toscana

Release 0.2.3

Apolline Ferry

Apr 22, 2025

CONTENTS

An overview of the most useful functions is presented in part *Module contents*. For a deeper view of the package, see *All features*.

CHAPTER

ONE

MODULE CONTENTS

toscana.**calculate_village_distribution_characteristics**(*path_csv_folder*, *path_raster_files*, *path_final_output_folder*, *path_shapefiles*, *grid_gpd*, *path_meteorological_folder*, *village_name*, *village_INSEE_code=None*, *village_departement=None*, *average=True*, *population_column='POPULATION'*, *nb_tests=5*)

Calculate and gather in a dataframe the village characteristics and distribution characteristics.

The village characteristics include geographical data of the village (area, perimeter, coordinates) with administratives information (population, department). `departement` and `INSEE_code` are set to None by default to include the case of a territory studied that do not correspond to a french municipality. `village_name` could correspond to the name of territory studied if it is not a municipality. Then, results from SEBE simulation are used to obtain the distribution of the average annual irradiation and the associated fitted Johnson's SU distribution. Statistical indicators to describe the distribution and the fitted Johnson's SU are obtained, as well as two indicators, SVI and DFI, to describe the far mask (topography) and the meteorologial situation of the municipality. The dataframe is then exported in a csv file.

Parameters

- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **path_raster_files** (*pathlib.Path*) – path of the folder with temporary raster files (define in main function)
- **path_final_output_folder** (*pathlib.Path*) – path of the folder with the final results (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **grid_gpd** (*GeoDataFrame*) – geopandas grid file
- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where are saved all the meteorological files
- **village_name** (*str*) – municipality name
- **village_INSEE_code** (*str, optional*) – INSEE code of the municipality, by default None
- **village_departement** (*str, optional*) – departement the municipality, by default None
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True

- **population_column** (*str, optional*) – name of the column with number of inhabitants of the studied municipality in the municipality footprint shapefile, by default "POPULATION"
- **nb_tests** (*int, optional*) – number of time to test downloading the horizon file if there is a connection problem, by default 5

Returns
df – dataframe with all the characteristics of the village and of the solar distribution

Return type
Dataframe

toscana.**create_monthly_weather_file**(*path_meteorological_folder*, *list_month*, *path_gdf_centroid*, *average=True*)

Obtain monthly meteorological files from annual meteorological files for given months. Subfolders to save the monthly weather files are created.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **list_month** (*list*) – list of the months for which monthly weather files will be generated
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Raises

- **TypeError** – month must be a int (or a string of numbers)
- **AssertionError** – month must be between 1 and 12

toscana.**create_period_weather_file**(*name_folder_period*, *path_meteorological_folder*, *path_gdf_centroid*, *list_days*, *average=True*)

Obtain meteorological files for a given period from annual meteorological files. Subfolders to save the new weather files are created.

Parameters

- **name_folder_period** (*str*) – name of the folder where to save the meteorological files for the given period
- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **list_days** (*list*) – list of the day numbers of the year included in the given period
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Returns
path_period_weather_file – path of the folder with the created meteorological files for the given period

Return type
pathlib.Path

Raises

- **TypeError** – list_days must be a list of int (or a list of strings of numbers)
- **AssertionError** – day numbers must be between 1 and 365

toscana.**create_winter_summer_month_weather_file**(*path_meteorological_folder*, *path_gdf_centroid*, *list_winter_month=[]*, *list_summer_month=[]*, *average=True*)

Obtain meteorological files for two periods from annual meteorological files. One of the period should correspond to winter months, and is defined by default from January to March and from November to December. The second period should correspond to summer months, and is defined by default as all other months from the winter months. Subfolders to save the new weather files are created.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **list_winter_month** (*list, optional*) – list of the winter months, by default []
- **list_summer_month** (*list, optional*) – list_of the summer months, by default []
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Returns

- **path_folder_summer** (*pathlib.Path*) – path of the folder with the created summer meteorological files
- **path_folder_winter** (*pathlib.Path*) – path of the folder with the created winter meteorological files

Raises

- **TypeError** – list_winter_month and list_summer_month must be list of int (or list of strings of numbers)
- **AssertionError** – month must be between 1 and 12

toscana.**display_results**(*path_raster_files*, *path_final_output_folder*, *rectangle_elevation=True*, *rectangle_SEBE=False*, *bool_johnsonsu=True*, *column_prefix='sol_'*, *statistics='mean'*, *save_plot=True*, *name_plot='Distribution_irradiation.png'*, *bool_title_histo=False*, *title_histo='Distribution of average \nannual rooftop irradiation'*, *bool_title_elevation=False*, *title_elevation="Map of the municipality's elevation"*, *bool_title_SEBE=False*, *title_SEBE="Map of the municipality's average \nannual irradiation on surfaces"*, *bool_title_stats=False*, *title_stats="Map of the municipality's average annual rooftop\nirradiation per building"*, *label_stats='Irradiation (kWh/m^2)'*)

Display the results : the resample DSM of the village, the raster with SEBE results, the shapefile with building footprints (with buffer) and their average annual irradiation on rooftops and the histogram with the average annual irradiation received by rooftops and the associated Johnson's SU fitted distribution.

path_raster_files
[pathlib.Path] path of the folder with temporary raster files (define in main function)

path_final_output_folder
[pathlib.Path] path of the folder with the final results (define in main function)

rectangle_elevation
[bool, optional] boolean value to display or not the borders of the elevation file (recommended to set to False if the raster is a raster coming from *clip_raster* function), by default True

rectangle_SEBE
[bool, optional] boolean value to display or not the borders of the merge SEBE raster (recommended to set to False if the raster is a raster coming from *clip_raster* function), by default False

bool_johnsonsu
[bool, optional] boolean value to display or not the Johnson's SU fitted distribution, by default True

column_prefix
[str, optional] prefix of the column that have been chosen to create the statistics, by default 'sol_'

statistics
[str, optional] name of the suffix of the column to display, by default 'mean'

save_plot
[bool, optional] boolean value to save or not a figure with the distribution, by default True

name_plot
[str, optional] name of the file saved with the figure of the distribution, by default "Distribution_irradiation.png

bool_title_histo
[bool, optional] boolean value to display or not the title, by default False

title_histo
[str, optional] title of the figure, by default "Distribution of average annual rooftop irradiation"

bool_title_elevation
[bool, optional] boolean value to display or not the title, by default False

title_elevation
[str, optional] title of the figure, by default "Map of the municipality's elevation"

bool_title_SEBE
[bool, optional] boolean value to display or not the title, by default False

title_SEBE
[_type_, optional] title of the figure, by default f"Map of the municipality's average

annual irradiation on surfaces"

bool_title_stats
[bool, optional] boolean value to display or not the title, by default False

title_stats
[_type_, optional] title of the figure, by default f"Map of the municipality's average annual rooftop

irradiation per building"

label_stats
[str, optional] lalbel of the colorbar, by default "Irradiation (kWh/m^2)"

toscana.**download_and_extract_BDTOPO_data**(*departement*, *path_raw_data_folder*)

Download shapefiles (municipalities footprints and buildings footprints) from the BDTOPO database on the IGN website giving a the number of the `departement`.

A packed folder is downloaded and then extracted. The version downloaded is the 15-12-2023 version.

Parameters

- **departement** (*str*) – departement the municipality
- **path_raw_data_folder** (*pathlib.Path*) – path of the folder with raw data (downloaded) files (define in main function)

Returns

- **path_departement_municipalities_footprint** (*pathlib.Path*) – path of the municipalities footprints for the department of interest
- **path_input_departement_building** (*pathlib.Path*) – path of the buildings footprints for the department of interest

toscana.**download_extract_and_merge_DEM_from_OpenDEM**(*path_raw_data_folder*, *path_shapefiles*, *bool_France=True*, *threshold=100*)

Download and extract the DEM raster tile from the OpenDEM website according to the municipality footprint extent.

The raster tiles are then merged together to have a unique merge DEM. First, the edge tile are obtained. If the municipality footprint extent is close to the border of the tile, the neighbouring tile is also download. All the tiles that overlap with the extent of municipality footprint are downloaded and then merge together. If the studied territory is not located in France, `bool_France` should be set to False.

Parameters

- **path_raw_data_folder** (*pathlib.Path*) – path of the folder with raw data (downloaded) files (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **bool_France** (*bool, optional*) – boolean value to indicate or not if the territory of interest is located in France or not, by default True
- **threshold** (*int, optional*) – minimal value (in meters) of the difference between the municipality extent and the border of the raster tile, by default 100

Returns
path_merge_dem – path where is saved the merged DEM

Return type
pathlib.Path

toscana.**iterate_on_grid**(*grid_gpd*, *path_final_output_folder*, *path_clip_files*, *path_raster_files*, *path_meteorological_folder*, *path_csv_folder*, *path_meteorological_subfolder=''*, *wall_limit=0.1*, *bool_global=True*, *utc=1*, *bool_save_sky_irradiance=True*, *albedo=0.15*, *restart_tile=1*, *average=True*)

Function used to iterate on the grid (run simulation for each grid tile) : the DSM, DHM and grid are clipped at the extent of one tile, wall aspects and wall heights are calculated and then SEBE simulation are run.

SEBE calculation are not done for the tiles for which the meteorological files could not have been downloaded. The iteration starts with the first tile (by default) but can be changed by changing restart_tile. Exceptions are included to consider if SEBE calculation could not be run : problem when calculating the sky irradiance distribution (if direct and diffuse component are derived from global, error could appear when reprojecting the component on each patch of the sky vault) or if some wrong values are present in meteorological data (averaged or not). The list of the tiles for which the SEBE calculation could not have been done (due to a missing meteorological files, to an error in the calculation of the diffuse and direct component from the global, or due to an other error) are saved in a csv files with the corresponding error.

