*file_path*)

Extract projection, data bands numbers and valuees from raster

Parameters

- file_path - String name of input GeoTff file path

Outputs

- counts - Number of bans in raster
- crs - Projection system of raster
- data_vals - Numpy array of raster values

raster_rewrite(*in_raster*, *out_raster*, *nodata*)

Rewrite a raster to reproject and change no data value

Parameters

- in_raster - String name of input GeoTff file path
- out_raster - String name of output GeoTff file path
- nodata - Float value of data that is treated as no data

Outputs Reproject and replace raster with nodata = -1

vtra.preprocess.create_transport_networks module

Pre-process networks

Purpose

Creating post-processed transport networks with attributes

From pre-processed input Shapefiles and collected network attributes data

For all Province road networks: [‘Lao Cai’, ‘Binh Dinh’, ‘Thanh Hoa’]

For all transport modes at national scale: [‘road’, ‘rail’, ‘air’, ‘inland’, ‘coastal’,’multi’]

Input data requirements

Correct paths to all files and correct input parameters

Edge and node shapefiles for all Province roads and national-scale networks

All Geometries in Edge Shapefiles should be valid LineStrings

All Geometries in Node Shapefiles should be valid Points

A. **Node Shapefiles should contain following column names and attributes**
    - node_id - String node ID
    - geometry - Shapely Point geometry of nodes
    - attributes - Multiple types depending upon sector and context
B. **Edge Shapefiles should contain following column names and attributes:**
    - edge_id - String Edge ID
    - from_node - String node ID that should be present in node_id column
    - to_node - String node ID that should be present in node_id column
    - geometry - Shapely LineString geometry of edges

Results

1. Excel sheets with post-processed network nodes and edges
2. Shapefiles with post-processed network nodes and edges
3. **All nodes have the following attributes:**
    - node_id - String Node ID
    - name - String name in Vietnamese/English
    - tons - Float assigned cargo freight tonnage using node
    - population - Float assigned passenger/population number using node
    - capacity - Float assigned capacity in tons/passenger numbers/other units
    - geometry - Shapely Point geometry of node

4. **Attributes only present in inland and coastal port nodes**
    - port_type - String name of type of port: inland or sea
    - port_class - String name of class of port: class1A (international) or class1 (domestic hub)
5. **All edges have the following attributes:**
    - edge_id - String edge ID
    - g_id - Integer edge ID
    - from_node - String node ID that should be present in node_id column
    - to_node - String node ID that should be present in node_id column
    - geometry - Shapely LineString geometry of edge
    - terrain - String name of terrain of edge
    - level - Integer number for edge level: National, Provincial, Local, etc.
    - width - Float width in meters of edge
    - length - Float estimated length in kilometers of edge
    - min_speed - Float estimated minimum speed in km/hr on edge
    - max_speed - Float estimated maximum speed in km/hr on edge
    - min_time - Float estimated minimum time of travel in hours on edge
    - max_time - Float estimated maximum time of travel in hours on edge
    - min_time_cost - Float estimated minimum cost of time in USD on edge
    - max_time_cost - Float estimated maximum cost of time in USD on edge
    - min_tariff_cost - Float estimated minimum tariff cost in USD on edge
    - max_tariff_cost - Float estimated maximum tariff cost in USD on edge
    - vehicle_co - Integer number of daily vehicle counts on edge
6. **Attributes only present in Province and national roads edges**
    - surface - String value for surface
    - road_class - Integer between 1 and 6
    - road_cond - String value: paved or unpaved
    - asset_type - String name of type of asset

References

1. Pant, R., Koks, E.E., Russell, T., Schoenmakers, R. & Hall, J.W. (2018). Analysis and development of model for addressing climate change/disaster risks in multi-modal transport networks in Vietnam. Final Report, Oxford Infrastructure Analytics Ltd., Oxford, UK.

2. All input data folders and files referred to in the code below.

main()

Pre-process networks

1. **Specify the paths from where to read and write:**
    - Input data
    - Intermediate calcuations data
    - Output results
2. **Supply input data and parameters**

- Paths of the mode files: List of tuples of strings
- Names of modes: List of strings
- Unit weight of vehicle assumed for each mode: List of float types
- Ranges of usage factors for each mode to represent uncertainty in cost estimations: List of tuples of float types
- Ranges of speeds for each mode to represent uncertainty in speeds: List of tuple of float types
- Names of all industry sector and crops in VITRANSS2 and IFPRI datasets: List of string types
- Names of commodity/industry columns for which min-max tonnage column names already exist: List of string types
- Percentage of OD flow we want to send along path: Float type

3. **Give the paths to the input data files:**
   - Pre-proccesed network shapefiles
   - Costs of modes Excel file
   - Road properties Excel file
4. Specify the output files and paths to be created

vtra.preprocess.cvts module

vtra.preprocess.cvts_speeds module

main()

Traffic speed assignment script vehicle_id, edge_path, time_stamp

vtra.preprocess.national_modes_od_creation module

Pre-process OD matrix

Purpose

Create national scale OD matrices at node and province levels from:

- VITRANSS2 province-scale OD data
- IFPRI crop data at 1km resolution

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Excel file with VITRANSS2 OD data with columns:**
   - o - Integer values of Origin Province ID
   - d - Integer values of Destination Province ID
   - industry and crop names - Float values of tons between OD Provinces
   - modes - Float values to infer model split values for OD Provinces
3. **Geotiff files with IFPRI crop data:**
   - tons - Float values of production tonnage at each grid cell
   - geometry - Raster grid cell geometry
4. **Shapefile of RiceAtlas data:**

- month production columns - tonnage of rice for each month
- geometry - Shapely Polygon geometry of Provinces

5. **Shapefile of Provinces**
   - od_id - Integer Province ID corresponding to OD ID
   - name_eng - String name of Province in English
   - geometry - Shapely Polygon geometry of Provinces
6. **Shapefile of Communes**
   - population - Float values of populations in Communes
   - geometry - Shapely Polygon geometry of Communes
7. **Shapefiles of network nodes**
   - node_id - String node ID
   - geometry - Shapely point geometry of nodes
8. **Shapefiels of network edges**
   - vehicle_co - Count of vehiles only for roads
   - geometry - Shapely LineString geometry of edges

Results

1. **Excel workbook with sheet of mode-wise and total OD flows**
   - origin - String node ID of origin node
   - destination - String node ID of destination node
   - o_region - String names of origin Province
   - d_region - String names of destination Province
   - commodity_names - Float values of daily tonnages of commodities/industries between OD nodes from VITRANSS2 and IFPRI data (except rice)
   - min_rice - Float values of minimum daily tonnages of rice between OD nodes
   - max_rice - Float values of maximum daily tonnages of rice between OD nodes
   - min_tons - Float values of minimum daily tonnages between OD nodes
   - max_tons - Float values of maximum daily tonnages between OD nodes

References

1. Pant, R., Koks, E.E., Russell, T., Schoenmakers, R. & Hall, J.W. (2018). Analysis and development of model for addressing climate change/disaster risks in multi-modal transport networks in Vietnam. Final Report, Oxford Infrastructure Analytics Ltd., Oxford, UK.
2. All input data folders and files referred to in the code below.

assign_crop_od_flows_to_nodes(*national_ods_df*, *province_path*, *province_name_col*, *calc_path*, *crop_data_path*, *crop_names*, *rice_prod_months*, *modes_df*, *modes*, *od_fracs_crops*, *o_id_col*, *d_id_col*)

