gmtra Documentation

Release 0.1

Elco Koks

Aug 22, 2019

Contents:

This is the documentation of the code to perform a global transport asset risk analysis for earthquakes, floods and cyclones.

CHAPTER 1

Data requirements

- All transport data is based on OpenStreetMap (OSM), which can be freely downloaded. The planet file used in Koks et al. (in review) is downloaded at July 17, 2018. However, to run the code, the latest planet.osm.pbf file can be used.
- Global earthquake and cyclone hazard data is available from the UNISDR Global Assessment Report 2015 data portal (https://risk.preventionweb.net).
- In Koks et al. (in review), global fluvial and surface SSBN flood hazard data (May 2017 version) is used with the permission of Fathom Global (http://www.fathom.global/).
- The coastal flood maps are developed by the Joint Research Centre of the European Commission.

CHAPTER 2

Prepare data paths

Copy *config.template.json* to *config.json* and edit the paths for data and figures, for example:

```
{
        "data_path": "/home/<user>/projects/GMTRA/data",
        "hazard_path": "/home/<user>/projects/GMTRA/hazard_data",
        "figure_path": "/home/<user>/projects/GMTRA/figures"
}
```

CHAPTER 3

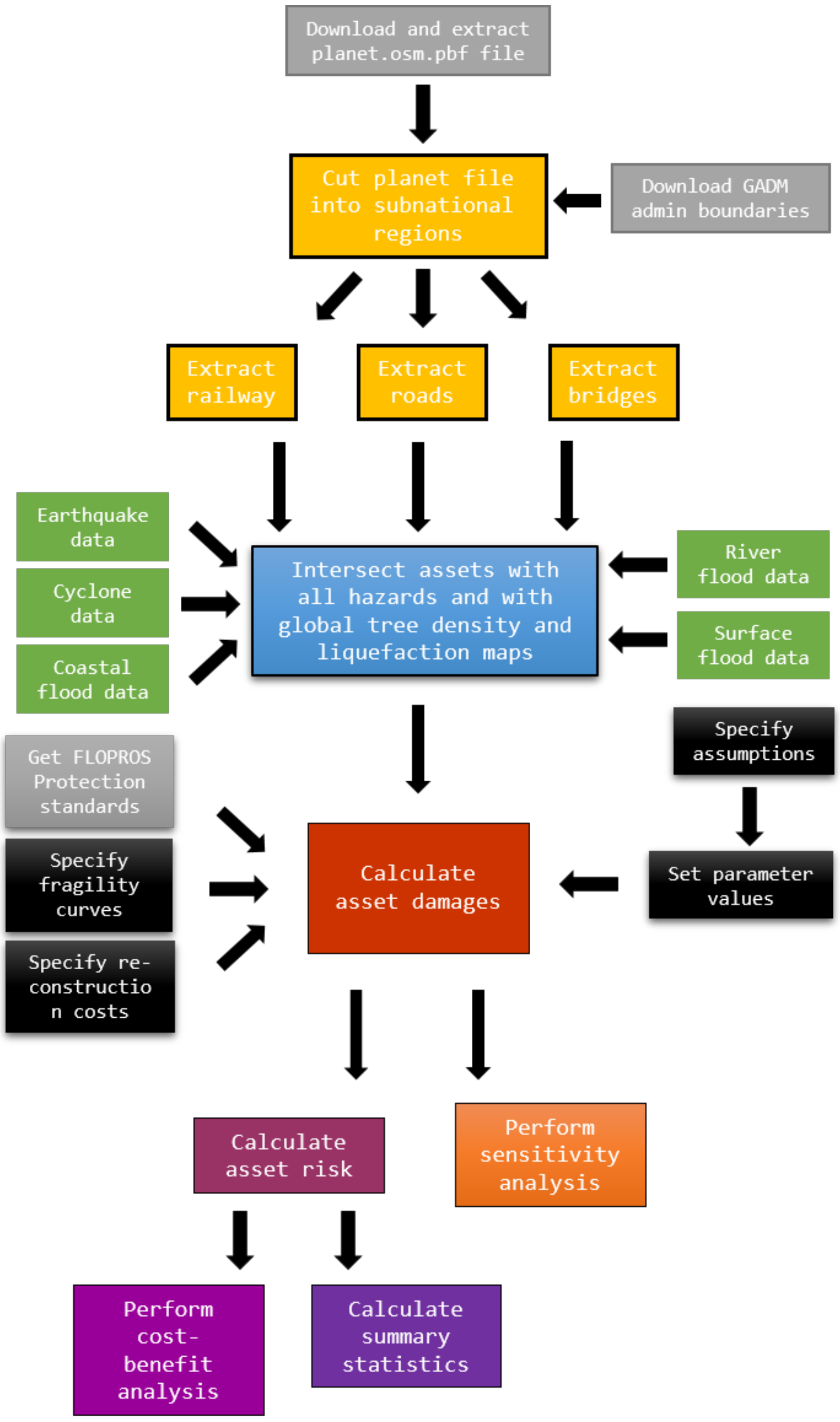
Python requirements

Recommended option is to use a [miniconda](https://conda.io/miniconda.html) environment to work in for this project, relying on conda to handle some of the trickier library dependencies.

```
# Add conda-forge channel for extra packages
conda config --add channels conda-forge

# Create a conda environment for the project and install packages
conda env create -f environment.yml
activate GMTRA
```