Parameters

- **grid_gpd** (*GeoDataFrame*) – geopandas grid file

- **path_final_output_folder** (*path-like*) – path of the folder where to save the final results (define in main function)
- **path_clip_files** (*path-like*) – path of the folder with temporary clip files (define in main function)
- **path_raster_files** (*path-like*) – path of the folder with temporary raster files (define in main function)
- **path_meteorological_folder** (*path-like*) – path of the folder where are saved all the meteorological files
- **path_meteorological_subfolder** (*path-like*) – path of the subfolder where are saved the meteorological files for that simulation
- **path_csv_folder** (*path-like*) – path of the folder with temporary csv files (define in main function)
- **wall_limit** (*float, optional*) – minimum difference of height to consider a pixel as a wall, by default 0.1
- **bool_global** (*bool, optional*) – boolean value to calculate or not the diffuse and direct irradiance values from global irradiance values, by default True
- **utc** (*int, optional*) – value of utc time, by default 1 from -12 to 12, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **bool_save_sky_irradiance** (*bool, optional*) – boolean to save or not the sky irradiance data, by default True
- **albedo** (*float, optional*) – value of the albedo, by default 0.15 from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **restart_tile** (*int, optional*) – number of the first tile on which to run SEBE simulation (to change to start not from the beginning), by default 1
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True

Raises

AssertionError – restart_tile must be smaller than the number of grid tiles and higher than 1 (first tile)

toscana.**launch_iterate_on_grid_per_month**(*grid_gpd*, *list_month*, *path_final_output_folder*, *path_clip_files*, *path_raster_files*, *path_meteorological_folder*, *path_csv_folder*, *wall_limit=0.1*, *bool_global=True*, *utc=1*, *bool_save_sky_irradiance=True*, *restart_tile=1*, *average=True*, *list_albedo_month=[]*, *albedo_m=0.15*)

Function used to iterate on the grid for several months.

Parameters

- **grid_gpd** (*GeoDataFrame*) – geopandas grid file
- **list_month** (*list*) – list of the months that should be simulated
- **path_final_output_folder** (*path-like*) – path of the folder where to save the final results (define in main function)
- **path_clip_files** (*path-like*) – path of the folder with temporary clip files (define in main function)
- **path_raster_files** (*path-like*) – path of the folder with temporary raster files (define in main function)

- **path_meteorological_folder** (*path-like*) – path of the folder where are saved all the meteorological files
- **path_csv_folder** (*path-like*) – path of the folder with temporary csv files (define in main function)
- **wall_limit** (*float, optional*) – minimum difference of height to consider a pixel as a wall, by default 0.1
- **bool_global** (*bool, optional*) – boolean value to calculate or not the diffuse and direct irradiance values from global irradiance values, by default True
- **utc** (*int, optional*) – value of utc time, by default 1 from -12 to 12, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **bool_save_sky_irradiance** (*bool, optional*) – boolean to save or not the sky irradiance data, by default True
- **restart_tile** (*int, optional*) – number of the first tile on which to run SEBE simulation (to change to start not from the beginning), by default 1
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True
- **list_albedo_month** (*list, optional*) – monthly value of the albedo, by default [] from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **albedo_m** (*float, optional*) – average albedo value (for all the months) if monthly values are not defined, by default 0.15 from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation

Raises

- **TypeError** – list_month must be a list of int (or a list of strings with numbers)
- **AssertionError** – value in list_month should be between 1 and 12

toscana.**obtain_grid**(*path_shapefiles*, *projection='IGNF:ETRS89LAEA'*, *horizontal_spacing=1000*, *vertical_spacing=1000*)

Create the grid in order to run the SEBE simulation for the studied village.

It is a grid of 1000m (horizontal_spacing) per 1000m (vertical_spacing) per default (in IGNF ETRS89LAEA reference system by default, modify ``projection``to change it).

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **horizontal_spacing** (*int, optional*) – horizontal size of grid tiles, by default 1000
- **vertical_spacing** (*int, optional*) – vertical size of grid tiles, by default 1000

Returns

grid_gpd – geopandas grid file

Return type

GeoDataFrame

toscana.**obtain_meteorological_files**(*path_meteorological_folder*, *path_shapefiles*, *save_temp_file=True*, *average=True*, *option_no_average_if_one_problem=False*, *option_no_average_for_the_error_tile=False*, *nb_tests=5*)

Obtain the meteorological files needed to run the SEBE simulations.

Folders to save the files are created. Centroids of the grid tiles are created. They are modified to include the 12 closest centroids. The metorological files are obtained in epw format, and then transform in txt format, needed for the simulations. These temporary files are saved only if `save_temp_file` is set to True. An average of the meteorological file from the tile and the 12 closest tiles is done if `average` is set to True (default value) (due to the low resolution of the meteorological data). If there is some meteorological files that could not be downloaded, the average is done only with the valid neighbouring tiles (default). `option_no_average_if_one_problem` and `option_no_average_for_the_error_tile` will be used only if `average` is set to True.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where to save all the meteorological files
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **save_temp_file** (*bool, optional*) – boolean value to save or not temporary meteorological files, by default True
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True
- **option_no_average_if_one_problem** (*bool, optional*) – boolean value to don't average any meteorological file if there is one problem for one tile or one of the neighbouring tile among all the tiles, by default False
- **option_no_average_for_the_error_tile** (*bool, optional*) – boolean value to don't average the meteorological file if there is one problem for the studied tile or one of the neighbouring tile of the studied tile, by default False
- **nb_tests** (*int, optional*) – number of time to test downloading the meteorological file if there is a connection problem, by default 5

toscana.**post_process**(*path_final_output_folder*, *grid_gpd*, *path_shapefiles*, *path_csv_folder*, *distance=-1.5*, *column_prefix='sol_'*, *statistics='mean'*, *bool_count=True*, *bool_sum=True*, *bool_mean=True*, *average=True*, *bool_global=True*)

Function to post process the results from iteration on the grid with the SEBE simulation.

The clip SEBE rasters are merged in one raster, this raster is clipped at the municipality extent Buffers are created for buildings (to not consider edges values when doing zonal statistics). Then zonal statistics are done to obtain an average annual irradiation value per building. The buildings shapefile with zonal statistics is post processed to remove the na values.

Parameters

- **path_final_output_folder** (*pathlib.Path*) – path of the folder with the final results (define in main function)
- **grid_gpd** (*GeoDataFrame*) – geopandas grid file (from define_grid)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **distance** (*float, optional*) – Size of the buffer (distance from the original edge to the final edge (Negative : remove size of the shapefile), by default -1.5

- **column_prefix** (*str, optional*) – prefix of the column that have been chosen to create the statistics, by default 'sol_'
- **statistics** (*str, optional*) – name of the suffix of the column where to verify that there is no nan value, by default 'mean'
- **bool_count** (*bool, optional*) – boolean value to obtain or not the number of pixel inside each shape in zonal statistics of merge SEBE raster, by default True
- **bool_sum** (*bool, optional*) – boolean value to obtain or not the sum of the pixel values inside each shape in zonal statistics of merge SEBE raster, by default True
- **bool_mean** (*bool, optional*) – boolean value to obtain or not the average of the pixel values inside each shape in zonal statistics of merge SEBE raster, by default True
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True
- **bool_global** (*bool, optional*) – boolean value to indicate if the diffuse and direct irradiance values were estimated or not from global irradiance values, by default True

toscana.**preprocess_municipality_buildings**(*path_shapefiles*, *path_municipalities_dep*, *village_INSEE_code*, *path_buildings_dep*)

Pre process the municipality and the building footprints.

It allows to obtain the buildings and municipality footprints of the studied municiaplity and reproject them in the good geographical reference system. Invalid features are also removed.

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_municipalities_dep** (*pathlib.Path*) – path of the municipalities footprints for the department of interest
- **village_INSEE_code** (*str*) – INSEE code of the municipality to study
- **path_buildings_dep** (*pathlib.Path*) – path of the shapefile with buildings from the selected departement

toscana.**preprocess_raster_file**(*path_raster_files*, *path_shapefiles*, *path_DEM*, *projection='IGNF:ETRS89LAEA'*, *extra_size=1000*, *resample_method=1*, *height_column='HAUTEUR'*, *pixel_resolution=1*)

Preprocess and create the raster files that are needed for the SEBE simulation.

A larger DEM than the grid extent is needed because of the resampling. The DEM is resampled in order to have a resolution of 1m. This resolution is needed to add the height of building to create the DSM. Only buildings with height above zero are kept for the analysis. A DHM is also created for the calculation of wall height and wall aspect.

Parameters

- **path_raster_files** (*pathlib.Path*) – path of the folder with temporary raster files (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_DEM** (*pathlib.Path*) – path of the raw input DEM, it is supposed to have 25m resolution and IGNF ETRS89LAEA projection

- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **extra_size** (*int, optional*) – extra width and height to add to the extent of the grid, by default 1000
- **resample_method** (*int, optional*) – Select the resampling method (1 = bilinear), by default 1 from 0 to 11, see gdal:warpreproject documentation
- **height_column** (*str, optional*) – column name with the height of buildings, by default "HAUTEUR"
- **pixel_resolution** (*int, optional*) – Select the resolution of the rasters, by default 1

Raises

- **AssertionError** – extra_size should be high enough to enable the resampling (extra_size < 4*resolution_x or extra_size <4*resolution_y)
- **AttributeError** – height_column not in the dataframe
- **AttributeError** – height_column must contain numbers (int or float)

toscana.**transform_days_into_period**(*start_day*, *start_month*, *end_day*, *end_month*)

Obtain a list with the day numbers of the year for a given period, defined with the date of the first day (day of the month and month of the year) and the date of the last day of the period (day of the month and month of the year).