Assign IFPRI crop values to OD nodes based on VITRANSS 2 OD distributions

- Based on VITRANSS 2 OD distributions

Parameters

- national_ods_df - List of lists of Pandas dataframes
- province_path - Path of province shapefile

- province_name_col - String name of column containing province names
- calc_path - Path to store intermediary calculations
- crop_data_path - Path to crop datasets
- crop_names - List of string of crop names in IFPRI datasets
- rice_prod_months - Geopandas dataframe of RiceAtlas crop values with minimum and maximum monthly production as fraction of annual production
- modes_df - List of Geopnadas dataframes with nodes of each transport mode
- modes - List of strings of names of transport modes
- od_fracs_crops - Pandas dataframe of crop OD distributions and modal splits as per VITRANSS 2 data
- o_id_col - String name of Origin province ID column
- d_id_col - String name of Destination province ID column

- **Outputs**

  national_ods_df - List of Lists of Pandas dataframes with columns:

  - origin - Origin node ID
  - o_region - Origin province name
  - destination - Destination node ID
  - d_region - Destination province ID
  - min_crop - Minimum Tonnage values for the named crop
  - max_crop - Maximum Tonnage values for the named crop

assign_daily_min_max_tons_rice(*crop_df*, *rice_prod_df*)

Estimate minimum and maximum daily rice tonnages

Parameters

- crop_df - Geopandas dataframe of crop points with annual tonnages
- rice_prod_df - Geopandas dataframe of RiceAtlas crop values with minimum and maximum monthly production as fraction of annual production

Outputs

crop_df - Geopandas dataframe of crop points with new columns:

- min_rice - Minimum daily rice tonnages
- max_rice - Maximum daily rice tonnages

assign_industry_od_flows_to_nodes(*national_ods_df*, *ind_cols*, *modes_df*, *modes*, *od_fracs*, *o_id_col*, *d_id_col*)

Assign VITRANSS 2 OD flows to nodes

Parameters

- national_ods_df - List of lists of Pandas dataframes
- ind_cols - List of strings of names of indsutry columns
- modes_df - List of Geopnadas dataframes with nodes of each transport mode
- modes - List of strings of names of transport modes
- od_fracs - Pandas dataframe of Industry OD flows and modal splits
- o_id_col - String name of Origin province ID column

- d_id_col - String name of Destination province ID column

Outputs

national_ods_df - List of Lists of Pandas dataframes with columns:

- origin - Origin node ID
- o_region - Origin province name
- destination - Destination node ID
- d_region - Destination province ID
- ind - Tonnage values for the named industry

assign_node_weights_by_commune_population_proximity(*commune_path*, *nodes*, *commune_pop_col*)

Assign weights to nodes based on their nearest commune populations

- By finding the communes that intersect with the Voronoi extents of nodes

Parameters

- commune_path - Path of commune shapefile
- nodes_in - Path of nodes shapefile
- commune_pop_col - String name of column containing commune population values

Outputs

- nodes - Geopandas dataframe of nodes with new column called weight

assign_province_name_id_to_nodes(*province_path*, *nodes_in*, *province_name_col*, *province_id_col*)

Match the nodes to their province names and province IDs

- By finding the province that contains or is nearest to the node

Parameters

- province_path - Path of province shapefile
- nodes_in - Path of nodes shapefile
- province_name_col - String name of column containing province names
- province_id_col - String name of column containing province ID's that match VITRANSS 2 OD ids

Outputs

- nodes - Geopandas dataframe of nodes with new columns called province_name and od_id

assign_road_weights(*nodes*, *edges_in*, *aadt_column*)

Assign weights to nodes on the road network

- By finding the total AADT counts converging on the node

Parameters

- nodes - Geopandas dataframe of nodes
- edges_in - Path of edges shapefile
- aadt_column - String name of column containing AADT values

Outputs

- nodes - Geopandas dataframe of nodes with new column called weight

ifpri_crop_od_split (*vitranss2_od_data_file*, *modes*, *o_id_col*, *d_id_col*, *crop_cols*)

Create IFPRI crop OD modal split estimates from VITRANSS2 OD values

- Combine the crop-wise OD values with the mode-wise OD values
- Estimate the OD modal-split from the mode-wsie OD values

Parameters

- vitranss2_od_data_file - Excel file with VITRANSS2 OD data
- modes - List of strings of mode types, e.g. ['road', 'rail', 'air', 'inland', 'coastal']
- o_id_col - String name of name of Origin column
- d_id_col - String name of name of Destination column
- crop_cols - List of strings of crop names in VITRANSS 2 OD data

Outputs

- od_fracs_crops - Pandas dataframe of VITRANSS2 crop OD data with modal splits

main ()

Pre-process national OD matrix

1. **Specify the paths from where to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results
2. **Supply input data and parameters**
   - Names of modes: List of strings
   - OD column names in OD file: String types
   - Names of industry columns in VITRANSS2 data: List of string types
   - Names of crop columns in VITRANSS2 data: List of string types
   - Names of crops in IFPRI crop data: List of string types
   - Names of months in Rice Atlas data: List of string types
3. **Give the paths to the input data files:**
   - Network nodes files
   - VITRANSS 2 OD file
   - IFPRI crop data files
   - Rice Atlas data shapefile
   - Province boundary and stats data shapefile
   - Commune boundary and stats data shapefile
4. Specify the output files and paths to be created

riceatlas_crop_minmax (*riceatlas_crop_file*, *crop_month_fields*)

Create MIN_MAX fractions of rice production estimates for provinces

- By reading data from the RiceAtlas data
- And estimating the minimum and maximum monthly values > 0

Parameters

- riceatlas_crop_file - Shapefile with RiceAtlas data

- crop_month_fields - List of strings of names of columns indicating rice monthly production

Outputs

rice_prod_months - Geopandas dataframe of RiceAtlas crop values

- min_frac - minimum month of production as fraction of annual production
- max_frac - maximum month of production as fraction of annual production

`vitranss2_od_split`(*vitranss2_od_data_file*, *modes*, *o_id_col*, *d_id_col*)

Create VITRANSS2 OD modal split estimates

- Combine the commodity-wise OD values with the mode-wise OD values
- Estimate the OD modal-split from the mode-wsie OD values

Parameters

- vitranss2_od_data_file - Excel file with VITRANSS2 OD data
- modes - List of strings of mode types, e.g. ['road', 'rail', 'air', 'inland', 'coastal']
- o_id_col - String name of name of Origin column
- d_id_col - String name of name of Destination column

Outputs

- od_fracs - Pandas dataframe of VITRANSS 2 commodity OD data with modal splits

vtra.preprocess.province_roads_access_od_creation module

Pre-process accessibility-based provincial OD matrix

Purpose

Create province scale OD matrices between roads connecting villages to nearest communes:

- Net revenue estimates of commune villages
- IFPRI crop data at 1km resolution

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Geotiff files with IFPRI crop data:**
   - tons - Float values of production tonnage at each grid cell
   - geometry - Raster grid cell geometry
3. **Shapefile of RiceAtlas data:**
   - month production columns - tonnage of rice for each month
   - geometry - Shapely Polygon geometry of Provinces
4. **Shapefile of Provinces**
   - od_id - Integer Province ID corresponding to OD ID
   - name_eng - String name of Province in English
   - geometry - Shapely Polygon geometry of Provinces
5. **Shapefile of Communes**
   - population - Float values of populations in Communes
   - nfrims - Float values of number of firms in Provinces

- netrevenue - Float values of Net Revenue in Provinces
- argi_prop - Float values of proportion of agrivculture firms in Provinces
- geometry - Shapely Polygon geometry of Communes

