WISC Documentation

Release 0.2

Elco Koks

Mar 11, 2020

Documentation

A simple ReadtheDocs reference guide to all functions used to estimate losses per building type due to Windstorms in Europe.

Note that the explanation of the functions in this reference guide is mainly focused on the explanation of the inputs and outputs. Please refer to the source code for a step-by-step explanation of the code.

CHAPTER 1

Getting started

Recommended option is to use a Miniconda environment to work in for this project, relying on conda to handle some of the trickier library dependencies.

The most easiest way to do so is to use the requirements.yml file as provided in the github page.

To create the environment using the yaml file:

```
conda env create -f environment.yml
```

In case of no access to the GitHub page, the other option would be to copy-paste the list below and save this to an *environment.yml* file (note the indendation):

```
name: WISC

dependencies:
        - python=3.6
        - gdal
        - numpy
        - pandas
        - geopandas
        - pathos
        - rasterio
        - rasterstats
        - matplotlib
        - basemap
        - SAlib
        - urllib3
        - scikit-learn
        - xlrd
        - tqdm
```

CHAPTER 2

Utils

2.1 Load config file

`scripts.utils.load_config()`

Read config.json

2.2 Clean directory

`scripts.utils.clean_dir(dirpath)`

This function can be used to fully clear a directory.

Arguments: *dirpath* (string) – path to directory to be cleared from files

2.3 Remove files from directory

`scripts.utils.remove_files(dirpath, startname)`

This function can be used to delete specific files from a directory. In general this function is used to clean country files from the 'calc' directory.

Arguments: *dirpath* (string) – path to directory in which the files should be removed

startname (string) – the substring to be searched for in the files

2.4 Create folder structure

`scripts.utils.create_folder_structure(data_path, country)`

Create the directory structure for the output.

Arguments: *base_path (string) – path to directory where folder structure should be created.

regionalized (bool) – specify whether also the folders for a regionalized analyse should be created (default: **True**)

2.5 Integer to date

`scripts.utils.`**`int2date`**(*argdate: int*)

If you have date as an integer, use this method to obtain a datetime.date object.

Arguments: *argdate* (int) – Date as a regular integer value (example: **20160618**)

Returns: *dateandtime.date* – A date object which corresponds to the given value **argdate**.

2.6 Get integer from string

`scripts.utils.`**`get_num`**(*x*)

Grab all integers from string.

Arguments: *x* (string) – string containing integers

Returns: *integer* – created from string

2.7 Create dictionary with country names in GeoFabrik

`scripts.utils.`**`country_dict_geofabrik`**()

Create a dictionary to convert ISO2 codes to Geofabrik country names.

Returns: *dictionary* – lookup between ISO2 codes and Geofabrik country names.

2.8 Download OSM file from GeoFabrik

`scripts.utils.`**`download_osm_file`**(*country*)

Download OSM file from Geofabrik.

Arguments: *country* (string) – ISO2 string code of country

Returns: *downloaded OSM file*

CHAPTER 3

Functions

3.1 Regional exposure

`scripts.functions.`**`region_exposure`**(*region*, *include_storms=True*, *event_set=False*, *sens_analysis_storms=[]*, *save=True*)

Create a GeoDataframe with exposure and hazard information for each building in the specified region.

Arguments: *region* (string) – NUTS3 code of region to consider.

include_storms (bool) – if set to False, it will only return a list of buildings and their characteristics (default: **True**)

event_set (bool) – if set to True, we will calculate the exposure for the event set instead of the historical storms (default: **True**)

sens_analysis_storms (list) – if empty, it will fill with default list

save (bool) – boolean to decide whether you want to save the output to a csv file (default: **True**)

Returns: *GeoDataFrame* with all **hazard** and **exposure** values.

3.2 Regional losses

`scripts.functions.`**`region_losses`**(*region*, *storm_event_set=False*, *sample=(5, 0, 95, 20, 80)*)

Estimation of the losses for all buildings in a region for a pre-defined list of storms.