Parameters

- **start_day** (*int*) – day number of the month of the first day of the period
- **start_month** (*int*) – month number of the year of the first day of the period
- **end_day** (*int*) – day number of the month of the last day of the period
- **end_month** (*int*) – month number of the year of the last day of the period

Returns

list_days – list with the day number of the year of all the days in the given period

Return type

list

CHAPTER

TWO

ALL FEATURES

2.1 toscana.data_post_processing package

2.1.1 Module contents

toscana.data_post_processing.**create_buffer**(*path_buildings*, *path_buildings_buffer*, *distance=-1.5*)

Create a (negative) buffer for building footprints (towards the center).

It is done in order to not consider the edges of the buildings (for zonal statistics). The default `distance` of the buffer : -1.5m.

Parameters

- **path_buildings** (*pathlib.Path*) – path of the shapefile with buildings, to which are applied the buffer (for example path_municipality_buildings_reproject_valid_sup_0 obtained in select_buildings_height_sup_0)
- **path_buildings_buffer** (*pathlib.Path*) – path to save the layer (buildings with buffer)
- **distance** (*float, optional*) – Size of the buffer (distance from the original edge to the final edge. Negative : remove size of the shapefile), by default -1.5

toscana.data_post_processing.**merge_SEBE_raster**(*grid_gdp*, *path_merge_SEBE_raster*, *path_final_output_folder*, *path_csv_folder*, *average=True*, *bool_global=True*)

Merge the raster files obtained from the SEBE simulation.

The function take the raster files in the different subfolders (one for each grid tile). Only the rasters obtained from the tile where the SEBE calculation could be done are merged. A verification on each raster files and on the merge raster is done to be sure that there is only values above zero in the merge raster.

Parameters

- **grid_gdp** (*GeoDataFrame*) – geopandas grid file (from *define_grid*)
- **path_merge_SEBE_raster** (*pathlib.Path*) – path where to save the merge raster file with SEBE results
- **path_final_output_folder** (*pathlib.Path*) – path of the folder with the final results
- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True

- **bool_global** (*bool, optional*) – boolean value to indicate if the diffuse and direct irradiance values were estimated or not from global irradiance values, by default True

toscana.data_post_processing.**merge_raster**(*list*, *path_merge_raster*)

Merge several raster files.

Parameters

- **list** (*list*) – list of the path of the raster files to merge
- **path_merge_raster** (*pathlib.Path*) – path where to save the merge raster layer

toscana.data_post_processing.**post_process**(*path_final_output_folder*, *grid_gpd*, *path_shapefiles*, *path_csv_folder*, *distance=-1.5*, *column_prefix='sol_'*, *statistics='mean'*, *bool_count=True*, *bool_sum=True*, *bool_mean=True*, *average=True*, *bool_global=True*)

Function to post process the results from iteration on the grid with the SEBE simulation.

The clip SEBE rasters are merged in one raster, this raster is clipped at the municipality extent Buffers are created for buildings (to not consider edges values when doing zonal statistics). Then zonal statistics are done to obtain an average annual irradiation value per building. The buildings shapefile with zonal statistics is post processed to remove the na values.

Parameters

- **path_final_output_folder** (*pathlib.Path*) – path of the folder with the final results (define in main function)
- **grid_gpd** (*GeoDataFrame*) – geopandas grid file (from define_grid)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **distance** (*float, optional*) – Size of the buffer (distance from the original edge to the final edge (Negative : remove size of the shapefile), by default -1.5
- **column_prefix** (*str, optional*) – prefix of the column that have been chosen to create the statistics, by default 'sol_'
- **statistics** (*str, optional*) – name of the suffix of the column where to verify that there is no nan value, by default 'mean'
- **bool_count** (*bool, optional*) – boolean value to obtain or not the number of pixel inside each shape in zonal statistics of merge SEBE raster, by default True
- **bool_sum** (*bool, optional*) – boolean value to obtain or not the sum of the pixel values inside each shape in zonal statistics of merge SEBE raster, by default True
- **bool_mean** (*bool, optional*) – boolean value to obtain or not the average of the pixel values inside each shape in zonal statistics of merge SEBE raster, by default True
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True
- **bool_global** (*bool, optional*) – boolean value to indicate if the diffuse and direct irradiance values were estimated or not from global irradiance values, by default True

2.2 toscana.data_pre_processing package

2.2.1 Submodules

2.2.2 toscana.data_pre_processing.grid_creation module

module to create a grid

toscana.data_pre_processing.grid_creation.**define_grid**(*extent_points*, *path_grid*, *projection='IGNF:ETRS89LAEA'*, *horizontal_spacing=1000*, *vertical_spacing=1000*)

Obtain the grid that will be used for the simulation.

It is a grid of 1000m (horizontal_spacing) per 1000m (vertical_spacing) per default.

Parameters

- **extent_points** (*str*) – string with the extent points of the commune extent (extents_point obtained in *obtain_municipality_extent*)
- **path_grid** (*pathlib.Path*) – path where to save the layer with the grid
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **horizontal_spacing** (*int, optional*) – horizontal size of grid tiles, by default 1000
- **vertical_spacing** (*int, optional*) – vertical size of grid tiles, by default 1000

Returns
grid_gpd – geopandas grid file

Return type
GeoDataFrame

Raises

- **AssertionError** – horizontal_spacing too large (compared to horizontal length of municipality extent)
- **AssertionError** – vertical_spacing too large (compared to vertical length of municipality extent)

toscana.data_pre_processing.grid_creation.**obtain_grid**(*path_shapefiles*, *projection='IGNF:ETRS89LAEA'*, *horizontal_spacing=1000*, *vertical_spacing=1000*)

Create the grid in order to run the SEBE simulation for the studied village.

It is a grid of 1000m (horizontal_spacing) per 1000m (vertical_spacing) per default (in IGNF ETRS89LAEA reference system by default, modify ``projection``to change it).

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **horizontal_spacing** (*int, optional*) – horizontal size of grid tiles, by default 1000

- **vertical_spacing** (*int, optional*) – vertical size of grid tiles, by default 1000

Returns
grid_gpd – geopandas grid file

Return type
GeoDataFrame

toscana.data_pre_processing.grid_creation.**obtain_municipality_extent**(*path_reproject_municipality_footprint*, *path_municipality_extent*, *projection='IGNF:ETRS89LAEA'*)

Obtain the rectangle extent of the municipality (from the municipality footprint) and obtain extents points (in IGNF ETRS89LAEA reference system by default, modify ``projection``to change it).

Parameters

- **path_reproject_municipality_footprint** (*pathlib.Path*) – path of the municipality footprint shapefile reproject in the new neference system (for example path_reproject_municipality_footprint obtained in *reproject_municipality_footprint*)
- **path_municipality_extent** (*pathlib.Path*) – path where to save the layer with the municipality extent shapefile
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"

Returns
extent_points – string with the extent points of the commune extent (format needed for qgis simulation)

Return type
str

2.2.3 toscana.data_pre_processing.meteorological_files module

toscana.data_pre_processing.meteorological_files.**create_monthly_weather_file**(*path_meteorological_folder*, *list_month*, *path_gdf_centroid*, *average=True*)

Obtain monthly meteorological files from annual meteorological files for given months. Subfolders to save the monthly weather files are created.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **list_month** (*list*) – list of the months for which monthly weather files will be generated
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Raises

- **TypeError** – month must be a int (or a string of numbers)
- **AssertionError** – month must be between 1 and 12

toscana.data_pre_processing.meteorological_files.**create_period_weather_file**(*name_folder_period*, *path_meteorological_folder*, *path_gdf_centroid*, *list_days*, *average=True*)

Obtain meteorological files for a given period from annual meteorological files. Subfolders to save the new weather files are created.

Parameters

- **name_folder_period** (*str*) – name of the folder where to save the meteorological files for the given period
- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **list_days** (*list*) – list of the day numbers of the year included in the given period
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Returns

path_period_weather_file – path of the folder with the created meteorological files for the given period

Return type

pathlib.Path

Raises

- **TypeError** – list_days must be a list of int (or a list of strings of numbers)
- **AssertionError** – day numbers must be between 1 and 365

toscana.data_pre_processing.meteorological_files.**create_winter_summer_month_weather_file**(*path_meteorologic*
path_gdf_centroid
list_winter_month=
list_summer_mont
av-
er-
age=True)

Obtain meteorological files for two periods from annual meteorological files. One of the period should correspond to winter months, and is defined by default from January to March and from November to December. The second period should correspond to summer months, and is defined by default as all other months from the winter months. Subfolders to save the new weather files are created.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **list_winter_month** (*list, optional*) – list of the winter months, by default []
- **list_summer_month** (*list, optional*) – list_of the summer months, by default []

- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Returns

- **path_folder_summer** (*pathlib.Path*) – path of the folder with the created summer meteorological files
- **path_folder_winter** (*pathlib.Path*) – path of the folder with the created winter meteorological files

Raises

- **TypeError** – list_winter_month and list_summer_month must be list of int (or list of strings of numbers)
- **AssertionError** – month must be between 1 and 12

toscana.data_pre_processing.meteorological_files.**obtain_meteorological_files**(*path_meteorological_folder*, *path_shapefiles*, *save_temp_file=True*, *average=True*, *option_no_average_if_one_problem=*, *option_no_average_for_the_error_ti*, *nb_tests=5*)

Obtain the meteorological files needed to run the SEBE simulations.