6. **Shapefiles of network nodes**
   - node_id - String node ID
   - geometry - Shapely point geometry of nodes
7. **Shapefiles of network edges**
   - vehicle_co - Count of vehiles only for roads
   - geometry - Shapely LineString geometry of edges
8. **Shapefiles of Commune center points**
   - object_id - Integer ID of point
   - geometry - Shapely point geometry of points
9. **Shapefiles of Village center points**
   - object_id - Integer ID of points
   - geometry - Shapely point geometry of points

Results

1. **Excel workbook with sheet of province-wise OD flows**
   - origin - String node ID of origin node
   - destination - String node ID of destination node
   - crop_names - Float values of daily tonnages of IFPRI crops (except rice) between OD nodes
   - min_rice - Float values of minimum daily tonnages of rice between OD nodes
   - max_rice - Float values of maximum daily tonnages of rice between OD nodes
   - min_croptons - Float values of minimum daily tonnages of crops between OD nodes
   - max_croptons - Float values of maximum daily tonnages of crops between OD nodes
   - min_agrirev - Float value of Minimum daily revenue of agriculture firms between OD nodes
   - max_agrirev - Float value of Maximum daily revenue of agriculture firms between OD nodes
   - min_noagrirev - Float value of Minimum daily revenue of non-agriculture firms between OD nodes
   - max_noagrirev - Float value of Maximum daily revenue of non-agriculture firms between OD nodes
   - min_netrev - Float value of Minimum daily revenue of all firms between OD nodes
   - max_netrev - Float value of Maximum daily revenue of all firms between OD nodes

References

1. Pant, R., Koks, E.E., Russell, T., Schoenmakers, R. & Hall, J.W. (2018). Analysis and development of model for addressing climate change/disaster risks in multi-modal transport networks in Vietnam. Final Report, Oxford Infrastructure Analytics Ltd., Oxford, UK.
2. All input data folders and files referred to in the code below.

assign_io_rev_costs_crops(*x*, *cost_dataframe*, *rice_prod_file*, *crop_month_fields*, *province*, *x_cols*, *ex_rate*)

Assign crop tonnages to daily net revenues

Parameters

- x - Pandas DataFrame of values
- cost_dataframe - Pandas DataFrame of conversion of tonnages to net revenues
- rice_prod_file - Shapefile of RiceAtlas monthly production value
- province - Stirng name of province
- x_cols - List of string names of crops
- ex_rate - Exchange rate from VND millions to USD

Outputs

- min_croprev - Float value of Minimum daily revenue of crops
- max_croprev - Float value of Maximum daily revenue of crops

assign_monthly_tons_crops (*x*, *rice_prod_file*, *crop_month_fields*, *province*, *x_cols*)

Assign crop tonnages to OD pairs

Parameters

- x - Pandas DataFrame of values
- rice_prod_file - Shapefile of RiceAtlas monthly production value
- crop_month_fields - Lsit of strings of month columns in Rice Atlas shapefile
- province - Stirng name of province
- x_cols - List of string names of crops

Outputs

- min_croptons - Float value of Minimum daily tonnages of crops
- max_croptons - Float value of Maximum daily tonnages of crops

crop_od_pairs (*start_points*, *end_points*, *crop_name*)

Assign crop tonnages to OD pairs

Parameters

- start_points - GeoDataFrame of start points for Origins
- end_points - GeoDataFrame of potential end points for Destinations
- crop_name - String name of crop

Outputs

od_pairs_df - Pandas DataFrame wit columns:

- origin - Origin node ID
- destination - Destination node ID
- crop - Tonnage values for the named crop
- netrev_argi - Daily Net revenue of agriculture firms in USD
- netrev_noargi - Daily Net revenue of non-agriculture firms in USD

crop_values_to_province_od_nodes (*province_ods_df*, *province_geom*, *calc_path*, *crop_data_path*, *crop_names*, *nodes*, *sindex_nodes*, *prov_commune_center*, *sindex_commune_center*, *node_id*, *object_id*)

Assign IFPRI crop values to OD nodes in provinces

- Based on finding nearest nodes to crop production sites as Origins
- And finding nearest commune centers as Destinations

Parameters

- province_ods_df - List of lists of Pandas dataframes
- province_geom - Shapely Geometry of province
- calc_path - Path to store intermediary calculations
- crop_data_path - Path to crop datasets
- crop_names - List of string of crop names in IFPRI datasets
- nodes - GeoDataFrame of province road nodes
- sindex_nodes - Spatial index of province road nodes
- prov_commune_center - GeoDataFrame of province commune center points
- sindex_commune_center - Spatial index of commune center points
- node_id - String name of Node ID column
- object_id - String name of commune ID column

Outputs

province_ods_df - List of Lists of Pandas dataframes with columns:

- origin - Origin node ID
- destination - Destination node ID
- crop - Tonnage values for the named crop

main()

Pre-process provincial-scale OD

1. **Specify the paths from where to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results
2. **Supply input data and parameters**
   - Names of the Provinces: List of strings
   - Exchange rate to convert 2012 Net revenue in million VND values to USD in 2016
   - Names of crops in IFPRI crop data
   - Names of months in Rice Atlas data
   - Name of column for netrevenue of communes in VND millions
   - Name of column for numebr of firms in communes
   - Name of column for proportion of agriculture firms in communes
   - Name of Node ID column
   - Name of commune ID column
3. **Give the paths to the input data files:**
   - Network nodes files
   - IFPRI crop data files
   - Rice Atlas data shapefile
   - Province boundary and stats data shapefile
   - Commune boundary and stats data shapefile

- Population points shapefile for locations of villages
- Commune center points shapefile

netrev_od_pairs (*start_points*, *end_points*)

Assign crop tonnages to OD pairs

Parameters

- start_points - GeoDataFrame of start points for Origins
- end_points - GeoDataFrame of potential end points for Destinations

Outputs

od_pairs_df - Pandas DataFrame with columns:

- origin - Origin node ID
- destination - Destination node ID
- netrev_argi - Net revenue of agriculture firms
- netrev_noargi - Net revenue of non-agriculture firms

netrevenue_values_to_province_od_nodes (*province_ods_df*, *prov_communes*, *commune_sindex*, *netrevenue*, *n_firms*, *agri_prop*, *prov_pop*, *prov_pop_sindex*, *nodes*, *sindex_nodes*, *prov_commune_center*, *sindex_commune_center*, *node_id*, *object_id*, *exchange_rate*)

Assign commune level netrevenue values to OD nodes in provinces

- Based on finding nearest nodes to village points with netrevenues as Origins
- And finding nearest commune centers as Destinations

Parameters

- province_ods_df - List of lists of Pandas dataframes
- prov_communes - GeoDataFrame of commune level statistics
- commune_sindex - Spatial index of communes
- netrevenue - String name of column for netrevenue of communes in VND millions
- nfirm - String name of column for numebr of firms in communes
- agri_prop - Stirng name of column for proportion of agriculture firms in communes
- prov_pop - GeoDataFrame of population points in Province
- prov_pop_sindex - Spatial index of population points in Province
- nodes - GeoDataFrame of province road nodes
- sindex_nodes - Spatial index of province road nodes
- prov_commune_center - GeoDataFrame of province commune center points
- sindex_commune_center - Spatial index of commune center points
- node_id - String name of Node ID column
- object_id - String name of commune ID column
- exchange_rate - Float value for exchange rate from VND million to USD

Outputs

province_ods_df - List of Lists of Pandas dataframes with columns:

- origin - Origin node ID

- destination - Destination node ID
- netrev_argi - Net revenue of agriculture firms
- netrev_noargi - Net revenue of non-agriculture firms

vtra.preprocess.transport_network_inputs module

Utility functions for transport networks

Purpose

Helper functions to create post-processeed networks with attributes from specific types of input datasets