Arguments: *region* (string) – nuts code of region to consider.

storm_event_set (bool) – calculates all regions within a country parallel. Set to **False** if you have little capacity on the machine (default: **True**).

sample (tuple) – tuple of parameter values. This is a dummy placeholder, should be filled with either **load_sample(country)** values or **sens_analysis_param_list**.

Returns: *pandas Dataframe* – pandas dataframe with all buildings of the region and their **losses** for each wind storm.

3.3 Regional sensitivity analysis

`scripts.functions.`**`region_sens_analysis`**(*region*, *samples*, *sens_analysis_storms=[]*, *save=True*)

Perform a sensitivity analysis for the specified region, based on a predefined list of storms.

Arguments: *region* (string) – nuts code of region to consider.

samples (list) – list o tuples, where each tuple is a **unique** set of parameter values.

sens_analysis_storms (list) – if empty, it will fill with default list

save (bool) – boolean to decide whether you want to save the output to a csv file (default: **True**)

Returns: *list* – list with the total losses per storm for all parameter combinations

3.4 Loss calculation

`scripts.functions.`**`loss_calculation`**(*storm*, *country*, *output_table*, *max_dam*, *curves*, *sample*)

Calculate the losses per storm.

Arguments: *storm* (string) – date of the storm.

region (string) – NUTS3 code of region to consider.

output_table (GeoDataFrame) – GeoDataFrame with all buildings and the wind speed values for each storm.

max_dam (numpy array) – table with maximum damages per building type/land-use class.

curves (pandas dataframe) – fragility curves for the different building types.

sample (list) – ratios of different curves used in this study. See the **Sensitivity analysis documentation** for an explanation.

Returns: *pandas Series* – losses to all buildings for the specified storm

3.5 Fetch buildings from OpenStreetMap

`scripts.functions.`**`fetch_buildings`**(*data_path*, *country*, *region=''*, *regional=False*)

This function directly reads the building data from osm, instead of first converting it to a shapefile

Arguments: *data_path* (string) – string of data path where all data is located.

country (string) – ISO2 code of country to consider.

region (string) – NUTS3 code of region to consider.

regional (boolean) – Boolean to decide whether this should be done on a regional level or for the entire country (default: **False**).

Returns: *Geodataframe* – Geopandas dataframe with all buildings in the specified **region** or **country**.

3.6 Create .poly files

`scripts.functions.`**`poly_files`**(*data_path*, *country*)

This function will create the .poly files from the nuts 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* (string) – string of data path where all data is located.

country (string) – ISO2 code of country to consider.

Returns: *.poly file* – .poly file for each NUTS3 in a new directory in the working directory of the country.

3.7 Extract buildings from OpenStreetMap

`scripts.functions.`**`extract_buildings`**(*area*, *country*, *NUTS3=True*)

Extracts building from OpenStreetMap pbf file and saves it to an ESRI shapefile

Arguments: *area* (string) – name of area to clip

country (string) – ISO2 code of country to consider.

NUTS3 (bool) – specify whether it will be a clip of NUTS3 region or the whole country (default: **True**)

3.8 Convert buildings table from EPSG:4326 to EPSG:3035

`scripts.functions.`**`convert_buildings`**(*area*, *country*)

Converts the coordinate system from EPSG:4326 to EPSG:3035.

Arguments: *area* (string) – name of area (most often NUTS3) for which buildings should be converted to European coordinate system.

country (string) – ISO2 code of country to consider.

Returns: *GeoDataframe* – Geopandas dataframe with all buildings of the selected area

3.9 Build list of storms

`scripts.functions.`**`get_storm_list`**(*data_path*)

Small function to create a list of with path strings to all storms.

Arguments: *data_path* (string) – string of data path where all data is located.

Returns: *list* – list with the path strings of all storms

3.10 Build list of storms for event set

`scripts.functions.`**`get_event_storm_list`**(*data_path*)

Small function to create a list of with path strings to all storms in the event set.

Arguments: *data_path* (string) – string of data path where all data is located.