3.1 1. GMTRA workflow


Download and extract planet.osm.pbf file
Cut planet file into subnational regions
Download GADM admin boundaries
Extract railway
Extract roads
Extract bridges
Earthquake data
Cyclone data
Coastal flood data
Intersect assets with all hazards and with global tree density and liquefaction maps
River flood data
Surface flood data
Get FLOPROS Protection standards
Specify fragility curves
Specify re-constructio n costs
Calculate asset damages
Specify assumptions
Set parameter values
Calculate asset risk
Perform sensitivity analysis
Perform cost-benefit analysis
Calculate summary statistics

3.2 2. OSM and Python

To be able to extract the data from OpenStreetMap (OSM), a few steps and downloads are required.

3.2.1 Add attributes to osmconf.ini

We need to add a few attributes to the osmconf.ini file to be able to extract everything we want.

1. Find the location of the osmconf.ini of the conda enviroment you are working in. It is generally located here:

Windows:

```
%USERPROFILE%\AppData\Local\Continuum\miniconda3\envs\py36\Library\share\gdal
```

Linux:

```
/home/<user>/.conda/envs/py36/share/gdal
```

2. On line 48 (the attributes of **[lines]**), add **railway, bridge, service**, when they are not there yet.

Note: make sure they are not in the list twice! This will cause problems.

3.2.2 Download osmconvert

Osmconvert is a small tool that can be used to clip OSM data.

1. Download **osmconvert64** from http://wiki.openstreetmap.org/wiki/Osmconvert.
2. Create a new directory in your working directory (where all the data is stored as well), called **osmconvert**
3. Move to the tool into this directory.

3.3 3. Utils

`gmtra.utils.`**`load_config`**`()`

Read config.json

`gmtra.utils.`**`clean_fluvial_dirs`**`(`*hazard_path*`)`

Remove all the data we do not use.

Arguments: *hazard_path* : file path to location of all hazard data.

`gmtra.utils.`**`load_osm_data`**`(`*data_path*, *country*`)`

Load osm data for an entire country.

Arguments: *data_path* : file path to location of all data.

country : unique ID of the country for which we want to extract data from OpenStreetMap. Must be matching with the country ID used for saving the .osm.pbf file.

`gmtra.utils.`**`load_osm_data_region`**`(`*data_path*, *region*`)`

Load osm data for a specific region.

Arguments: *data_path* : file path to location of all data.

region : unique ID of the region for which we want to extract data from OpenStreetMap. Must be matching with the region ID used for saving the .osm.pbf file.

gmtra.utils.**load_hazard_map**(*hzd_path*)
Load specific hazard map.

Arguments: *hzd_path* : file path to location of the hazard map.

gmtra.utils.**load_ssbn_hazard**(*hazard_path*, *country_full*, *country_ISO2*, *flood_type*, *flood_type_abb*, *rp*)
Function to load a SSBN hazard map.

Arguments: *hazard_path* : Path to location of all hazard data.

country_full : Full name of country. Obtained from **create_folder_lookup**.

country_ISO2 : ISO2 country code of the country.

flood_type : Specifies whether it is a **pluvial** or **fluvial** flood.

flood_type_abb : Abbrevated code of the flood type. **FU** for river flooding, **PU** for surface flooding.

rp : Return period of the flood map we want to extract.

Returns: *array*: NumPy Array with the raster values.

affine : Affine of the **array**.

gmtra.utils.**gdf_clip**(*gdf*, *clip_geom*)
Function to clip a GeoDataFrame with a shapely geometry.

Arguments: *gdf* : geopandas GeoDataFrame that we want to clip.

clip_geom : shapely geometry of region for which we do the calculation.

Returns: *gdf* : clipped geopandas GeoDataframe

gmtra.utils.**sum_tuples**(*l*)
Function to sum a list of tuples.

Arguments: *l* : list of tuples.

Returns: *tuple* : a tuple with the sum of the list of tuples.

gmtra.utils.**set_prot_standard**(*x*, *prot_lookup*, *events*)
Function to set all values to zero below protection standard.

Arguments: *x* : row in a GeoDataFrame that represents an unique infrastructure asset.

prot_lookup : dictionary with protection standards for each region.

events : A list with the unique hazard events in row **x**.

Returns: *x* : row in a GeoDataFrame that represents an unique infrastructure asset with zero values for no flooding.

gmtra.utils.**sensitivity_risk**(*RPS*, *loss_list*, *events*, *param_length*)
Function to estimate the monetary risk for a particular hazard within the sensitivity analysis.