References

1. Pant, R., Koks, E.E., Russell, T., Schoenmakers, R. & Hall, J.W. (2018). Analysis and development of model for addressing climate change/disaster risks in multi-modal transport networks in Vietnam. Final Report, Oxford Infrastructure Analytics Ltd., Oxford, UK.
2. All input data folders and files referred to in the code below.

assign_asset_type_to_province_roads(*x*)

Assign asset types to roads assets in Vietnam

The types are assigned based on our understanding of: 1. The reported asset code in the data

Parameters x - Pandas DataFrame with numeric asset code

Returns asset type - Which is either of (Bridge, Dam, Culvert, Tunnel, Spillway, Road)

assign_asset_type_to_province_roads_from_file(*asset_code*, *asset_type_list*)

Assign asset types to roads assets in Vietnam based on values in file

The types are assigned based on our understanding of: 1. The reported asset code in the data

Parameters

- asset_code - Numeric value for code of asset
- asset_type_list - List of Strings wiht names of asset types

Returns asset_type - String name of type of asset

assign_assumed_width_to_national_roads_from_file(*x*, *flat_width_range_list*, *mountain_width_range_list*)

Assign widths to national roads assets in Vietnam

The widths are assigned based on our understanding of: 1. The class of the road which is not reliable 2. The number of lanes 3. The terrain of the road

Parameters

- **x - Pandas DataFrame row with values**
  - road_class - Integer value of road class
  - lanenum__s - Integer value of number of lanes on road
- flat_width_range_list - List of tuples containing (from_width, to_width, assumed_width)
- moiuntain_width_range_list - List of tuples containing (from_width, to_width, assumed_width)

Returns assumed_width - Float assigned width of the road asset based on design specifications

assign_assumed_width_to_province_roads(*x*)

Assign widths to Province roads assets in Vietnam

Parameters x : int value for width of asset

Returns int assigned width of the road asset based on design specifications

assign_assumed_width_to_province_roads_from_file(*asset_width*, *width_range_list*)

Assign widths to Province roads assets in Vietnam

The widths are assigned based on our understanding of:

1. The reported width in the data which is not reliable
2. A design specification based understanding of the assumed width based on ranges of values

Parameters

- asset_width - Numeric value for width of asset
- width_range_list - List of tuples containing (from_width, to_width, assumed_width)

Returns assumed_width - assigned width of the raod asset based on design specifications

assign_min_max_speeds_to_national_roads_from_file(*x*, *flat_width_range_list*, *mountain_width_range_list*)

Assign speeds to national roads in Vietnam

The speeds are assigned based on our understanding of: 1. The class of the road 2. The estimated speed from the CVTS data 3. The terrain of the road

Parameters

x - Pandas DataFrame of values

- road_class - Integer value of road class
- terrain - String value of road terrain
- est_speed - Float value of estimated speed from CVTS data

- flat_width_range_list - List of tuples containing design speeds
- moiuntain_width_range_list - List of tuples containing design speeds

Returns

- Float minimum assigned speed in km/hr
- Float maximum assigned speed in km/hr

assign_minmax_tariff_costs_multi_modal_apply(*x*, *cost_dataframe*)

Assign tariff costs on multi-modal network links in Vietnam

Parameters

- **x - Pandas dataframe with values**
  - port_type - String name of port type
  - from_mode - String name of mode
  - to_mode - String name of mode
  - other_mode - String name of mode
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_tariff_cost - Float minimum assigned tariff cost in USD/ton
- max_tariff_cost - Float maximum assigned tariff cost in USD/ton

assign_minmax_tariff_costs_national_roads_apply(*x*, *cost_dataframe*)

Assign tariff costs on national roads in Vietnam

The costs are assigned based on our understanding of:

1. The vehicle counts on roads

Parameters

- **x - Pandas dataframe with values**
  - vehicle_co - Count of number of vehicles on road
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_tariff_cost - Float minimum assigned tariff cost in USD/ton
- max_tariff_cost - Float maximum assigned tariff cost in USD/ton

assign_minmax_tariff_costs_networks_apply(*x*, *cost_dataframe*)

Assign tariff costs on networks in Vietnam

Parameters

- **x - Pandas dataframe with values**
  - length - Float length of edge in km
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_tariff_cost - Float minimum assigned tariff cost in USD/ton
- max_tariff_cost - Float maximum assigned tariff cost in USD/ton

assign_minmax_tariff_costs_province_roads_apply(*x*, *cost_dataframe*)

Assign tariff costs on Province roads in Vietnam

The costs are assigned based on our understanding of:

1. The types of assets
2. The levels of classification of assets: 0-National, 1-Provinical, 2-Local, 3-Other
3. The terrain where the assets are located: Flat or Mountain or No information

Parameters

- **x - Pandas dataframe with values**
  - code - Numeric code for type of asset
  - level - Numeric code for level of asset
  - terrain - String value of the terrain of asset
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_tariff_cost - Float minimum assigned tariff cost in USD/ton
- max_tariff_cost - Float maximum assigned tariff cost in USD/ton

assign_minmax_time_costs_national_roads_apply(*x*, *cost_dataframe*)

Assign time costs on national roads in Vietnam

The costs are assigned based on our understanding of:

1. The vehicle counts on roads
2. The levels of classification of assets: 0-National, 1-Provinical, 2-Local, 3-Other
3. The terrain where the assets are located: Flat or Mountain or No information

Parameters

- **x - Pandas dataframe with values**

  - vehicle_co - Count of number of vehicles on road
  - code - Numeric code for type of asset
  - level - Numeric code for level of asset
  - terrain - String value of the terrain of asset
  - length - Float length of edge in km
  - min_speed - Float minimum assigned speed in km/hr
  - max_speed - Float maximum assigned speed in km/hr
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_time_cost - Float minimum assigned cost of time in USD
- max_time_cost - Float maximum assigned cost of time in USD

assign_minmax_time_costs_networks_apply(*x*, *cost_dataframe*)

Assign time costs on networks in Vietnam

Parameters

- **x - Pandas dataframe with values**
  - length - Float length of edge in km
  - min_speed - Float minimum assigned speed in km/hr
  - max_speed - Float maximum assigned speed in km/hr
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_time_cost - Float minimum assigned cost of time in USD
- max_time_cost - Float maximum assigned cost of time in USD

assign_minmax_time_costs_province_roads_apply(*x*, *cost_dataframe*)

Assign time costs on Province roads in Vietnam

The costs are assigned based on our understanding of:

1. The types of assets
2. The levels of classification of assets: 0-National, 1-Provinical, 2-Local, 3-Other
3. The terrain where the assets are located: Flat or Mountain or No information

Parameters

- **x - Pandas dataframe with values**
  - code - Numeric code for type of asset
  - level - Numeric code for level of asset
  - terrain - String value of the terrain of asset
  - length - Float length of edge in km
  - min_speed - Float minimum assigned speed in km/hr
  - max_speed - Float maximum assigned speed in km/hr
- cost_dataframe - Pandas Dataframe with costs

Returns

- min_time_cost - Float minimum assigned cost of time in USD

- max_time_cost - Float maximum assigned cost of time in USD

assign_minmax_travel_speeds_province_roads_apply(*x*)

Assign travel speeds to roads assets in Vietnam

The speeds are assigned based on our understanding of:

1. The types of assets
2. The levels of classification of assets: 0-National, 1-Provinical, 2-Local, 3-Other
3. The terrain where the assets are located: Flat or Mountain or No information