Returns: *list* – list with the path strings of all storms

3.11 Load storms for sensitivity analysis

`scripts.functions.`**`load_sens_analysis_storms`**(*storm_name_list=['19991203', '19900125', '20090124', '20070118', '19991226']*)

This file load the storms used to perform the sensitivity analysis.

Arguments: *storm_name_list* (list) – list of storms to include in the sensitivity analysis. The default storms are **Anatol**, **Daria**, **Klaus**, **Kyrill** and **Lothar**.

Returns: *storm_list* (list) – same list of storms but now with full paths to location in directory.

3.12 Load table of maximum damages

`scripts.functions.`**`load_max_dam`**(*data_path*)

Small function to load the excel with maximum damages.

Arguments: *data_path* (string) – string of data path where all data is located.

Returns: *dataframe* – pandas dataframe with maximum damages per landuse.

3.13 Load fragility curves

`scripts.functions.`**`load_curves`**(*data_path*)

Small function to load the csv file with the different fragility curves.

Arguments: *data_path* (string) – string of data path where all data is located.

Returns: *dataframe* – pandas dataframe with fragility curves

3.14 Load curves ratios

`scripts.functions.`**`load_sample`**(*country*)

Will load the ratio of each curve and landuse to be used.

Arguments: *country* (string) – ISO2 code of country to consider.

Returns: *tuple* – tuple with ratios for the selected country. See the **documentation** for an explanation.

['c2', 'c3', 'c4','lu1','lu2']

3.15 Load OpenStreetMap data

`scripts.functions.`**`load_osm_data`**(*data_path, country, region='', regional=False*)

This function loads the OSM file for the country

Arguments: *data_path* (string) – string of data path where all data is located.

country (string) – ISO2 code of country to consider.

region (string) – NUTS3 code of region to consider.

regional (boolean) – Boolean to decide whether this should be done on a regional level or for the entire country (default: **False**).

Returns: opened OSM file to use in the fetch_roads function.

3.16 Clip landuse

`scripts.functions.`**`clip_landuse`**(*data_path*, *country*, *region*, *outrast_lu*)

Clip the landuse from the European Corine Land Cover (CLC) map to the considered country.

Arguments: *data_path* (string) – string of data path where all data is located.

country (string) – ISO2 code of country to consider.

outrast_lu (string) – string path to location of Corine Land Cover dataset

Returns: *raster* (geotiff) – returns a raster file with only the landuse of the clipped **region** or **country**.

3.17 Clip OpenStreetMap file

`scripts.functions.`**`clip_osm`**(*data_path*, *osm_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: *osm_path* (string) – path string to the osm.pbf file of the continent associated with the country.

area_pol (string) – path string to the .poly file, made through the 'create_poly_files' function.

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

Returns: a clipped .osm.pbf file.

3.18 Obtain raster value from geotiff grid

`scripts.functions.`**`get_raster_value`**(*centroid*, *out_image*, *out_transform*)

Small function to obtain raster value from rastermap, using point_query.

Arguments: *centroid* (shapely geometry) – shapely geometry of centroid of the building.

out_image (numpy array) – numpy array of grid.

out_transform (Affine) – georeference of numpy array.

Returns: *integer* – raster value corresponding to location of centroid

3.19 Create excel file with summary statistics

`scripts.functions.`**`summary_statistics_losses`**()

This function creates the file 'output_storms.xlsx'. This file is required to create the summary figures.

Returns: *output_storms.xlsx* (excel file) – Excel file with summary outcomes

CHAPTER 4

Analyze

4.1 Create exposure table for a country

`scripts.analyze.`**`exposure`**(*country*, *include_storms=True*, *parallel=True*, *save=True*)

Creation of exposure table of the specified country.

Arguments: *country* (string) – ISO2 code of country to consider.

include_storms (bool) – if set to False, it will only return a list of buildings and their characteristics (default: **True**).

parallel (bool) – calculates all regions within a country parallel. Set to False if you have little capacity on the machine (default: **True**).

save (bool) – boolean to decide whether you want to save the output to a csv file (default: **True**).