Arguments: *RPS* : list of return periods in floating probabilities (i.e. [1/10,1/20,1/50]).

loss_list : list of lists with a monetary value per return period within each inner list.

Returns: *collect_risks* : a list of all risks for each inner list of the input list.

gmtra.utils.**monetary_risk**(*RPS*, *x*, *events*)
Function to estimate the monetary risk for a particular hazard.

Arguments: *RPS* : list of return periods in floating probabilities (i.e. [1/10,1/20,1/50]).

x : list of lists with a monetary value per return period within each inner list.

events : list of events that correspond with the return periods in the inner lists of **x**.

Returns: *collect_risks* : a list of all risks for each inner list of the input list.

`gmtra.utils.`**`exposed_length_risk`**`(`*x*, *hzd*, *RPS*`)`

Function to estimate risk in terms of exposed kilometers.

Arguments: *x* : row in a GeoDataFrame that represents an unique infrastructure asset.

hzd : abbrevation of the hazard we want to intersect. **EQ** for earthquakes, **Cyc** for cyclones, **FU** for river flooding, **PU** for surface flooding and **CF** for coastal flooding.

RPS : list of return periods in floating probabilities (i.e. [1/10,1/20,1/50]). Should match with the hazard we are considering.

Returns: *risk value* : a floating number which represents the annual exposed kilometers of infrastructure.

`gmtra.utils.`**`total_length_risk`**`(`*x*, *RPS*`)`

Function to estimate risk if all assets would have been exposed.

Arguments: *x* : row in a GeoDataFrame that represents an unique infrastructure asset.

RPS : list of return periods in floating probabilities (i.e. [1/10,1/20,1/50]). Should match with the hazard we are considering.

`gmtra.utils.`**`square_m2_cost_range`**`(`*x*`)`

Function to specify the range of possible costs for a bridge.

Arguments: *x* : row in a GeoDataFrame that represents an unique bridge asset.

Returns: *list*: a list with the range of possible bridge costs.

`gmtra.utils.`**`extract_value_from_gdf`**`(`*x*, *gdf_sindex*, *gdf*, *column_name*`)`

Arguments: *x* : row in a geopandas GeoDataFrame. *gdf_sindex* : spatial index of dataframe of which we want to extract the value.

gdf : GeoDataFrame of which we want to extract the value.

column_name : column that contains the value we want to extract.

Returns: extracted value from other GeoDataFrame

`gmtra.utils.`**`create_folder_lookup`**`()`

Function to create a dictionary in which we can lookup the folder path where the surface and river flood maps are located for a country.

`gmtra.utils.`**`map_roads`**`()`

Mapping function to create a dictionary with an aggregated list of road types.

Returns: *dictionary* : A dictionary with road types and their aggregated equivalent

`gmtra.utils.`**`map_railway`**`()`

Mapping function to create a dictionary with an aggregated list of railway types.

Returns: *dictionary* : A dictionary with road types and their aggregated equivalent

`gmtra.utils.`**`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-linest answer-115285

Arguments: *line* : a shapely LineString object with WGS-84 coordinates

Optional Arguments: *ellipsoid* : string name of an ellipsoid that *geopy* understands (see http://geopy.readthedocs.io/en/latest/#module-geopy.distance)

Returns: Length of line in meters

3.4 4. Preprocessing functions

3.4.1 Download planet OSM

`gmtra.preprocessing.`**`planet_osm`**`()`

This function will download the planet file from the OSM servers.

3.4.2 Create osm.pbf files for a single country

`gmtra.preprocessing.`**`single_country`**`(`*country*, *regionalized=False*, *create_poly_files=False*`)`

Clip a country from the planet osm file and save to individual osm.pbf files

This function has the option to extract individual regions

Arguments: *country* : The country for which we want extract the data.

Keyword Arguments: *regionalized* : Default is **False**. Set to **True** will parallelize the extraction over all regions within a country.

create_poly_files : Default is **False**. Set to **True** will create new .poly files.

3.4.3 Create osm.pbf files for all specified countries

`gmtra.preprocessing.`**`all_countries`**`(`*subset=[]*, *regionalized=False*, *reversed_order=False*`)`

Clip all countries from the planet osm file and save them to individual osm.pbf files

Optional Arguments: *subset* : allow for a pre-defined subset of countries. REquires ISO3 codes. Will run all countries if left empty.

regionalized : Default is **False**. Set to **True** if you want to have the regions of a country as well.

reversed_order : Default is **False**. Set to **True** to work backwards for a second process of the same country set to prevent overlapping calculations.

Returns: clipped osm.pbf files for the defined set of countries (either the whole world by default or the specified subset)

3.4.4 Create global shapefiles

`gmtra.preprocessing.`**`global_shapefiles`**`(`*regionalized=False*`)`

This function will simplify shapes and add necessary columns, to make further processing more quickly

For now, we will make use of the latest GADM data: https://gadm.org/download_world.html

Optional Arguments: *regionalized* : Default is **False**. Set to **True** will also create the global_regions.shp file.