Parameters

x - Pandas dataframe with values

- code - Numeric code for type of asset
- level - Numeric code for level of asset
- terrain - String value of the terrain of asset

Returns

- Float minimum assigned speed in km/hr
- Float maximum assigned speed in km/hr

assign_national_road_class(*x*)

Assign road speeds to national roads

Parameters

x - Pandas DataFrame of values

- capkth__ca - String value of road class
- vehicle_co - Float value of number of vehicles on road

Returns

- Integer value of road class

assign_national_road_conditions(*x*)

Assign road conditions as paved or unpaved to national roads

Parameters

x - Pandas DataFrame of values

- **loai_mat__** - String value of road surface

Returns String value of road as paved or unpaved

assign_national_road_terrain(*x*)

Assign terrain as flat or mountain to national roads

Parameters

x - Pandas DataFrame of values

- **dia_hinh__** - String value of type of terrain

Returns String value of terrain as flat or mountain

assign_province_road_conditions(*x*)

Assign road conditions as paved or unpaved to Province roads

Parameters

x - Pandas DataFrame of values

- code - Numeric code for type of asset

- level - Numeric code for level of asset

Returns String value as paved or unpaved

create_port_names(*x*, *port_names_df*)

Add port names in Vietnamese to port data

Parameters

- **x - Pandas DataFrame with values**
  - port_type - String type of port
  - cangbenid - Integer ID of inland port
  - objectid - Integer ID of sea port
- port_names_df - Pandas DataFrame with port names

Returns name - Vietnamese name of port

multi_modal_shapefile_to_dataframe(*edges_in*, *mode_properties_file*, *mode_name*, *length_threshold*, *usage_factors*)

Create multi-modal network dataframe from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- mode_properties_file - String path to Excel file with mode attributes
- mode_name - String name of mode
- length_threshold - Float value of threshold in km of length of multi-modal links
- usage_factor - Tuple of 2-float values between 0 and 1

Returns edges - Geopandas DataFrame with network edge topology and attributes

multi_modal_shapefile_to_network(*edges_in*, *mode_properties_file*, *mode_name*, *length_threshold*, *utilization_factors*)

Create multi-modal igraph network dataframe from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- mode_properties_file - String path to Excel file with mode attributes
- mode_name - String name of mode
- length_threshold - Float value of threshold in km of length of multi-modal links
- usage_factor - Tuple of 2-float values between 0 and 1

Returns G - Igraph object with network edge topology and attributes

national_road_shapefile_to_dataframe(*edges_in*, *road_properties_file*, *usage_factors*)

Create national network dataframe from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- road_properties_file - String path to Excel file with road attributes
- usage_factor - Tuple of 2-float values between 0 and 1

Returns edges: Geopandas DataFrame with network edge topology and attributes

national_road_shapefile_to_network(*edges_in*, *road_properties_file*, *usage_factors*)

Create national igraph network from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- road_properties_file - String path to Excel file with road attributes
- usage_factor - Tuple of 2-float values between 0 and 1

Returns G - Igraph object with network edge topology and attributes

network_shapefile_to_dataframe(*edges_in*, *mode_properties_file*, *mode_name*, *speed_min*, *speed_max*, *usage_factors*)

Create network dataframe from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- mode_properties_file - String path to Excel file with mode attributes
- mode_name - String name of mode
- speed_min - Float value of minimum assgined speed
- speed_max - Float value of maximum assgined speed
- usage_factor - Tuple of 2-float values between 0 and 1

Returns edges - Geopandas DataFrame with network edge topology and attributes

network_shapefile_to_network(*edges_in*, *mode_properties_file*, *mode_name*, *speed_min*, *speed_max*, *utilization_factors*)

Create igraph network from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- mode_properties_file - String path to Excel file with mode attributes
- mode_name - String name of mode
- speed_min - Float value of minimum assgined speed
- speed_max - Float value of maximum assgined speed
- usage_factor - Tuple of 2-float values between 0 and 1

Returns G - Igraph object with network edge topology and attributes

province_shapefile_to_dataframe(*edges_in*, *road_terrain*, *road_properties_file*, *usage_factors*)

Create province network dataframe from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- road_terrain - String name of terrain: flat or mountanious
- road_properties_file - String path to Excel file with road attributes
- usage_factor - Tuple of 2-float values between 0 and 1

Returns edges - Geopandas DataFrame with network edge topology and attributes

province_shapefile_to_network(*edges_in*, *road_terrain*, *road_properties_file*, *usage_factors*)

Create province igraph network from inputs

Parameters

- edges_in - String path to edges file/network Shapefile
- road_terrain - String name of terrain: flat or mountanious
- road_properties_file - String path to Excel file with road attributes
- usage_factor - Tuple of 2-float values between 0 and 1

Returns G - Igraph object with network edge topology and attributes

read_setor_nodes (*node_file_with_ids*, *sector*)
Create port data with attributes

Parameters

- ports_file_with_ids - String path of GeoDataFrame with port IDs
- sector - String path of sector

Returns ports_with_id - GeoPandas DataFrame with port attributes

read_waterway_ports (*ports_file_with_ids*, *ports_file_with_names*)
Create port data with attributes

Parameters

- ports_file_with_ids - String path of GeoDataFrame with port IDs
- ports_file_with_names - String path of GeoDataFrame with port names

Returns ports_with_id - GeoPandas DataFrame with port attributes

vtra.stats package

Submodules

vtra.stats.air_water_impacts module

Sum max/min total flow exposed under hazard scenarios at air and water network nodes

aggregate_flows (*nodes_df*, *flows_df*)

join_hazards (*nodes_with_flows_df*, *hazards_df*)

main ()

read_airports (*airports_file*)

read_flows (*flows_file*)

read_hazards (*hazard_file*)

read_ports (*ports_file_with_ids*, *ports_file_with_names*)

summarise (*nodes_with_hazards_df*)

vtra.stats.boundary_hazard_percentages module

Summarise network-hazard intersections per-boundary (district, commune or province)

Purpose

Collect network-hazard intersection attributes

- Combine with boundary Polygons to collect network-boundary intersection attributes
- Write final results to an Excel sheet

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Shapefiles of network-hazard intersections results with attributes:**
   - edge_id or node_id - String/Integer/Float Edge ID or Node ID of network
   - length - Float length of edge intersecting with hazards
   - geometry - Shapely geometry of edges as LineString or nodes as Points

3. **Shapefile of administrative boundaries of Vietnam with attributes:**
    - province_i - String/Integer ID of Province
    - pro_name_e - String name of Province in English
    - district_i - String/Integer ID of District
    - dis_name_e - String name of District in English
    - commune_id - String/Integer ID of Commune
    - name_eng - String name of Commune in English
    - geometry - Shapely geometry of boundary Polygon

Results

1. **Excel sheet of network-hazard-boundary intersection with attributes:**
    - edge_id/node_id - String name of intersecting edge ID or node ID
    - length - Float length of intersection of edge LineString and hazard Polygon: Only for edges
    - province_id - String/Integer ID of Province
    - province_name - String name of Province in English
    - district_id - String/Integer ID of District
    - district_name - String name of District in English
    - commune_id - String/Integer ID of Commune
    - commune_name - String name of Commune in English

main()