Folders to save the files are created. Centroids of the grid tiles are created. They are modified to include the 12 closest centroids. The metorological files are obtained in epw format, and then transform in txt format, needed for the simulations. These temporary files are saved only if `save_temp_file` is set to True. An average of the meteorological file from the tile and the 12 closest tiles is done if `average` is set to True (default value) (due to the low resolution of the meteorological data). If there is some meteorological files that could not be downloaded, the average is done only with the valid neighbouring tiles (default). `option_no_average_if_one_problem` and `option_no_average_for_the_error_tile` will be used only if `average` is set to True.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where to save all the meteorological files
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **save_temp_file** (*bool, optional*) – boolean value to save or not temporary meteorological files, by default True
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True
- **option_no_average_if_one_problem** (*bool, optional*) – boolean value to don't average any meteorological file if there is one problem for one tile or one of the neighbouring tile among all the tiles, by default False
- **option_no_average_for_the_error_tile** (*bool, optional*) – boolean value to don't average the meteorological file if there is one problem for the studied tile or one of the neighbouring tile of the studied tile, by default False
- **nb_tests** (*int, optional*) – number of time to test downloading the meteorological file if there is a connection problem, by default 5

toscana.data_pre_processing.meteorological_files.**transform_days_into_period**(*start_day*, *start_month*, *end_day*, *end_month*)

Obtain a list with the day numbers of the year for a given period, defined with the date of the first day (day of the month and month of the year) and the date of the last day of the period (day of the month and month of the year).

Parameters

- **start_day** (*int*) – day number of the month of the first day of the period
- **start_month** (*int*) – month number of the year of the first day of the period
- **end_day** (*int*) – day number of the month of the last day of the period
- **end_month** (*int*) – month number of the year of the last day of the period

Returns
list_days – list with the day number of the year of all the days in the given period

Return type
list

2.2.4 toscana.data_pre_processing.raster_preprocessing module

toscana.data_pre_processing.raster_preprocessing.**create_DHM**(*path_resample_DEM*, *path_buildings_sup_0*, *large_extent_grid*, *path_DHM_temp_1*, *path_DHM_temp_2*, *path_DHM*, *pixel_resolution=1*, *height_column='HAUTEUR'*)

Create the DHM (only building height above ground level) with a resolution of 1m (default).

First step : create a raster with zero value at the same size/extent than the DSM. Second step : create the DHM by adding building height. By default, the column name with the height of buildings is "HAUTEUR" (name of the column in BDTOPO database). Third step : replace the nodata values of the DHM by 0.

Parameters

- **path_resample_DEM** (*pathlib.Path*) – path of the DEM (path_resample_DEM from resample_DEM)
- **path_buildings_sup_0** (*pathlib.Path*) – path of the shapefile with buildings (and heights) (path_buildings_sup_0 from select_buildings_height_sup_0)
- **large_extent_grid** (*str*) – string with the large extent of the grid (from obtain_large_grid_extent)
- **path_DHM_temp_1** (*pathlib.Path*) – path where to save a temporary layer used to create the final DHM
- **path_DHM_temp_2** (*pathlib.Path*) – path where to save a second temporary layer used to create the final DHM
- **path_DHM** (*pathlib.Path*) – path where to save the created DHM
- **pixel_resolution** (*int, optional*) – Resolution of the DHM, by default 1
- **height_column** (*str, optional*) – column name with the height of buildings, by default "HAUTEUR"

toscana.data_pre_processing.raster_preprocessing.**create_DSM**(*path_resample_DEM*, *path_buildings_sup_0*, *large_extent_grid*, *path_DSM*, *path_DSM_temp*, *pixel_resolution=1*, *height_column='HAUTEUR'*)

Create the DSM from the DEM (resample at 1m resolution (default)).

By default, the column name with the height of buildings is "HAUTEUR" (name of the column in BDTOPO database). The nodata values of the DSM are replaced by 0.

Parameters

- **path_resample_DEM** (*pathlib.Path*) – path of the DEM (path_resample_DEM from resample_DEM)
- **path_buildings_sup_0** (*pathlib.Path*) – path of the shapefile with buildings (and heights) (path_buildings_sup_0 from select_buildings_height_sup_0)
- **large_extent_grid** (*str*) – string with the large extent of the grid (from obtain_large_grid_extent)
- **path_DSM** (*pathlib.Path*) – path where to save the layer with the created DSM
- **path_DSM_temp** (*pathlib.Path*) – path where to save a temporary layer used to create the final DSM
- **pixel_resolution** (*int, optional*) – Resolution of the DSM, by default 1
- **height_column** (*str, optional*) – column name with the height of buildings, by default "HAUTEUR"

toscana.data_pre_processing.raster_preprocessing.**download_extract_and_merge_DEM_from_OpenDEM**(*path_raw_da*
path_shapefil
bool_France=
thresh-
old=100)

Download and extract the DEM raster tile from the OpenDEM website according to the municipality footprint extent.

The raster tiles are then merged together to have a unique merge DEM. First, the edge tile are obtained. If the municipality footprint extent is close to the border of the tile, the neighbouring tile is also download. All the tiles that overlap with the extent of municipality footprint are downloaded and then merge together. If the studied territory is not located in France, `bool_France` should be set to False.

Parameters

- **path_raw_data_folder** (*pathlib.Path*) – path of the folder with raw data (downloaded) files (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **bool_France** (*bool, optional*) – boolean value to indicate or not if the territory of interest is located in France or not, by default True
- **threshold** (*int, optional*) – minimal value (in meters) of the difference between the municipality extent and the border of the raster tile, by default 100

Returns

path_merge_dem – path where is saved the merged DEM

Return type
pathlib.Path

toscana.data_pre_processing.raster_preprocessing.**preprocess_raster_file**(*path_raster_files*, *path_shapefiles*, *path_DEM*, *projection='IGNF:ETRS89LAEA'*, *extra_size=1000*, *resample_method=1*, *height_column='HAUTEUR'*, *pixel_resolution=1*)

Preprocess and create the raster files that are needed for the SEBE simulation.

A larger DEM than the grid extent is needed because of the resampling. The DEM is resampled in order to have a resolution of 1m. This resolution is needed to add the height of building to create the DSM. Only buildings with height above zero are kept for the analysis. A DHM is also created for the calculation of wall height and wall aspect.

Parameters

- **path_raster_files** (*pathlib.Path*) – path of the folder with temporary raster files (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_DEM** (*pathlib.Path*) – path of the raw input DEM, it is supposed to have 25m resolution and IGNF ETRS89LAEA projection
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **extra_size** (*int, optional*) – extra width and height to add to the extent of the grid, by default 1000
- **resample_method** (*int, optional*) – Select the resampling method (1 = bilinear), by default 1 from 0 to 11, see gdal:warpreproject documentation
- **height_column** (*str, optional*) – column name with the height of buildings, by default "HAUTEUR"
- **pixel_resolution** (*int, optional*) – Select the resolution of the rasters, by default 1

Raises

- **AssertionError** – extra_size should be high enough to enable the resampling (extra_size < 4*resolution_x or extra_size <4*resolution_y)
- **AttributeError** – height_column not in the dataframe
- **AttributeError** – height_column must contain numbers (int or float)

toscana.data_pre_processing.raster_preprocessing.**resample_DEM**(*path_clip_large_DEM*, *path_resample_DEM*, *large_extent_grid*, *resample_method=1*, *resample_resolution=1*)

Resample the original DEM to obtain a DEM of 1m resolution (default), default option is bilinear resample.

Parameters

- **path_clip_large_DEM** (*pathlib.Path*) – path with the DEM to resample (path_output from clip_DEM_large)
- **path_resample_DEM** (*pathlib.Path*) – path where to save the layer with the resampled DEM
- **large_extent_grid** (*str*) – string with the large extent of the grid, from obtain_large_grid_extent
- **resample_method** (*int, optional*) – Select the resampling method (1= bilinear) , by default 1 from 0 to 11, see gdal:warpreproject documentation
- **resample_resolution** (*int, optional*) – Select the resampling resolution, by default 1

Raises

AssertionError – resample_method must be between 0 and 11 (available resampling method)

toscana.data_pre_processing.raster_preprocessing.**select_buildings_height_sup_0**(*path_reproject_municipality_bu*
path_buildings_sup_0,
height_column='HAUTEUR')

Select only buildings with a height above 0 (necessary to create DSM).

By default, the column name with the height of buildings is "HAUTEUR" (name of the column in BDTOPO database).

Parameters

- **path_reproject_municipality_buildings** (*pathlib.Path*) – path of the shapefile layer with the buildings (path_reproject_municipality_buildings from reproject buildings/check validity)
- **path_buildings_sup_0** (*pathlib.Path*) – path where to save the layer with the shapefile with only buildings with a height above 0
- **height_column** (*str, optional*) – column name with the height of buildings, by default "HAUTEUR"

2.2.5 toscana.data_pre_processing.shp_preprocessing module

module to pre-process shapefiles

toscana.data_pre_processing.shp_preprocessing.**check_validity**(*path_input_shapefiles*,
path_valid_shapefiles)

Verify the validity of buildings footprints (remove invalid features).

Parameters

- **path_input_shapefiles** (*pathlib.Path*) – path of the shapefile to verify (path_reproject_municipality_buildings obtained in *reproject_buildings* for example)
- **path_valid_shapefiles** (*pathlib.Path*) – path where to save the shapefile layer with only valid features

toscana.data_pre_processing.shp_preprocessing.**download_and_extract_BDTOPO_data**(*departement*,
path_raw_data_folder)

Download shapefiles (municipalities footprints and buildings footprints) from the BDTOPO database on the IGN website giving a the number of the `departement`.

A packed folder is downloaded and then extracted. The version downloaded is the 15-12-2023 version.

Parameters

- **departement** (*str*) – departement the municipality
- **path_raw_data_folder** (*pathlib.Path*) – path of the folder with raw data (downloaded) files (define in main function)

Returns

- **path_departement_municipalities_footprint** (*pathlib.Path*) – path of the municipalities footprints for the department of interest
- **path_input_departement_building** (*pathlib.Path*) – path of the buildings footprints for the department of interest

toscana.data_pre_processing.shp_preprocessing.**obtain_municipality_buildings**(*path_input_departement_building*, *path_input_municipality_footprint*, *path_municipality_buildings*)

Obtain the buildings of the selected municipality from the buildings of the departement of interest.