3.4.5 Remove tiny shapes from large multipolygons

`gmtra.preprocessing.`**`remove_tiny_shapes`**`(`*x*, *regionalized=False*`)`

This function will remove the small shapes of multipolygons. Will reduce the size of the file.

Arguments: *x* : a geometry feature (Polygon) to simplify. Countries which are very large will see larger (unhabitated) islands being removed.

Optional Arguments: *regionalized* : Default is **False**. Set to **True** will use lower threshold settings (default: **False**).

Returns: *MultiPolygon* : a shapely geometry MultiPolygon without tiny shapes.

3.4.6 Create .poly files

`gmtra.preprocessing.`**`poly_files`**(*data_path*, *global_shape*, *save_shapefile=False*, *regionalized=False*)

This function will create the .poly files from the world shapefile. These .poly files are used to extract data from the openstreetmap files.

This function is adapted from the OSMPoly function in QGIS.

Arguments: *data_path* : base path to location of all files.

global_shape: exact path to the global shapefile used to create the poly files.

Optional Arguments: *save_shape_file* : Default is **False**. Set to **True** will the new shapefile with the countries that we include in this analysis will be saved.

regionalized : Default is **False**. Set to **True** will perform the analysis on a regional level.

Returns: *.poly file* for each country in a new dir in the working directory.

3.4.7 Clip a region from a larger .osm.pbf file

`gmtra.preprocessing.`**`clip_osm`**(*data_path*, *planet_path*, *area_poly*, *area_pbf*)

Clip the an area osm file from the larger continent (or planet) file and save to a new osm.pbf file. This is much faster compared to clipping the osm.pbf file while extracting through ogr2ogr.

This function uses the osmconvert tool, which can be found at http://wiki.openstreetmap.org/wiki/Osmconvert.

Either add the directory where this executable is located to your environmental variables or just put it in the 'scripts' directory.

Arguments: *continent_osm*: path string to the osm.pbf file of the continent associated with the country.

area_poly: path string to the .poly file, made through the 'create_poly_files' function.

area_pbf: path string indicating the final output dir and output name of the new .osm.pbf file.

Returns: a clipped .osm.pbf file.

3.4.8 Merge SSBN maps within a country

`gmtra.preprocessing.`**`merge_SSBN_maps`**(*country*)

Function to merge SSBN maps to a country level.

Arguments: *country* : ISO3 code of the country for which we want to merge the river and surface flood maps to country level.

3.4.9 Extract bridges from OpenStreetMap

`gmtra.preprocessing.`**`region_bridges`**`(`*n*`)`

This function will extract all bridges from OpenStreetMap for the specified region.

Arguments: *n* : the index ID of a region in the specified shapefile with all the regions.

Returns: *GeoDataFrame* : A geopandas GeoDataFrame with all bridges in a region. Will also save this to a .csv file.

3.5 5. Fetch asset data

3.5.1 Fetch roads

`gmtra.fetch.`**`roads`**`(`*data_path*, *area_name*, *regional=False*`)`

Function to extract all road assets from an .osm.pbf file.

Arguments: *data_path* : file path to location of all data.

area_name: Admin code of the countryor region for which we want to extract the roads.

Optional Arguments: *regional* : Set to True if we want to extract a region.

3.5.2 Fetch railway

`gmtra.fetch.`**`railway`**`(`*data_path*, *country*, *regional=True*`)`

Function to extract all railway assets from an .osm.pbf file.

Arguments: *data_path* : file path to location of all data.

area_name: Admin code of the countryor region for which we want to extract the roads.

Optional Arguments: *regional* : Set to True if we want to extract a region.

3.5.3 Fetch bridges

`gmtra.fetch.`**`bridges`**`(`*data_path*, *area_name*, *regional=True*`)`

Function to extract all bridges from an .osm.pbf file.

Arguments: *data_path* : file path to location of all data.

area_name: Admin code of the countryor region for which we want to extract the roads.

Optional Arguments: *regional* : Set to True if we want to extract a region.

3.6 6. Hazard extraction

3.6.1 Polygonize a single hazard file

`gmtra.hazard.`**`single_polygonized`**`(`*flood_scen*, *region*, *geometry*, *country_ISO3*, *hzd='FU'*`)`

Function to overlay a surface or river flood hazard map with the infrastructure assets.

Arguments: *flood_scen* : Unique ID for the flood scenario to be used.

region : Unique ID of the region that is intersected.

geometry : Shapely geometry of the region that is being intersected.

country_ISO3 : ISO3 code of the country in which the region is situated. Required to get the FATHOM flood maps.

Optional Arguments: *hzd* : Default is **FU**. Can be changed to **PU** for surface flooding.

Returns: *gdf* : A GeoDataFrame where each row is a poylgon with the same flood depth.