Summarise intersections

1. **Specify the paths from where you to read and write:**
    - Input data
    - Intermediate calcuations data
    - Output results
2. **Supply input data and parameters**
    - Names of the three Provinces - List of string types
    - Names of modes - List of strings
    - Names of output modes - List of strings
    - Names of hazard bands - List of integers
    - Names of hazard thresholds - List of integers
    - Condition 'Yes' or 'No' is the users wants to process results
3. **Give the paths to the input data files:**
    - Commune boundary and stats data shapefile
    - Hazard datasets description Excel file
    - String name of sheet in hazard datasets description Excel file
4. Specify the output files and paths to be created

vtra.stats.flow_senstivities module

Summarise hazard data

Get OD data and process it

main()

vtra.stats.national_roads_criticality_risk_stats module

Road network risks and adaptation maps

main()

vtra.stats.network_boundary_stats module

Summarise length of edges/number of nodes within each boundary (commune, district, province)

Purpose

Collect network attributes

- Combine with boundary Polygons to collect network-boundary intersection attributes
- Write final results to an Excel sheet

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Shapefiles of networks with attributes:**
    - edge_id or node_id - String/Integer/Float Edge ID or Node ID of network
    - length - Float length of edge intersecting with hazards
    - geometry - Shapely geometry of edges as LineString or nodes as Points
3. **Shapefile of administrative boundaries of Vietnam with attributes:**
    - province_i - String/Integer ID of Province
    - pro_name_e - String name of Province in English
    - district_i - String/Integer ID of District
    - dis_name_e - String name of District in English
    - commune_id - String/Integer ID of Commune
    - name_eng - String name of Commune in English
    - geometry - Shapely geometry of boundary Polygon

Results

1. **Excel sheet of network-hazard-boundary intersection with attributes:**
    - edge_id/node_id - String name of intersecting edge ID or node ID
    - length - Float length of intersection of edge LineString and hazard Polygon: Only for edges
    - province_id - String/Integer ID of Province
    - province_name - String name of Province in English
    - district_id - String/Integer ID of District
    - district_name - String name of District in English

- commune_id - String/Integer ID of Commune
- commune_name - String name of Commune in English

main()

Summarise

1. **Specify the paths from where you to read and write:**
   - Input data
   - Intermediate calcuations data
   - Output results
2. **Supply input data and parameters**
   - Names of the three Provinces - List of string types
   - Names of modes - List of strings
   - Names of output modes - List of strings
   - Names of hazard bands - List of integers
   - Names of hazard thresholds - List of integers
   - Condition 'Yes' or 'No' is the users wants to process results
3. **Give the paths to the input data files:**
   - Commune boundary and stats data shapefile
   - String name of sheet in hazard datasets description Excel file
4. Specify the output files and paths to be created

vtra.stats.network_hazard_stats module

Summarise per-hazard total intersections (for the whole system)

Purpose

Collect network-hazard intersection attributes

- Combine with boundary Polygons to collect network-boundary intersection attributes
- Write final results to an Excel sheet

Input data requirements

1. Correct paths to all files and correct input parameters
2. **Shapefiles of network-hazard intersections results with attributes:**
   - edge_id or node_id - String/Integer/Float Edge ID or Node ID of network
   - length - Float length of edge intersecting with hazards
   - geometry - Shapely geometry of edges as LineString or nodes as Points
3. **Shapefile of administrative boundaries of Vietnam with attributes:**
   - province_i - String/Integer ID of Province
   - pro_name_e - String name of Province in English
   - district_i - String/Integer ID of District
   - dis_name_e - String name of District in English
   - commune_id - String/Integer ID of Commune

- name_eng - String name of Commune in English
- geometry - Shapely geometry of boundary Polygon

Results

1. **Excel sheet of network-hazard-boundary intersection with attributes:**
    - edge_id/node_id - String name of intersecting edge ID or node ID
    - length - Float length of intersection of edge LineString and hazard Polygon: Only for edges
    - province_id - String/Integer ID of Province
    - province_name - String name of Province in English
    - district_id - String/Integer ID of District
    - district_name - String name of District in English
    - commune_id - String/Integer ID of Commune
    - commune_name - String name of Commune in English

hazard_data_summary(*hazard_network_dataframe*, *network_dataframe*)

main()

Summarise

1. **Specify the paths from where you to read and write:**
    - Input data
    - Intermediate calcuations data
    - Output results
2. **Supply input data and parameters**
    - Names of the three Provinces - List of string types
    - Names of modes - List of strings
    - Names of output modes - List of strings
    - Names of hazard bands - List of integers
    - Names of hazard thresholds - List of integers
    - Condition 'Yes' or 'No' is the users wants to process results
3. **Give the paths to the input data files:**
    - Commune boundary and stats data shapefile
    - Hazard datasets description Excel file
    - String name of sheet in hazard datasets description Excel file
4. Specify the output files and paths to be created

vtra.stats.network_stats_calculations module

Summarise hazard data

Get OD data and process it

main()

vtra.stats.province_roads_criticality_risk_stats module

Provincial road network risks and adaptation maps

main()

1.5.2 Submodules

1.5.3 vtra.transport_flow_and_failure_functions module

Functions used in the provincial and national-scale network failure analysis

add_igraph_generalised_costs(*G*, *vehicle_numbers*, *tonnage*)

edge_failure_sampling(*failure_scenarios*, *edge_column*)

Criteria for selecting failure samples

Parameters

- **– Pandas DataFrame of failure scenarios** (*failure_scenarios*) –
- **– String name of column to select failed edge ID's** (*edge_column*) –

Returns

Return type edge_failure_samples - List of lists of failed edge sets

identify_all_failure_paths(*network_df_in*, *edge_failure_set*, *flow_dataframe*, *path_criteria*)

Identify all paths that contain an edge

Parameters

- **– Pandas DataFrame of network** (*network_df*) –
- **– List of string edge ID's** (*edge_failure_set*) –
- **– Pandas DataFrame of list of edge paths** (*flow_dataframe*) –
- **– String name of column of edge paths in flow dataframe** (*path_criteria*) –
- **Outputs** –
- **-------** –
- **– Pandas DataFrame of network** – With removed edges
- **– List of integer indexes** (*edge_path_index*) – Of locations of paths in flow dataframe

igraph_scenario_edge_failures(*network_df_in*, *edge_failure_set*, *flow_dataframe*, *vehicle_weight*, *path_criteria*, *tons_criteria*, *cost_criteria*, *time_criteria*)

Estimate network impacts of each failures When the tariff costs of each path are fixed by vehicle weight

Parameters

- **– Pandas DataFrame of network** (*network_df_in*) –
- **– List of string edge ID's** (*edge_failure_set*) –
- **– Pandas DataFrame of list of edge paths** (*flow_dataframe*) –
- **– Float weight of vehcile weight** (*vehicle_weight*) –
- **– String name of column of edge paths in flow dataframe** (*path_criteria*) –
- **– String name of column of path tons in flow dataframe** (*tons_criteria*) –

- **– String name of column of path costs in flow dataframe** (*cost_criteria*) –
- **– String name of column of path travel time in flow dataframe** (*time_criteria*) –

Returns **edge_failure_dictionary** – With attributes edge_id - String name or list of failed edges origin - String node ID of Origin of disrupted OD flow destination - String node ID of Destination of disrupted OD flow no_access - Boolean 1 (no reroutng) or 0 (rerouting) new_cost - Float value of estimated cost of OD journey after disruption new_distance - Float value of estimated distance of OD journey after disruption new_path - List of string edge ID's of estimated new route of OD journey after disruption new_time - Float value of estimated time of OD journey after disruption

Return type list[94][dict[95]]

igraph_scenario_edge_failures_changing_tonnages(*network_df_in*, *edge_failure_set*, *flow_dataframe*, *vehicle_weight*, *path_criteria*, *tons_criteria*, *cost_criteria*, *time_criteria*)