Parameters

- **path_input_departement_building** (*pathlib.Path*) – path of the shapefile with buildings from the selected departement
- **path_input_municipality_footprint** (*pathlib.Path*) – path of the municipality footprint (path_output_municipality_footprint obtained in *obtain_municipality_footprint*)
- **path_municipality_buildings** (*pathlib.Path*) – path where to save the layer with buildings of the municipality

toscana.data_pre_processing.shp_preprocessing.**obtain_municipality_footprint**(*path_departement_municipalities_f*
path_output_municipality_footprint
INSEE_code)

Obtain the municipality footprint of the selected municipality among all the municipalities of the selected departement using the INSEE_code.

Parameters

- **path_departement_municipalities_footprint** (*pathlib.Path*) – path of the municipalities footprint for the department of interest
- **path_output_municipality_footprint** (*pathlib.Path*) – path where to save the layer with the municipality footprint of the selected municipality
- **INSEE_code** (*str*) – INSEE code of the municipality to study

toscana.data_pre_processing.shp_preprocessing.**preprocess_municipality_buildings**(*path_shapefiles*, *path_municipalities_dep*, *village_INSEE_code*, *path_buildings_dep*)

Pre process the municipality and the building footprints.

It allows to obtain the buildings and municipality footprints of the studied municiaplity and reproject them in the good geographical reference system. Invalid features are also removed.

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)

- **path_municipalities_dep** (*pathlib.Path*) – path of the municipalities footprints for the department of interest
- **village_INSEE_code** (*str*) – INSEE code of the municipality to study
- **path_buildings_dep** (*pathlib.Path*) – path of the shapefile with buildings from the selected departement

toscana.data_pre_processing.shp_preprocessing.**reproject_shapefiles_2154_to_IGNF**(*path_shapefiles_2154*, *path_shapefiles_IGNF*)

Reproject shapefiles from EPSG 2154 reference system into IGNF : ETRS89LAEA reference system.

Parameters

- **path_shapefiles_2154** (*pathlib.Path*) – path of the shapefile in the reference system EPSG:2154 (path_output_municipality_footprint obtained in *obtain_municipality_footprint or path_municipality_buildings from obtain_municipality_buildings*)
- **path_shapefiles_IGNF** (*pathlib.Path*) – path where to save the layer (shapefile reproject in IGNF: ETRS89LAEA reference system)

2.2.6 Module contents

toscana.data_pre_processing.**check_validity**(*path_input_shapefiles*, *path_valid_shapefiles*)

Verify the validity of buildings footprints (remove invalid features).

Parameters

- **path_input_shapefiles** (*pathlib.Path*) – path of the shapefile to verify (path_reproject_municipality_buildings obtained in *reproject_buildings* for example)
- **path_valid_shapefiles** (*pathlib.Path*) – path where to save the shapefile layer with only valid features

toscana.data_pre_processing.**create_monthly_weather_file**(*path_meteorological_folder*, *list_month*, *path_gdf_centroid*, *average=True*)

Obtain monthly meteorological files from annual meteorological files for given months. Subfolders to save the monthly weather files are created.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **list_month** (*list*) – list of the months for which monthly weather files will be generated
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Raises

- **TypeError** – month must be a int (or a string of numbers)
- **AssertionError** – month must be between 1 and 12

toscana.data_pre_processing.**create_period_weather_file**(*name_folder_period*, *path_meteorological_folder*, *path_gdf_centroid*, *list_days*, *average=True*)

Obtain meteorological files for a given period from annual meteorological files. Subfolders to save the new weather files are created.

Parameters

- **name_folder_period** (*str*) – name of the folder where to save the meteorological files for the given period
- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **list_days** (*list*) – list of the day numbers of the year included in the given period
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Returns

path_period_weather_file – path of the folder with the created meteorological files for the given period

Return type

pathlib.Path

Raises

- **TypeError** – list_days must be a list of int (or a list of strings of numbers)
- **AssertionError** – day numbers must be between 1 and 365

toscana.data_pre_processing.**create_winter_summer_month_weather_file**(*path_meteorological_folder*, *path_gdf_centroid*, *list_winter_month=[]*, *list_summer_month=[]*, *average=True*)

Obtain meteorological files for two periods from annual meteorological files. One of the period should correspond to winter months, and is defined by default from January to March and from November to December. The second period should correspond to summer months, and is defined by default as all other months from the winter months. Subfolders to save the new weather files are created.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where ared saved the initial meteorological files
- **path_gdf_centroid** (*pathlib.Path*) – path of the geopandas file with the centroid of the actual grid (obtained in *create_centroid*)
- **list_winter_month** (*list, optional*) – list of the winter months, by default []
- **list_summer_month** (*list, optional*) – list_of the summer months, by default []
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True

Returns

- **path_folder_summer** (*pathlib.Path*) – path of the folder with the created summer meteorological files

- **path_folder_winter** (*pathlib.Path*) – path of the folder with the created winter meteorological files

Raises

- **TypeError** – list_winter_month and list_summer_month must be list of int (or list of strings of numbers)
- **AssertionError** – month must be between 1 and 12

toscana.data_pre_processing.**define_grid**(*extent_points*, *path_grid*, *projection='IGNF:ETRS89LAEA'*, *horizontal_spacing=1000*, *vertical_spacing=1000*)

Obtain the grid that will be used for the simulation.

It is a grid of 1000m (`horizontal_spacing`) per 1000m (`vertical_spacing`) per default.

Parameters

- **extent_points** (*str*) – string with the extent points of the commune extent (extents_point obtained in *obtain_municipality_extent*)
- **path_grid** (*pathlib.Path*) – path where to save the layer with the grid
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **horizontal_spacing** (*int, optional*) – horizontal size of grid tiles, by default 1000
- **vertical_spacing** (*int, optional*) – vertical size of grid tiles, by default 1000

Returns
grid_gpd – geopandas grid file

Return type
GeoDataFrame

Raises

- **AssertionError** – horizontal_spacing too large (compared to horizontal length of municipality extent)
- **AssertionError** – vertical_spacing too large (compared to vertical length of municipality extent)

toscana.data_pre_processing.**download_and_extract_BDTOPO_data**(*departement*, *path_raw_data_folder*)

Download shapefiles (municipalities footprints and buildings footprints) from the BDTOPO database on the IGN website giving a the number of the `departement`.

A packed folder is downloaded and then extracted. The version downloaded is the 15-12-2023 version.

Parameters

- **departement** (*str*) – departement the municipality
- **path_raw_data_folder** (*pathlib.Path*) – path of the folder with raw data (downloaded) files (define in main function)

Returns

- **path_departement_municipalities_footprint** (*pathlib.Path*) – path of the municipalities footprints for the department of interest
- **path_input_departement_building** (*pathlib.Path*) – path of the buildings footprints for the department of interest

toscana.data_pre_processing.**download_extract_and_merge_DEM_from_OpenDEM**(*path_raw_data_folder*, *path_shapefiles*, *bool_France=True*, *threshold=100*)

Download and extract the DEM raster tile from the OpenDEM website according to the municipality footprint extent.

The raster tiles are then merged together to have a unique merge DEM. First, the edge tile are obtained. If the municipality footprint extent is close to the border of the tile, the neighbouring tile is also download. All the tiles that overlap with the extent of municipality footprint are downloaded and then merge together. If the studied territory is not located in France, `bool_France` should be set to False.

Parameters

- **path_raw_data_folder** (*pathlib.Path*) – path of the folder with raw data (downloaded) files (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **bool_France** (*bool, optional*) – boolean value to indicate or not if the territory of interest is located in France or not, by default True
- **threshold** (*int, optional*) – minimal value (in meters) of the difference between the municipality extent and the border of the raster tile, by default 100

Returns
path_merge_dem – path where is saved the merged DEM

Return type
pathlib.Path

toscana.data_pre_processing.**obtain_grid**(*path_shapefiles*, *projection='IGNF:ETRS89LAEA'*, *horizontal_spacing=1000*, *vertical_spacing=1000*)

Create the grid in order to run the SEBE simulation for the studied village.

It is a grid of 1000m (`horizontal_spacing`) per 1000m (`vertical_spacing`) per default (in IGNF ETRS89LAEA reference system by default, modify ``projection``to change it).

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"
- **horizontal_spacing** (*int, optional*) – horizontal size of grid tiles, by default 1000
- **vertical_spacing** (*int, optional*) – vertical size of grid tiles, by default 1000

Returns
grid_gpd – geopandas grid file

Return type
GeoDataFrame

toscana.data_pre_processing.**obtain_meteorological_files**(*path_meteorological_folder*, *path_shapefiles*, *save_temp_file=True*, *average=True*, *option_no_average_if_one_problem=False*, *option_no_average_for_the_error_tile=False*, *nb_tests=5*)

Obtain the meteorological files needed to run the SEBE simulations.

Folders to save the files are created. Centroids of the grid tiles are created. They are modified to include the 12 closest centroids. The metorological files are obtained in epw format, and then transform in txt format, needed for the simulations. These temporary files are saved only if `save_temp_file` is set to True. An average of the meteorological file from the tile and the 12 closest tiles is done if `average` is set to True (default value) (due to the low resolution of the meteorological data). If there is some meteorological files that could not be downloaded, the average is done only with the valid neighbouring tiles (default). `option_no_average_if_one_problem` and `option_no_average_for_the_error_tile` will be used only if `average` is set to True.

Parameters

- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where to save all the meteorological files
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **save_temp_file** (*bool, optional*) – boolean value to save or not temporary meteorological files, by default True
- **average** (*bool, optional*) – boolean value to do or not an average of the meteorological files, by default True
- **option_no_average_if_one_problem** (*bool, optional*) – boolean value to don't average any meteorological file if there is one problem for one tile or one of the neighbouring tile among all the tiles, by default False
- **option_no_average_for_the_error_tile** (*bool, optional*) – boolean value to don't average the meteorological file if there is one problem for the studied tile or one of the neighbouring tile of the studied tile, by default False
- **nb_tests** (*int, optional*) – number of time to test downloading the meteorological file if there is a connection problem, by default 5

toscana.data_pre_processing.**obtain_municipality_buildings**(*path_input_departement_building*, *path_input_municipality_footprint*, *path_municipality_buildings*)

Obtain the buildings of the selected municipality from the buildings of the departement of interest.