3.6.2 Polygonize multiple hazard files

`gmtra.hazard.`**`multiple_polygonized`**(*region*, *geometry*, *hzd_list*, *hzd_names*)

Function to overlay a set of hazard maps with infrastructure assets. This function is used for Earthquakes, Coastal floods and Cyclones (the hazard maps with global coverage).

Arguments: *region* : Unique ID of the region that is intersected.

geometry : Shapely geometry of the region that is being intersected.

hzd_list : list of file paths to each hazard. Make sure *hzd_list* and *hzd_names* are matching.

hzd_names : llist of unique hazard IDs. Most often these are the return periods.

Returns: *gdf* : A GeoDataFrame where each row is a poylgon with the same hazard value.

3.6.3 Intersect an infrastructure asset with hazard maps

`gmtra.hazard.`**`intersect_hazard`**(*x*, *hzd_reg_sindex*, *hzd_region*, *liquefaction=False*)

Function to intersect an infrastructure asset (within a GeoDataFrame) with the hazard maps.

Arguments: *x* : row in a GeoDataFrame that represents an unique infrastructure asset.

hzd_reg_sindex : Spatial Index of the hazard GeoDataFrame.

hzd_region : GeoDataFrame of a unique hazard map.

Optional arguments: *liquefaction* : Default is **False**. Set to **True** if you are intersecting with the liquefaction map.

Returns: *tuple* : a shapely.geometry of the part of the asset that is affected and the average hazard value in this intersection.

3.6.4 Intersect all assets in a region with a hazard

`gmtra.hazard.`**`region_intersection`**(*n*, *hzd*, *rail=False*)

Function to intersect all return periods of a particualar hazard with all road or railway assets in the specific region.

Arguments: *n* : the index ID of a region in the specified shapefile with all the regions.

hzd : abbrevation of the hazard we want to intersect. **EQ** for earthquakes, **Cyc** for cyclones, **FU** for river flooding, **PU** for surface flooding and **CF** for coastal flooding.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *output* : a GeoDataFrame with all intersections between the infrastructure assets and the specified hazard. Will also be saved as .feather file.

3.6.5 Intersect all assets in a region with a global liquefaction map

`gmtra.hazard.`**`get_liquefaction_region`**(*n*, *rail=False*)

Function to intersect all return periods of a particualar hazard with all road or railway assets in the specific region.

Arguments: *n* : the index ID of a region in the specified shapefile with all the regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *output* : a GeoDataFrame with all intersections between the infrastructure assets and the liquefaction map. Will be saved as .feather file.

3.6.6 Intersect all assets in a region with a global tree density map

`gmtra.hazard.`**`get_tree_density`**(*n*, *rail=False*)

Function to intersect all return periods of a particualar hazard with all road or railway assets in the specific region.

Arguments: *n* : the index ID of a region in the specified shapefile with all the regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *output* : a GeoDataFrame with all intersections between the infrastructure assets and the liquefaction map. Will be saved as .feather file.

3.7 7. Exposure analysis

3.7.1 Estimate exposure of all roads in a region

`gmtra.exposure.`**`regional_roads`**(*n*, *prot_lookup*, *data_path*)

Function to get summarized exposure values for each region for all road assets.

Arguments: *n* : the index ID of a region in the specified shapefile with all the regions.

prot_lookup : dictionary with dike design standards for a region.

data_path : file path to location of all data.

Returns: *dataframe* : a pandas DataFrame with exposure statistics.

3.7.2 Estimate exposure of all railway in a region

`gmtra.exposure.`**`regional_railway`**(*n*, *prot_lookup*, *data_path*)

Function to get summarized exposure values for each region for all railway assets.

Arguments: *n* : the index ID of a region in the specified shapefile with all the regions.

prot_lookup : dictionary with dike design standards for a region.

data_path : file path to location of all data.

Returns: *dataframe* : a pandas DataFrame with exposure statistics.

3.8 8. Damage assessment

3.8.1 Flood or cyclone damage to a unique road bridge

`gmtra.damage.`**`road_bridge_flood_cyclone`**(*x*, *design_table*, *depth_threshs*, *param_values*, *events*, *all_rps*, *sensitivity=False*)

Function to estimate the range of either flood or cyclone damages to an individual bridge asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

design_table : A NumPy array that represents the design standards for different bridge types, dependent on road type.

depth_thresh : A list with failure thresholds. Either contains flood depths or gust speeds.

param_values : A NumPy Array with sets of parameter values we would like to test.

events : A list with the unique hazard events in row **x**.

all_rps : A list with all return periods for the hazard that is being considered.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified bridge, based on the parameter set.

3.8.2 Flood or cyclone damage to a unique railway bridge

`gmtra.damage.`**`rail_bridge_flood_cyclone`**(*x*, *design_table*, *depth_threshs*, *param_values*, *events*, *all_rps*, *sensitivity=False*)

Function to estimate the range of either flood or cyclone damages to an individual bridge asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

design_table : A NumPy array that represents the design standards for different bridge types, dependent on road type.

depth_thresh : A list with failure thresholds. Either contains flood depths or gust speeds.

param_values : A NumPy Array with sets of parameter values we would like to test.

events : A list with the unique hazard events in row **x**.

all_rps : A list with all return periods for the hazard that is being considered.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified bridge, based on the parameter set.