Estimate network impacts of each failures When the tariff costs of each path depends on the changing tonnages

Parameters

- **– Pandas DataFrame of network** (*network_df_in*) –
- **– List of string edge ID's** (*edge_failure_set*) –
- **– Pandas DataFrame of list of edge paths** (*flow_dataframe*) –
- **– Float weight of vehcile weight** (*vehicle_weight*) –
- **– String name of column of edge paths in flow dataframe** (*path_criteria*) –
- **– String name of column of path tons in flow dataframe** (*tons_criteria*) –
- **– String name of column of path costs in flow dataframe** (*cost_criteria*) –
- **– String name of column of path travel time in flow dataframe** (*time_criteria*) –

Returns **edge_failure_dictionary** – With attributes edge_id - String name or list of failed edges origin - String node ID of Origin of disrupted OD flow destination - String node ID of Destination of disrupted OD flow no_access - Boolean 1 (no reroutng) or 0 (rerouting) new_cost - Float value of estimated cost of OD journey after disruption new_distance - Float value of estimated distance of OD journey after disruption new_path - List of string edge ID's of estimated new route of OD journey after disruption new_time - Float value of estimated time of OD journey after disruption

Return type list[96][dict[97]]

merge_failure_results(*flow_df_select*, *failure_df*, *tons_col*, *dist_col*, *time_col*, *cost_col*, *vehicle_col*, *changing_tonnages=True*)

Merge failure results with flow results

Parameters

- **flow_df_select** (*pandas.DataFrame*[98]) – edge flow values

[94] https://docs.python.org/3.6/library/stdtypes.html#list
[95] https://docs.python.org/3.6/library/stdtypes.html#dict
[96] https://docs.python.org/3.6/library/stdtypes.html#list
[97] https://docs.python.org/3.6/library/stdtypes.html#dict
[98] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame

- **failure_df** (*pandas.DataFrame*[99]) – edge failure values
- **tons_col** (*str*[100]) – name of column of tonnages in flow dataframe
- **dist_col** (*str*[101]) – name of column of distance in flow dataframe
- **time_col** (*str*[102]) – name of column of time in flow dataframe
- **cost_col** (*str*[103]) – name of column of cost in flow dataframe
- **vehicle_col** (*str*[104]) – name of column of vehicle counts in flow dataframe
- **changing_tonnages** (*bool*[105]) –

Returns **flow_df_select** – Of edge flow and failure values merged

Return type pandas.DataFrame[106]

network_failure_assembly_shapefiles(*edge_failure_dataframe*, *gdf_edges*, *save_edges=True*, *shape_output_path=''*)

Write results to Shapefiles

Outputs gdf_edges - a Shapefile with results of edge failure dataframe

Parameters

- **edge_failure_dataframe** – Pandas DataFrame of edge failure results
- **gdf_edges** – GeoDataFrame of network edge set with edge ID's and geometry
- **save_edges** (*bool*[107]) – Boolean condition to tell code to save created edge shapefile
- **shape_output_path** (*str*[108]) – Path where the output shapefile will be stored

network_od_path_estimations(*graph*, *source*, *target*, *tonnage*, *vehicle_weight*, *cost_criteria*, *time_criteria*)

Estimate the paths, distances, times, and costs for given OD pair

Parameters

- **graph** – igraph network structure
- **source** – String/Float/Integer name of Origin node ID
- **source** – String/Float/Integer name of Destination node ID
- **tonnage** (*float*[109]) – value of tonnage
- **vehicle_weight** (*float*[110]) – unit weight of vehicle
- **cost_criteria** (*str*[111]) – name of generalised cost criteria to be used: min_gcost or max_gcost
- **time_criteria** (*str*[112]) – name of time criteria to be used: min_time or max_time
- **fixed_cost** (*bool*[113]) –

Returns

[99] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame
[100] https://docs.python.org/3.6/library/stdtypes.html#str
[101] https://docs.python.org/3.6/library/stdtypes.html#str
[102] https://docs.python.org/3.6/library/stdtypes.html#str
[103] https://docs.python.org/3.6/library/stdtypes.html#str
[104] https://docs.python.org/3.6/library/stdtypes.html#str
[105] https://docs.python.org/3.6/library/functions.html#bool
[106] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame
[107] https://docs.python.org/3.6/library/functions.html#bool
[108] https://docs.python.org/3.6/library/stdtypes.html#str
[109] https://docs.python.org/3.6/library/functions.html#float
[110] https://docs.python.org/3.6/library/functions.html#float
[111] https://docs.python.org/3.6/library/stdtypes.html#str
[112] https://docs.python.org/3.6/library/stdtypes.html#str
[113] https://docs.python.org/3.6/library/functions.html#bool

- **edge_path_list** (*list[list]*) – nested lists of Strings/Floats/Integers of edge ID's in routes
- **path_dist_list** (*list[float]*) – estimated distances of routes
- **path_time_list** (*list[float]*) – estimated times of routes
- **path_gcost_list** (*list[float]*) – estimated generalised costs of routes

rearrange_minmax_values(*edge_failure_dataframe*)

Write results to Shapefiles

Parameters **edge_failure_dataframe** (*pandas.DataFrame*[114]) – with min-max columns

Returns **edge_failure_dataframe** – With columns where min < max

Return type pandas.DataFrame[115]

spatial_scenario_selection(*network_shapefile*, *polygon_shapefile*, *hazard_dictionary*, *data_dictionary*, *network_type='nodes'*, *name_province=''*)

Intersect network edges/nodes and boundary Polygons to collect boundary and hazard attributes

Parameters

- network_shapefile - Shapefile of edge LineStrings or node Points
- polygon_shapefile - Shapefile of boundary Polygons
- hazard_dictionary - Dictionary of hazard attributes
- data_dictionary - Dictionary of network-hazard-boundary intersection attributes
- network_type - String value -'edges' or 'nodes' - Default = 'nodes'
- name_province - String name of province if needed - Default = ''

Outputs

data_dictionary - Dictionary of network-hazard-boundary intersection attributes:

- edge_id/node_id - String name of intersecting edge ID or node ID
- length - Float length of intersection of edge LineString and hazard Polygon: Only for edges
- province_id - String/Integer ID of Province
- province_name - String name of Province in English
- district_id - String/Integer ID of District
- district_name - String name of District in English
- commune_id - String/Integer ID of Commune
- commune_name - String name of Commune in English
- hazard_attributes - Dictionary of all attributes from hazard dictionary

swap_min_max(*x*, *min_col*, *max_col*)

Swap columns if necessary

write_flow_paths_to_network_files(*save_paths_df*, *industry_columns*, *min_max_exist*, *gdf_edges*, *save_csv=True*, *save_shapes=True*, *shape_output_path=''*, *csv_output_path=''*)

Write results to Shapefiles

Outputs `gdf_edges` - a shapefile with minimum and maximum tonnage flows of all commodities/industries for each edge of network.