Parameters

- **path_input_departement_building** (*pathlib.Path*) – path of the shapefile with buildings from the selected departement
- **path_input_municipality_footprint** (*pathlib.Path*) – path of the municipality footprint (path_output_municipality_footprint obtained in *obtain_municipality_footprint*)
- **path_municipality_buildings** (*pathlib.Path*) – path where to save the layer with buildings of the municipality

toscana.data_pre_processing.**obtain_municipality_extent**(*path_reproject_municipality_footprint*, *path_municipality_extent*, *projection='IGNF:ETRS89LAEA'*)

Obtain the rectangle extent of the municipality (from the municipality footprint) and obtain extents points (in IGNF ETRS89LAEA reference system by default, modify ``projection``to change it).

Parameters

- **path_reproject_municipality_footprint** (*pathlib.Path*) – path of the municipality footprint shapefile reproject in the new neference system (for example path_reproject_municipality_footprint obtained in *reproject_municipality_footprint*)
- **path_municipality_extent** (*pathlib.Path*) – path where to save the layer with the municipality extent shapefile
- **projection** (*str, optional*) – name of the reference system of the grid, by default "IGNF:ETRS89LAEA"

Returns

extent_points – string with the extent points of the commune extent (format needed for qgis simulation)

Return type

str

toscana.data_pre_processing.**obtain_municipality_footprint**(*path_departement_municipalities_footprint*, *path_output_municipality_footprint*, *INSEE_code*)

Obtain the municipality footprint of the selected municipality among all the municipalities of the selected departement using the INSEE_code.

Parameters

- **path_departement_municipalities_footprint** (*pathlib.Path*) – path of the municipalities footprint for the department of interest
- **path_output_municipality_footprint** (*pathlib.Path*) – path where to save the layer with the municipality footprint of the selected municipality
- **INSEE_code** (*str*) – INSEE code of the municipality to study

toscana.data_pre_processing.**preprocess_municipality_buildings**(*path_shapefiles*, *path_municipalities_dep*, *village_INSEE_code*, *path_buildings_dep*)

Pre process the municipality and the building footprints.

It allows to obtain the buildings and municipality footprints of the studied municiaplity and reproject them in the good geographical reference system. Invalid features are also removed.

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_municipalities_dep** (*pathlib.Path*) – path of the municipalities footprints for the department of interest
- **village_INSEE_code** (*str*) – INSEE code of the municipality to study
- **path_buildings_dep** (*pathlib.Path*) – path of the shapefile with buildings from the selected departement

toscana.data_pre_processing.**reproject_shapefiles_2154_to_IGNF**(*path_shapefiles_2154*, *path_shapefiles_IGNF*)

Reproject shapefiles from EPSG 2154 reference system into IGNF : ETRS89LAEA reference system.

Parameters

- **path_shapefiles_2154** (*pathlib.Path*) – path of the shapefile in the reference system EPSG:2154 (path_output_municipality_footprint obtained in *obtain_municipality_footprint or path_municipality_buildings from obtain_municipality_buildings*)
- **path_shapefiles_IGNF** (*pathlib.Path*) – path where to save the layer (shapefile reproject in IGNF: ETRS89LAEA reference system)

toscana.data_pre_processing.**transform_days_into_period**(*start_day*, *start_month*, *end_day*, *end_month*)

Obtain a list with the day numbers of the year for a given period, defined with the date of the first day (day of the month and month of the year) and the date of the last day of the period (day of the month and month of the year).

Parameters

- **start_day** (*int*) – day number of the month of the first day of the period
- **start_month** (*int*) – month number of the year of the first day of the period
- **end_day** (*int*) – day number of the month of the last day of the period
- **end_month** (*int*) – month number of the year of the last day of the period

Returns
list_days – list with the day number of the year of all the days in the given period

Return type
list

2.3 toscana.results package

2.3.1 Submodules

2.3.2 toscana.results.results_vizualisation module

toscana.results.results_vizualisation.**display_SEBE_raster**(*path_raster*, *rectangle=False*, *bool_title_SEBE=False*, *title_SEBE="Map of the municipality's average \nannual irradiation on surfaces"*)

Display in a figure the raster of the merge SEBE raster with a Red Yellow Blue colorscale.

path_raster
[pathlib.Path] path of the raster to display, made for the merge raster with SEBE results (path_merge_SEBE_raster obtained in *merge_SEBE_raster* or path_clip_raster obtained in *clip_raster*)

rectangle
[bool, optional] boolean value to display or not the borders (recommended to set to False if the raster is a raster coming from *clip_raster function*), by default False

bool_title_SEBE
[bool, optional] boolean value to display or not the title, by default False

title_SEBE
[str, optional] title of the figure, by default f"Map of the municipality's average annual irradiation on surfaces"

toscana.results.results_vizualisation.**display_distribution**(*path_buildings_zonal_stats*, *column_name*, *path_final_output_folder*, *bool_johnsonsu=True*, *save_plot=True*, *name_plot='Distribution_irradiation.png'*, *bool_title_histo=False*, *title_histo='Distribution of average annual rooftop irradiation'*)

Display in a figure the histogram of a feature with values included in a shapefile (in the column named `column_name`), with optionally the Johnson's SU fitted distribution (default : True).

It is made to display the average annual irradiation received per building rooftop. The values are first stored in a csv file, then histogram and the Johnson's SU fit are calculated. Then, the histogram is displayed.

Parameters

- **path_buildings_zonal_stats** (*pathlib.Path*) – path of the shapefile containing the data (zonal statistics for example) to display in a histogram (path_buildings_buffer_zonal_stat_post_process obtained in *_post_process_buffer_zonal_stat* for example)
- **column_name** (*str*) – name of the column in the shapefile containing the values to display in the histogram (sol_mean for example)
- **path_final_output_folder** (*pathlib.Path*) – path of the folder where to save the final results (define in main function)
- **bool_johnsonsu** (*bool, optional*) – boolean value to display or not the Johnson's SU fitted distribution, by default True
- **save_plot** (*bool, optional*) – boolean value to save or not a figure with the distribution, by default True
- **name_plot** (*str, optional*) – name of the file saved with the figure of the distribution, by default "Distribution_irradiation.png
- **bool_title_histo** (*bool, optional*) – boolean value to display or not the title, by default False
- **title_histo** (*str, optional*) – title of the figure, by default "Distribution of average annual rooftop irradiation"

toscana.results.results_vizualisation.**display_elevation_file**(*path_raster*, *rectangle=True*, *bool_title_elevation=False*, *title_elevation="Map of the municipality's elevation"*)

Display in a figure the raster of an elevation file (DSM, DEM for example) with a grey scale.

Parameters

- **path_raster** (*pathlib.Path*) – path of the raster file to display (path_DSM obtained in *create_DSM* or path_DHM obtained in *create_DHM* for example)
- **rectangle** (*bool, optional*) – boolean value to display or not the borders (recommended to set to False if the raster is a raster coming from *clip_raster* function), by default True

- **bool_title_elevation** (*bool, optional*) – boolean value to display or not the title, by default False
- **title_elevation** (*str, optional*) – title of the figure, by default "Map of the municipality's elevation"

toscana.results.results_vizualisation.**display_histogram**(*data*, *nb_bins*, *x*, *fitting_parameters*, *R2*, *path_final_output_folder*, *bool_johnsonsu=True*, *save_plot=True*, *name_plot='Distribution_irradiation.png'*, *bool_title_histo=False*, *title_histo='Distribution of average annual rooftop irradiation'*)

Display the histogram with optionally the Johnson's SU distribution fitted to the distribution (default : True) (inputs coming from *calculate_histogram_and_johnsonsu_fit*)

The figure is saved in an output folder.

Parameters

- **data** (*DataFrame*) – dataframe with irradiation values
- **nb_bins** (*int*) – number of bins in the histogram
- **x** (*list*) – list of the x value used to display the Johnson's SU distribution
- **fitting_parameters** (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter
- **R2** (*float*) – best R2 coefficient between classic (init) or floc=xmax (xmax) methods
- **path_final_output_folder** (*pathlib.Path*) – path of the folder where to save the final results (define in main function)
- **bool_johnsonsu** (*bool, optional*) – boolean value to display or not the Johnson's SU fitted distribution, by default True
- **save_plot** (*bool, optional*) – boolean value to save or not a figure with the distribution, by default True
- **name_plot** (*str, optional*) – name of the file saved with the figure of the distribution, by default "Distribution_irradiation.png"
- **bool_title_histo** (*bool, optional*) – boolean value to display or not the title, by default False
- **title_histo** (*str, optional*) – title of the figure, by default "Distribution of average annual rooftop irradiation"

toscana.results.results_vizualisation.**display_results**(*path_raster_files*, *path_final_output_folder*, *rectangle_elevation=True*, *rectangle_SEBE=False*, *bool_johnsonsu=True*, *column_prefix='sol_'*, *statistics='mean'*, *save_plot=True*, *name_plot='Distribution_irradiation.png'*, *bool_title_histo=False*, *title_histo='Distribution of average \nannual rooftop irradiation'*, *bool_title_elevation=False*, *title_elevation="Map of the municipality's elevation"*, *bool_title_SEBE=False*, *title_SEBE="Map of the municipality's average \nannual irradiation on surfaces"*, *bool_title_stats=False*, *title_stats="Map of the municipality's average annual rooftop\nirradiation per building"*, *label_stats='Irradiation (kWh/m^2)'*)

Display the results : the resample DSM of the village, the raster with SEBE results, the shapefile with building footprints (with buffer) and their average annual irradiation on rooftops and the histogram with the average annual irradiation received by rooftops and the associated Johnson's SU fitted distribution.