3.8.3 Earthquake damage to a unique road bridge

`gmtra.damage.`**`road_bridge_earthquake`**(*x*, *eq_curve*, *param_values*, *events*, *all_rps*, *sensitivity=False*)

Function to estimate the range of earthquake damages to an individual bridge asset.

Arguments: *x* : A row in a geopandas GeoDataFrame that represents an individual asset.

eq_curve : A pandas DataFrame with unique damage curves for earthquake damages.

param_values : A NumPy Array with sets of parameter values we would like to test.

events : A list with the unique hazard events in row **x**.

all_rps : A list with all return periods for the hazard that is being considered.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified bridge, based on the parameter set.

3.8.4 Earthquake damage to a unique railway bridge

`gmtra.damage.`**`rail_bridge_earthquake`**(*x*, *eq_curve*, *param_values*, *vals_EQ*, *events*, *all_rps*, *sensitivity=False*)

Function to estimate the range of earthquake damages to an individual bridge asset.

Arguments: *x* : A row in a geopandas GeoDataFrame that represents an individual asset.

eq_curve : A pandas DataFrame with unique damage curves for earthquake damages.

param_values : A NumPy Array with sets of parameter values we would like to test.

vals_EQ : A list with the unique hazard events in row **x**.

all_rps : A list with all return periods for the hazard that is being considered.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified bridge, based on the parameter set.

3.8.5 Cyclone damage to a unique road asset

`gmtra.damage.`**`road_cyclone`**(*x*, *events*, *param_values*, *sensitivity=False*)

Function to estimate the range of cyclone damages to an individual road asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

events : A list with the unique hazard events in row **x**.

param_values : A NumPy Array with sets of parameter values we would like to test.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified asset, based on the parameter set.

3.8.6 Cyclone damage to a unique railway asset

`gmtra.damage.`**`rail_cyclone`**(*x*, *events*, *param_values*, *sensitivity=False*)

Function to estimate the range of cyclone damages to an individual railway asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

events : A list with the unique hazard events in row **x**.

param_values : A NumPy Array with sets of parameter values we would like to test.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified asset, based on the parameter set.

3.8.7 Earthquake damage to a unique road asset

`gmtra.damage.`**`road_earthquake`**(*x*, *global_costs*, *paved_ratios*, *frag_tables*, *events*, *wbreg_lookup*, *param_values*, *sensitivity=False*)

Function to estimate the range of earthquake damages to an individual road asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

global_costs : A pandas DataFrame with the total cost for different roads in different World Bank regions. These values are based on the ROCKS database.

paved_ratios : A pandas DataFrame with road pavement percentages per country for each road type.

frag_tables : A NumPy Array with a set of unique fragility tables which relate PGA to liquefaction to estimate the damage to the asset. *events* : A list with the unique earthquake events.

wbreg_lookup : a dictioniary that relates countries (in ISO3 codes) with World Bank regions.

param_values : A NumPy Array with sets of parameter values we would like to test.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified asset, based on the parameter set.

3.8.8 Earthquake damage to a unique railway asset

`gmtra.damage.`**`rail_earthquake`**(*x*, *frag_tables*, *events*, *param_values*, *sensitivity=False*)

Function to estimate the range of earthquake damages to an individual railway asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

frag_tables : A NumPy Array with a set of unique fragility tables which relate PGA to liquefaction to estimate the damage to the asset.

events : A list with the unique earthquake events.

param_values : A NumPy Array with sets of parameter values we would like to test.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified asset, based on the parameter set.

3.8.9 Flood damage to a unique road asset

`gmtra.damage.`**`road_flood`**(*x*, *global_costs*, *paved_ratios*, *flood_curve_paved*, *flood_curve_unpaved*, *events*, *wbreg_lookup*, *param_values*, *val_cols*, *sensitivity=False*)

Function to estimate the range of flood damages to an individual road asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

global_costs : A pandas DataFrame with the total cost for different roads in different World Bank regions. These values are based on the ROCKS database.

paved_ratios : A pandas DataFrame with road pavement percentages per country for each road type.

flood_curve_paved : A pandas DataFrame with a set of damage curves for paved roads.

flood_curve_unpaved : A pandas DataFrame with a set of damage curves for unpaved roads.

events : A list with the unique flood events.

wbreg_lookup : a dictioniary that relates a country ID (ISO3 code) with its World Bank region.

param_values : A NumPy Array with sets of parameter values we would like to test.

val_cols : A list with the unique flood events in row **x**.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified asset, based on the parameter set.

3.8.10 Flood damage to a unique railway asset

`gmtra.damage.`**`rail_flood`**(*x*, *curve*, *events*, *param_values*, *val_cols*, *wbreg_lookup*, *sensitivity=False*)

Function to estimate the range of flood damages to an individual road asset.