Parameters

- **save_paths_df** – Pandas DataFrame of OD flow paths and their tonnages

[114] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame
[115] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame

- **industry_columns** – List of string names of all OD commodities/industries indentified
- **min_max_exist** – List of string names of commodity/industry columns for which min-max tonnage column names already exist
- **gdf_edges** – GeoDataFrame of network edge set
- **save_csv** – Boolean condition to tell code to save created edge csv file
- **save_shapes** – Boolean condition to tell code to save created edge shapefile
- **shape_output_path** – Path where the output shapefile will be stored
- **csv_output_path** – Path where the output csv file will be stored

1.5.4 vtra.utils module

Shared plotting functions

class Style

Bases: tuple[116]

Style(color, zindex, label): class to hold an element's styles

Used to generate legend entries, apply uniform style to groups of map elements (See network_map.py for example.)

color
Alias for field number 0

label
Alias for field number 2

zindex
Alias for field number 1

assign_value_in_area_proportions(*poly_1_gpd*, *poly_2_gpd*, *poly_attribute*)

assign_value_in_area_proportions_within_common_region(*poly_1_gpd*, *poly_2_gpd*, *poly_attribute*, *common_region_id*)

count_points_in_polygon(*x*, *points_sindex*)

Count points in a polygon

Parameters

- **x** – row of dataframe
- **points_sindex** – spatial index of dataframe with points in the region to consider

Returns

Return type Number of points in polygon

extract_gdf_values_containing_nodes(*x*, *sindex_input_gdf*, *input_gdf*, *column_name*)

extract_nodes_within_gdf(*x*, *input_nodes*, *column_name*)

extract_value_from_gdf(*x*, *gdf_sindex*, *gdf*, *column_name*)

Access value

Parameters

- **x** – row of dataframe
- **gdf_sindex** – spatial index of dataframe of which we want to extract the value
- **gdf** – GeoDataFrame of which we want to extract the value

[116] https://docs.python.org/3.6/library/stdtypes.html#tuple

- **column_name** – column that contains the value we want to extract

Returns

Return type extracted value from other gdf

gdf_clip (*shape_in*, *clip_geom*)
Filter a file to contain only features within a clipping geometry

Parameters

- **shape_in** – path string to shapefile to be clipped
- **province_geom** – shapely geometry of province for what we do the calculation

Returns

Return type filtered dataframe

gdf_geom_clip (*gdf_in*, *clip_geom*)
Filter a dataframe to contain only features within a clipping geometry

Parameters

- **gdf_in** – geopandas dataframe to be clipped in
- **province_geom** – shapely geometry of province for what we do the calculation

Returns

Return type filtered dataframe

generate_weight_bins (*weights*, *n_steps=9*, *width_step=0.01*)
Given a list of weight values, generate <n_steps> bins with a width value to use for plotting e.g. weighted network flow maps.

generate_weight_bins_with_colour_gradient (*weights, n_steps=9, width_step=0.01, colours=['orange', 'red']*)
Given a list of weight values, generate <n_steps> bins with a width value to use for plotting e.g. weighted network flow maps.

get_axes (*extent=None*, *figsize=None*, *epsg=None*)
Get transverse mercator axes (default to Vietnam extent) EPSG:4756

get_data (*filename*)
Read in data (as array) and extent of each raster

get_district_label (*record*)

get_nearest_node (*x*, *sindex_input_nodes*, *input_nodes*, *id_column*)
Get nearest node in a dataframe

Parameters

- **x** – row of dataframe
- **sindex_nodes** – spatial index of dataframe of nodes in the network
- **nodes** – dataframe of nodes in the network
- **id_column** – name of column of id of closest node

Returns

Return type Nearest node to geometry of row

get_nearest_node_within_region (*x*, *input_nodes*, *id_column*, *region_id*)

get_node_edge_files (*mode_file_path*, *file_identification*)
Get the paths of edge and node files in folder

Parameters

- **mode_file_path** (*str*[117]) – path of mode file
- **file_identification** (*str*[118]) – name of file

Returns

- *edges_in* – Path of edges shapefile
- *nodes_in* – Path of nodes shapefile
- *Error Exception*
- ————
- *Prints error if node or edge file missing*

get_node_edge_files_in_path(*mode_file_path*)

Get the paths of edge and node files in folder

Parameters **mode_file_path** (*Path of mode file*) –

Returns

- **edges_in** (*Path of edges shapefile*)
- **nodes_in** (*Path of nodes shapefile*)
- *Error Exception*
- ————
- *Prints error if node or edge file missing*

get_region_plot_settings(*region*)

Common definition of region plot settings

legend_from_style_spec(*ax*, *styles*, *loc='lower left'*)

Plot legend

line_length(*line*, *ellipsoid='WGS-84'*)

Length of a line in meters, given in geographic coordinates.

Adapted from https://gis.stackexchange.com/questions/4022/looking-for-a-pythonic-way-to-calculate-the-length-of-a-wkt-li answer-115285

Parameters

- **line** – a shapely LineString object with WGS-84 coordinates.
- **ellipsoid** – string name of an ellipsoid that *geopy* understands (see http://geopy.readthedocs.io/en/latest/#module-geopy.distance).

Returns Length of line in kilometers.

load_config()

Read config.json

plot_basemap(*ax*, *data_path*, *focus='VNM'*, *neighbours=None*, *country_border='white'*, *plot_regions=True*, *plot_states=True*, *plot_districts=False*, *highlight_region=None*)

Plot countries and regions background

plot_basemap_labels(*ax*, *data_path*, *labels=None*, *province_zoom=False*, *plot_regions=True*, *plot_international_left=True*, *plot_international_right=True*)

Plot countries and regions background

plot_basemap_labels_large_region(*ax*, *data_path*)

plot_district_labels(*ax*, *data_path*, *highlight_region=None*)

round_sf(*x*, *places=1*)

Round number to significant figures

[117] https://docs.python.org/3.6/library/stdtypes.html#str
[118] https://docs.python.org/3.6/library/stdtypes.html#str

save_fig (*output_filename*)

scale_bar (*ax*, *length=100*, *location=(0.5, 0.05)*, *linewidth=3*)

Draw a scale bar

Adapted from https://stackoverflow.com/questions/32333870/how-can-i-show-a-km-ruler-on-a-cartopy-matplotlib-plot/35705477#35705477

Parameters

- **ax** (*axes*) –
- **length** (*int*[119]) – length of the scalebar in km.
- **location** (*tuple*[120]) – center of the scalebar in axis coordinates (ie. 0.5 is the middle of the plot)
- **linewidth** (*float*[121]) – thickness of the scalebar.

set_ax_bg (*ax*, *color='#c6e0ff'*)

Set axis background color

voronoi_finite_polygons_2d (*vor*, *radius=None*)

Reconstruct infinite voronoi regions in a 2D diagram to finite regions.

Source: https://stackoverflow.com/questions/36063533/clipping-a-voronoi-diagram-python

Parameters

- **vor** (*Voronoi*) – Input diagram
- **radius** (*float*[122], *optional*) – Distance to 'points at infinity'

Returns

- **regions** (*list of tuples*) – Indices of vertices in each revised Voronoi regions.
- **vertices** (*list of tuples*) – Coordinates for revised Voronoi vertices. Same as coordinates of input vertices, with 'points at infinity' appended to the end

within_extent (*x*, *y*, *extent*)

1.6 MIT License

```
MIT License

Copyright (c) 2018 oi-analytics

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
```

(continues on next page)

[119] https://docs.python.org/3.6/library/functions.html#int
[120] https://docs.python.org/3.6/library/stdtypes.html#tuple
[121] https://docs.python.org/3.6/library/functions.html#float
[122] https://docs.python.org/3.6/library/functions.html#float

(continued from previous page)

```
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.
```

1.7 Developers

- Elco Koks
- Raghav Pant
- Tom Russell
- Roald Schoenmakers

CHAPTER 2

Indexes and tables

- genindex
- modindex
- search

CHAPTER 3

Acknowledgements

This project has been developed by Oxford Infrastructure Analytics as part of a project funded by the World Bank.

[123] https://github.com/oi-analytics/vietnam-transport

Python Module Index

a

f

m

p

s

t

u

v

Index

A

I

J

L

M

N

P

R

S

T

V

W

Z