path_raster_files
[pathlib.Path] path of the folder with temporary raster files (define in main function)

path_final_output_folder
[pathlib.Path] path of the folder with the final results (define in main function)

rectangle_elevation
[bool, optional] boolean value to display or not the borders of the elevation file (recommended to set to False if the raster is a raster coming from *clip_raster* function), by default True

rectangle_SEBE
[bool, optional] boolean value to display or not the borders of the merge SEBE raster (recommended to set to False if the raster is a raster coming from *clip_raster* function), by default False

bool_johnsonsu
[bool, optional] boolean value to display or not the Johnson's SU fitted distribution, by default True

column_prefix
[str, optional] prefix of the column that have been chosen to create the statistics, by default 'sol_'

statistics
[str, optional] name of the suffix of the column to display, by default 'mean'

save_plot
[bool, optional] boolean value to save or not a figure with the distribution, by default True

name_plot
[str, optional] name of the file saved with the figure of the distribution, by default "Distribution_irradiation.png

bool_title_histo
[bool, optional] boolean value to display or not the title, by default False

title_histo
[str, optional] title of the figure, by default "Distribution of average annual rooftop irradiation"

bool_title_elevation
[bool, optional] boolean value to display or not the title, by default False

title_elevation
[str, optional] title of the figure, by default "Map of the municipality's elevation"

bool_title_SEBE
[bool, optional] boolean value to display or not the title, by default False

title_SEBE
[_type_, optional] title of the figure, by default f"Map of the municipality's average

annual irradiation on surfaces"

bool_title_stats
[bool, optional] boolean value to display or not the title, by default False

title_stats
[_type_, optional] title of the figure, by default f"Map of the municipality's average annual rooftop

irradiation per building"

label_stats
[str, optional] lalbel of the colorbar, by default "Irradiation (kWh/m^2)"

toscana.results.results_vizualisation.**display_zonal_stat_shapefile**(*path_shapefile*, *column_name*, *bool_title_stats=False*, *title_stats="Map of the municipality's average annual rooftop\nirradiation per building"*, *label_stats='Irradiation (kWh/m^2)'*)

Display in a figure a shapefile containing data (in the column `column_name`) to display with a Orange Red colorscale.

It displays the value if the cursor is put on a shape (a building footprint for example). It is made to be used with a building footprints shapefile with average annual solar irradiation received by buildings (`column_name` sol_mean for example).

path_shapefile
[pathlib.Path] path of the shapefile with the data (zonal statistics) to display (path_buildings_buffer_zonal_stat_post_process obtained in *_post_process_buffer_zonal_stat* for example)

column_name
[str] name of the column in the shapefile containing the values to display

bool_title_stats
[bool, optional] boolean value to display or not the title, by default False

title_stats
[_type_, optional] title of the figure, by default f"Map of the municipality's average annual rooftop

irradiation per building"

label_stats
[str, optional] lalbel of the colorbar, by default "Irradiation (kWh/m^2)"

AttributeError
name of the column (`column_name`) not found in the dataframe if `column_name` is too long (>10 characters), the name was probably shortened before saving

toscana.results.results_vizualisation.**generate_irradiation_csv_file**(*path_buildings_zonal_stats*, *column_name*, *path_final_output_folder*)

Generate a csv file with data coming from a shapefile contained in a column named `column_name`.

The na value are filled with 0. It is made to generate a csv file with irradiation value (annual irradiation received by building footprints).

Parameters

- **path_buildings_zonal_stats** (*pathlib.Path*) – path of the shapefile containing the data (zonal statistics for example) to export in a csv file (path_buildings_buffer_zonal_stat_post_process obtained in *_post_process_buffer_zonal_stat* for example)
- **column_name** (*str*) – name of the column in the shapefile containing the values to export (sol_mean for example)
- **path_final_output_folder** (*pathlib.Path*) – path of the folder where to save the final results (define in main function)

Returns
path_irradiation_csv – path of the csv file with data exported from the shapefile

Return type
pathlib.Path

Raises
AttributeError – name of the column (`column_name`) not found in the dataframe if `column_name` is too long (>10 characters), the name was probably shortened before saving

2.3.3 toscana.results.village_characteristics module

toscana.results.village_characteristics.**calculate_DFI**(*grid_gpd*, *path_average_folder*, *fn_average_files*, *path_csv_folder*)

Calculate the Diffuse Fraction Index (DFI).

It is calculated using the meteorological files (average) used for the simulation. For each tile of the grid, the diffuse irradiance value per hour (for hours between 10:00 and 15:00 (10 a.m. and 3 p.m.)) is summed. The same is done for the global irradiance. For each tile of the grid, the ratio is obtained. The average ratio over all grid tiles is the DFI. The tile of the grid for which the meteorological file could not have been downloaded are not considered in the calculation of the DFI.

Parameters

- **grid_gpd** (*GeoDataFrame*) – geopandas grid file
- **path_average_folder** (*pathlib.Path*) – path of the folder where are saved the meteorological txt files (average txt files for example)
- **fn_average_files** (*str*) – prefix name given to the txt files (average txt files for example)
- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)

Returns
DFI – Diffuse Fraction Index (DFI) calculated for the municipality

Return type
float

toscana.results.village_characteristics.**calculate_SVI**(*path_buildings_buffer_zonal_stat_post_process*, *path_shapefiles*, *path_csv_folder*, *nb_tests=5*)

Calculate the Sky View Index (SVI).

First, the centroid of buildings are obtained with their coordinates. It uses the centroid of buildings and horizon files from PVGIS API Then, these coordinates are used in the PVGIS API to obtain horizon files. The horizon files are then used to obtain the horizon profile at a specified place considering local surrounding topography and compared it with the horizon profile of a flat region (no surrounding topography). These horizon profiles are used to calculate the area of visible sky in both case. The ratio is obtained for all the considered buildings of the municipality. The average ratio over all the considered building is the SVI. If horizon profile is not available for one building, it moves to the next one, and the average is done only one the valid buildings. It saves in a csv file the SVI for each building or the error if the horizon profile was not available for this building.

Parameters

- **path_buildings_buffer_zonal_stat_post_process** (*pathlib.Path*) – path of shapefile with building footprints for which the SVI is wanted (path_buildings_buffer_zonal_stat_post_process from _post_process_buffer_zonal_stat for example)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **nb_tests** (*int, optional*) – number of time to test downloading the horizon file if there is a connection problem, by default 5

Returns
SVI – Sky View Index (SVI) calculated for the municipality

Return type
float

toscana.results.village_characteristics.**calculate_area_and_perimeter**(*path_shapefile*, *path_shapefile_perimeter_and_area*)

Calcualte the area and perimeter of a shapefile.

Parameters

- **path_shapefile** (*pathlib.Path*) – path of the shapefile for which the perimeter and area are wanted
- **path_shapefile_perimeter_and_area** (*pathlib.Path*) – path where to save the shapefile layer with perimeter and area values

toscana.results.village_characteristics.**calculate_village_characteristics**(*path_shapefiles*, *path_csv_folder*, *path_reproject_municipality_footprint*, *path_DEM_resample*, *population_column='POPULATION'*)

Calculate the area, the perimeter, the mean altitude of a municipality footprint, as well as the longitude and the latitude of the center of the village.

Zonal statistics with the resample DEM and the municipality footprint is used to obtain the mean altitude. The centroid of the municipality footprint is obtained to then obtain the coordinates of this centroid. Get the number of inhabitants of the municipality in the column `population_column` (default : "POPULATION") and return a nan if not available.

Parameters

- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **path_reproject_municipality_footprint** (*pathlib.Path*) – path of the municipality footprint shapefile reproject in the IGNF ETRS89LAEA reference system (for example path_reproject_municipality_footprint from reproject_municipality_footprint)
- **path_DEM_resample** (*pathlib.Path*) – path of the resampled DEM raster file (for example path_resample_DEM obtained in resample_DEM)
- **population_column** (*str, optional*) – name of the column with number of inhabitants of the studied municipality in the municipality footprint shapefile, by default "POPULATION"

Returns

- **area** (*float*) – area of the municipality footprint
- **perimeter** (*float*) – perimeter of the municipality footprint
- **mean_altitude** (*float*) – mean altitude value of the municipality footprint
- **latitude_center** (*float*) – latitude of the center of the municipality
- **longitude_center** (*float*) – longitude of the center of the municipality
- **population** (*int*) – population of the municipality (informed by IGN)

toscana.results.village_characteristics.**calculate_village_distribution_characteristics**(*path_csv_folder*, *path_raster_files*, *path_final_output_fol*, *path_shapefiles*, *grid_gpd*, *path_meteorological_*, *village_name*, *village_INSEE_code=N*, *village_departement=N*, *average=True*, *population_column='POPU*, *nb_tests=5*)

Calculate and gather in a dataframe the village characteristics and distribution characteristics.

The village characteristics include geographical data of the village (area, perimeter, coordinates) with administratives information (population, department). `departement` and `INSEE_code` are set to None by default to include the case of a territory studied that do not correspond to a french municipality. `village_name` could correspond to the name of territory studied if it is not a municipality. Then, results from SEBE simulation are used to obtain the distribution of the average annual irradiation and the associated fitted Johnson's SU distribution. Statistical indicators to describe the distribution and the fitted Johnson's SU are obtained, as well as two indicators, SVI and DFI, to describe the far mask (topography) and the meteorologial situation of the municipality. The dataframe is then exported in a csv file.