Arguments: *x* : row in geopandas GeoDataFrame that represents an individual asset.

curve : A pandas DataFrame with a set of damage curves for railways.

events : A list with the unique flood events.

param_values : A NumPy Array with sets of parameter values we would like to test.

val_cols : A list with the unique flood events in row **x**.

wbreg_lookup : a dictioniary that relates a country ID (ISO3 code) with its World Bank region.

Optional Arguments: *sensitivity* : Default is **False**. Set to **True** if you would like to return all damage values to be able to perform a sensitivity analysis.

Returns: *list* : A list with the range of possible damages to the specified asset, based on the parameter set.

3.8.11 Regional damage to all bridge assets

`gmtra.damage.`**`regional_bridge`**(*file*, *data_path*, *param_values*, *income_lookup*, *eq_curve*, *design_tables*, *depth_threshs*, *wind_threshs*, *rail=False*)

Function to estimate the summary statistics of all bridge damages in a region

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

param_values : A NumPy Array with sets of parameter values we would like to test.

income_lookup : A dictionary that relates a country ID (ISO3 code) with its World Bank income goup.

eq_curve : A pandas DataFrame with unique damage curves for earthquake damages.

design_table : A NumPy array that represents the design standards for different bridge types, dependent on road type.

depth_thresh : A list with failure depth thresholds.

wind_threshs : A list with failure wind gustspeed thresholds.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.8.12 Cyclone damage to all assets in a region

`gmtra.damage.`**`regional_cyclone`** (*file*, *data_path*, *events*, *param_values*, *rail=False*)

Function to estimate the summary statistics of all cyclone damages in a region to road assets. Cyclone damage for roads is currently based on clean-up cost and repairs.

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

events : A list with the unique cyclone events.

param_values : A NumPy Array with sets of parameter values we would like to test.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.8.13 Earthquake damage to all assets in a region

`gmtra.damage.`**`regional_earthquake`** (*file*, *data_path*, *global_costs*, *paved_ratios*, *events*, *wbreg_lookup*, *rail=False*)

Function to estimate the summary statistics of all earthquake damages in a region to road assets

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

global_costs : A pandas DataFrame with the total cost for different roads in different World Bank regions. These values are based on the ROCKS database.

paved_ratios : A pandas DataFrame with road pavement percentages per country for each road type.

events : A list with the unique earthquake events.

wbreg_lookup : a dictioniary that relates countries (in ISO3 codes) with World Bank regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.8.14 Flood damage to all assets in a region

`gmtra.damage.`**`regional_flood`** (*file*, *hzd*, *data_path*, *global_costs*, *paved_ratios*, *flood_curve_paved*, *flood_curve_unpaved*, *events*, *wbreg_lookup*, *rail=False*)

Function to estimate the summary statistics of all flood damages in a region to road assets

Arguments: *file* : path to the .feather file with all bridges of a region.

hzd : abbrevation of the hazard we want to intersect. **FU** for river flooding, **PU** for surface flooding and **CF** for coastal flooding.

data_path : file path to location of all data.

global_costs : A pandas DataFrame with the total cost for different roads in different World Bank regions. These values are based on the ROCKS database.

paved_ratios : A pandas DataFrame with road pavement percentages per country for each road type.

flood_curve_paved : A pandas DataFrame with a set of damage curves for paved roads.

flood_curve_unpaved : A pandas DataFrame with a set of damage curves for unpaved roads.

events : A list with the unique flood events.

wbreg_lookup : a dictioniary that relates a country ID (ISO3 code) with its World Bank region.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.9 9. Sensitivity analysis

3.9.1 Bridge sensitivity analysis

`gmtra.sensitivity.`**`regional_bridge`**(*file*, *data_path*, *param_values*, *income_lookup*, *eq_curve*, *design_tables*, *depth_threshs*, *wind_threshs*, *rail=False*)

Function to estimate the summary statistics of all bridge damages in a region

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

param_values : A NumPy Array with sets of parameter values we would like to test.

income_lookup : A dictionary that relates a country ID (ISO3 code) with its World Bank income goup.

eq_curve : A pandas DataFrame with unique damage curves for earthquake damages.

design_table : A NumPy array that represents the design standards for different bridge types, dependent on road type.

depth_thresh : A list with failure depth thresholds.

wind_threshs : A list with failure wind gustspeed thresholds.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.9.2 Cyclone damage sensitivity analysis

`gmtra.sensitivity.`**`regional_cyclone`**(*file*, *data_path*, *events*, *param_values*, *rail=False*)

Function to estimate the summary statistics of all cyclone damages in a region to road assets

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

events : A list with the unique cyclone events.

param_values : A NumPy Array with sets of parameter values we would like to test.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.9.3 Earthquake damage sensitivity analysis

`gmtra.sensitivity.`**`regional_earthquake`**(*file*, *data_path*, *global_costs*, *paved_ratios*, *events*, *wbreg_lookup*, *rail=False*)

Function to estimate the summary statistics of all earthquake damages in a region to road assets

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

global_costs : A pandas DataFrame with the total cost for different roads in different World Bank regions. These values are based on the ROCKS database.

paved_ratios : A pandas DataFrame with road pavement percentages per country for each road type.

events : A list with the unique earthquake events.

wbreg_lookup : a dictioniary that relates countries (in ISO3 codes) with World Bank regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with summary damage statistics for the loaded region.