Returns: *GeoDataframe* – Geopandas dataframe with all buildings of the country and potential exposure to wind

4.2 Estimate all losses for a country

`scripts.analyze.`**`losses`**(*country*, *parallel=True*, *event_set=False*, *save=True*)

Creation of exposure table of the specified country

Arguments: *country* (string) – ISO2 code of country to consider.

parallel (bool) – calculates all regions within a country parallel. Set to False if you have little capacity on the machine (default: **True**).

event_set (bool) – if set to True, we will calculate the losses for the event set instead of the historical storms (default: **True**).

save (bool) – boolean to decide whether you want to save the output to a csv file (default: **True**).

Returns: *GeoDataframe* – Geopandas dataframe with all buildings of the country and their **losses** for each wind storm.

4.3 Estimate risk per country

`scripts.analyze.`**`risk`**(*country*, *save=True*, *parallel=True*)

Estimate risk based on event set

Arguments: *country* (string) – ISO2 code of country to consider.

path_to_eventset (string) – the location of the event set in the data directory

save (bool) – boolean to decide whether you want to save the output to a csv file (default: **True**).

parallel (bool) – calculates all regions within a country parallel. Set to False if you have little capacity on the machine (default: **True**).

4.4 Create exposure table for all countries

`scripts.analyze.`**`all_countries_exposure`**()

Function to estimate the exposure for all countries consecutively.

4.5 Estimate losses for all countries

`scripts.analyze.`**`all_countries_losses`**()

Function to estimate the losses for all countries consecutively.

4.6 Estimate risk for all countries

`scripts.analyze.`**`all_countries_risk`**()

Function to estimate the risk for all countries consecutively.

CHAPTER 5

Sensitivity Analysis

5.1 Perform sensitivity analysis for a country

`scripts.sensitivity.`**`calculate`**(*country*, *parallel=True*, *save=True*)

Base function to perform the sensitivity analysis for a country.

Arguments: *country* (string) – ISO2 code of country to consider.

parallel (bool) – calculates all regions within a country parallel. Set to False if you have little capacity on the machine (default: **True**).

save (bool) – boolean to decide whether you want to save the output to a csv file (default: **True**).

Returns: *dataframe* – Pandas dataframe with all outcomes per parameter combination.

5.2 Prepare sensitivity analysis

`scripts.sensitivity.`**`prepare_sens_analysis`**(*storm_name_list=[]*)

Function to prepare the sensitivity analysis for a country.

Arguments: *storm_name_list* (list) – list of storms to include in sensitivity analysis. If kept empty, default storms will be used.

Returns: *param_values* (list) – list of 5000 combinations of parameter values to be used in sensitivity analysis.

storm_name_list (list) – list of storms to include in sensitivity analysis.

5.3 Read outcomes of sensitivity analysis

`scripts.sensitivity.`**`read_outcomes_sens_analysis`**()

Function to write the output of the sensitivity analysis to figures.

CHAPTER 6

Plotting

6.1 Create exposure table

`scripts.plotting.`**`loss_per_country`** (*figure_output_path='test_country.png'*)

This function is used to plot the total losses per year per country.

Arguments: *figure_output_path* (string) – path to location where you want to save the figure

Returns: *A saved figure*

6.2 Estimate losses per building

`scripts.plotting.`**`loss_per_sector`** (*figure_output_path='test_sector.png'*)

This function is used to plot the total losses for the following sectors: Residential,Industrial/Commercial,Transport,Other uses,Agriculture.

Arguments: *figure_output_path* (string) – path to location where you want to save the figure

Returns: *A saved figure*

6.3 Estimate losses per building

`scripts.plotting.`**`risk_map`** (*figure_output_path='test_risk_map.png'*)

This function is used to create a map with the total risk per region.

Arguments: *figure_output_path* (string) – path to location where you want to save the figure

Returns: *A saved figure*