3.9.4 Flood damage sensitivity analysis

`gmtra.sensitivity.`**`regional_flood`**(*file*, *hazard*, *data_path*, *global_costs*, *paved_ratios*, *flood_curve_paved*, *flood_curve_unpaved*, *events*, *wbreg_lookup*, *rail=False*)

Function to estimate the summary statistics of all flood damages in a region to road assets

Arguments: *file* : path to the .feather file with all bridges of a region.

data_path : file path to location of all data.

global_costs : A pandas DataFrame with the total cost for different roads in different World Bank regions. These values are based on the ROCKS database.

paved_ratios : A pandas DataFrame with road pavement percentages per country for each road type.

flood_curve_paved : A pandas DataFrame with a set of damage curves for paved roads.

flood_curve_unpaved : A pandas DataFrame with a set of damage curves for unpaved roads.

events : A list with the unique flood events.

wbreg_lookup : a dictioniary that relates a country ID (ISO3 code) with its World Bank region.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

Returns: *DataFrame* : a pandas DataFrame with all damage results for the loaded region per road or railway asset type.

3.10 10. Parallel functions

`gmtra.parallel.`**`bridge_extraction`**(*save_all=False*)

Function to extract all bridges from OpenStreetMap.

Optional Arguments: *save_all* : Default is **False**. Set to **True** if you would like to save all bridges of the world in one csv file. Will become a big csv!

`gmtra.parallel.`**`SSBN_merge`**(*from_=0*, *to_=235*)

Merge all countries parallel.

Optional Arguments: *from_* : Default is **0**. Set to a different value if you would like to select a different subset.

*to_* : Default is **235**. Set to a different value if you would like to select a different subset.

`gmtra.parallel.`**`tree_values`**(*rail=False*)

Function to run intersection with global tree density map for all regions parallel.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`liquefaction_overlays`**(*rail=False*)

Function to run intersection with global liquefaction map for all regions parallel.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`hazard_intersection`**(*hzd*, *rail=False*, *from_=0*, *to_=46433*)

Function to run intersection with hazard data for all regions parallel.

Arguments: *hzd* : abbrevation of the hazard we want to intersect. **FU** for river flooding, **PU** for surface flooding and **CF** for coastal flooding.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

*from_* : Default is **0**. Set to a different value if you would like to select a different subset.

*to_* : Default is **46433**. Set to a different value if you would like to select a different subset.

`gmtra.parallel.`**`exposure_analysis`**(*rail=False*)

Get exposure statistics for all road or railway assets in all regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`bridge_damage`**(*rail=False*)

Function to calculate the damage to bridges for all regions and all hazards.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`cyclone_damage`**(*rail=False*)

Function to calculate the cyclone damage to road or railway assets for all regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`earthquake_damage`**`(`*rail=False*`)`

Function to calculate the earthquake damage to road or railway assets for all regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`flood_damage`**`(`*hazard*, *rail=False*`)`

Function to calculate the flood damage to road or railway assets for all regions.

Arguments: *hzd* : abbrevation of the hazard we want to intersect. **FU** for river flooding, **PU** for surface flooding and **CF** for coastal flooding.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

`gmtra.parallel.`**`bridge_sensitivity`**`(`*rail=False*, *region_count=1000*`)`

Function to calculate the damage to bridges for all regions and all hazards.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

region_count : Default is **1000**. Change this number if you want to include a different amount of regions.

`gmtra.parallel.`**`cyclone_sensitivity`**`(`*rail=False*, *region_count=1000*`)`

Function to perform the caculations for a sensitivity analysis related to cyclone damage to road or railway assets for all regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

region_count : Default is **1000**. Change this number if you want to include a different amount of regions.

`gmtra.parallel.`**`earthquake_sensitivity`**`(`*rail=False*, *region_count=1000*`)`

Function to perform the caculations for a sensitivity analysis related to earthquake damage to road or railway assets for all regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

region_count : Default is **1000**. Change this number if you want to include a different amount of regions.

`gmtra.parallel.`**`flood_sensitivity`**`(`*hazard*, *rail=False*, *region_count=1000*`)`

Function to perform the caculations for a sensitivity analysis related to flood damage to road or railway assets for all regions.

Optional Arguments: *rail* : Default is **False**. Set to **True** if you would like to intersect the railway assets in a region.

region_count : Default is **1000**. Change this number if you want to include a different amount of regions.

Index