Parameters

- **path_csv_folder** (*pathlib.Path*) – path of the folder with temporary csv files (define in main function)
- **path_raster_files** (*pathlib.Path*) – path of the folder with temporary raster files (define in main function)
- **path_final_output_folder** (*pathlib.Path*) – path of the folder with the final results (define in main function)
- **path_shapefiles** (*pathlib.Path*) – path of the folder with temporary shapefiles (define in main function)
- **grid_gpd** (*GeoDataFrame*) – geopandas grid file
- **path_meteorological_folder** (*pathlib.Path*) – path of the folder where are saved all the meteorological files
- **village_name** (*str*) – municipality name
- **village_INSEE_code** (*str, optional*) – INSEE code of the municipality, by default None
- **village_departement** (*str, optional*) – departement the municipality, by default None
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True
- **population_column** (*str, optional*) – name of the column with number of inhabitants of the studied municipality in the municipality footprint shapefile, by default "POPULATION"
- **nb_tests** (*int, optional*) – number of time to test downloading the horizon file if there is a connection problem, by default 5

Returns

df – dataframe with all the characteristics of the village and of the solar distribution

Return type

Dataframe

`toscana.results.village_characteristics.`**johnsonsu_mean**(*fitting_parameters*)

Calculate the mean value of a Johnson's SU distribution from fitting parameters a,b,c,d.

Parameters

fitting_parameters (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter

Returns

mean – mean value of the Johnson's SU distribution

Return type
float

toscana.results.village_characteristics.**johnsonsu_median**(*fitting_parameters*)

Calculate the median value of a Johnson's SU distribution from fitting parameters a,b,c,d.

Parameters
fitting_parameters (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter

Returns
median – median value of the Johnson's SU distribution

Return type
float

toscana.results.village_characteristics.**johnsonsu_mode**(*fitting_parameters*)

Calculate the mode value of a Johnson's SU distribution from fitting parameters a,b,c,d.

Parameters
fitting_parameters (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter

Returns

- **mode** (*float*) – mode value of the Johnson's SU distribution
- **p_mode** (*float*) – probability value at mode's value of the Johnson's SU distribution

toscana.results.village_characteristics.**johnsonsu_std**(*fitting_parameters*)

Calculate the standard deviation value of a Johnson's SU distribution from fitting parameters a,b,d.

Parameters
fitting_parameters (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,d), d is the scale parameter

Returns
std – standard deviation value of the Johnson's SU distribution

Return type
float

toscana.results.village_characteristics.**johnsonsu_variance**(*fitting_parameters*)

Calculate the variance value of a Johnson's SU distribution from fitting parameters a,b,d.

Parameters
fitting_parameters (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,d), d is the scale parameter

Returns
variance – variance value of the Johnson's SU distribution

Return type
float

toscana.results.village_characteristics.**johnsonsu_width**(*fitting_parameters*, *divide*)

Calculate the spread of a Johnson's SU distribution at a certain percentage of the maximum value (probability value of the mode) from fitting parameters a,b,c,d. It looks for the two x values for which the probability value is equal to 1/`divide` of the probability value of the mode, and then calculate the difference between these two values. It is made to calculte the spread at one-third maximum (`divide` = 3).

Parameters

- **fitting_parameters** (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter
- **divide** (*float*) – value used to divide the probability maximum value and obtain a probability height at which the spread of the distribution is wanted. This value is used to search for the x values corresponding of this probability value in the Johnson's SU distribution from fitting parameters a,b,c,d

Returns

- **values_below_mode** (*float*) – x value with probability equal to 1/divide of the probability value of the mode, x value smaller than the mode value
- **values_above_mode** (*float*) – x value with probability equal to 1/divide of the probability value of the mode, x value higher than the mode value
- **spread** (*float*) – spread of the distribution at 1/divide maximum

2.3.4 Module contents

2.4 toscana.solar_simulation package

2.4.1 Module contents

toscana.solar_simulation.**calculate_wallheight_wallaspect**(*path_DHM_clip*, *path_wallheight_clip*, *path_wallaspect_clip*, *wall_limit=0.1*)

Calculate wallheight and wallaspect rasters from DHM (or with DSM, both are possible), defining a limit of height to consider wall (0.1m by default (for DHM) (around 3m for DSM)).

Parameters

- **path_DHM_clip** (*path-like*) – path of the clip raster (DHM or DSM) (path_clip_raster from *clip_raster*)
- **path_wallheight_clip** (*path-like*) – path where to save the layer (wallheight raster clip files)
- **path_wallaspect_clip** (*path-like*) – path where to save the layer (wallaspect raster clip files)
- **wall_limit** (*float, optional*) – minimum difference of height to consider a pixel as a wall, by default 0.1

Raises

AssertionError – wall_limit must be greater than 0

toscana.solar_simulation.**iterate_on_grid**(*grid_gpd*, *path_final_output_folder*, *path_clip_files*, *path_raster_files*, *path_meteorological_folder*, *path_csv_folder*, *path_meteorological_subfolder=''*, *wall_limit=0.1*, *bool_global=True*, *utc=1*, *bool_save_sky_irradiance=True*, *albedo=0.15*, *restart_tile=1*, *average=True*)

Function used to iterate on the grid (run simulation for each grid tile) : the DSM, DHM and grid are clipped at the extent of one tile, wall aspects and wall heights are calculated and then SEBE simulation are run.

SEBE calculation are not done for the tiles for which the meteorological files could not have been downloaded. The iteration starts with the first tile (by default) but can be changed by changing restart_tile. Exceptions are

included to consider if SEBE calculation could not be run : problem when calculating the sky irradiance distribution (if direct and diffuse component are derived from global, error could appear when reprojecting the component on each patch of the sky vault) or if some wrong values are present in meteorological data (averaged or not). The list of the tiles for which the SEBE calculation could not have been done (due to a missing meteorological files, to an error in the calculation of the diffuse and direct component from the global, or due to an other error) are saved in a csv files with the corresponding error.

Parameters

- **grid_gpd** (*GeoDataFrame*) – geopandas grid file
- **path_final_output_folder** (*path-like*) – path of the folder where to save the final results (define in main function)
- **path_clip_files** (*path-like*) – path of the folder with temporary clip files (define in main function)
- **path_raster_files** (*path-like*) – path of the folder with temporary raster files (define in main function)
- **path_meteorological_folder** (*path-like*) – path of the folder where are saved all the meteorological files
- **path_meteorological_subfolder** (*path-like*) – path of the subfolder where are saved the meteorological files for that simulation
- **path_csv_folder** (*path-like*) – path of the folder with temporary csv files (define in main function)
- **wall_limit** (*float, optional*) – minimum difference of height to consider a pixel as a wall, by default 0.1
- **bool_global** (*bool, optional*) – boolean value to calculate or not the diffuse and direct irradiance values from global irradiance values, by default True
- **utc** (*int, optional*) – value of utc time, by default 1 from -12 to 12, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **bool_save_sky_irradiance** (*bool, optional*) – boolean to save or not the sky irradiance data, by default True
- **albedo** (*float, optional*) – value of the albedo, by default 0.15 from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **restart_tile** (*int, optional*) – number of the first tile on which to run SEBE simulation (to change to start not from the beginning), by default 1
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True

Raises

AssertionError – restart_tile must be smaller than the number of grid tiles and higher than 1 (first tile)

toscana.solar_simulation.**launch_iterate_on_grid_per_month**(*grid_gpd*, *list_month*, *path_final_output_folder*, *path_clip_files*, *path_raster_files*, *path_meteorological_folder*, *path_csv_folder*, *wall_limit=0.1*, *bool_global=True*, *utc=1*, *bool_save_sky_irradiance=True*, *restart_tile=1*, *average=True*, *list_albedo_month=[]*, *albedo_m=0.15*)

Function used to iterate on the grid for several months.

Parameters

- **grid_gpd** (*GeoDataFrame*) – geopandas grid file
- **list_month** (*list*) – list of the months that should be simulated
- **path_final_output_folder** (*path-like*) – path of the folder where to save the final results (define in main function)
- **path_clip_files** (*path-like*) – path of the folder with temporary clip files (define in main function)
- **path_raster_files** (*path-like*) – path of the folder with temporary raster files (define in main function)
- **path_meteorological_folder** (*path-like*) – path of the folder where are saved all the meteorological files
- **path_csv_folder** (*path-like*) – path of the folder with temporary csv files (define in main function)
- **wall_limit** (*float, optional*) – minimum difference of height to consider a pixel as a wall, by default 0.1
- **bool_global** (*bool, optional*) – boolean value to calculate or not the diffuse and direct irradiance values from global irradiance values, by default True
- **utc** (*int, optional*) – value of utc time, by default 1 from -12 to 12, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **bool_save_sky_irradiance** (*bool, optional*) – boolean to save or not the sky irradiance data, by default True
- **restart_tile** (*int, optional*) – number of the first tile on which to run SEBE simulation (to change to start not from the beginning), by default 1
- **average** (*bool, optional*) – boolean value to indicate if an average of the meteorological files was done or not, by default True
- **list_albedo_month** (*list, optional*) – monthly value of the albedo, by default [] from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **albedo_m** (*float, optional*) – average albedo value (for all the months) if monthly values are not defined, by default 0.15 from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation

Raises

- **TypeError** – list_month must be a list of int (or a list of strings with numbers)
- **AssertionError** – value in list_month should be between 1 and 12

toscana.solar_simulation.**run_SEBE_simulation**(*path_DSM_clip*, *path_wallheight_clip*, *path_wallaspect_clip*, *path_average_meteorological_file*, *path_output_SEBE_temp_folder*, *path_sky_irradiance*, *path_roof_irradiance*, *bool_global=True*, *utc=1*, *bool_save_sky_irradiance=True*, *albedo=0.15*)

Launch the SEBE algorithm to calculate irradiation on surfaces, precising an average `albedo` value of surfaces (default : 0.15), the time zone (default :1), calculating direct and diffuse irradiance from global irradiance if necessary (default: True) and saving the sky irradiance distribution (distribution of irradiance components) (default : True).

Parameters

- **path_DSM_clip** (*path-like*) – path of the DSM clip raster (path_output from *clip_raster*)
- **path_wallheight_clip** (*path-like*) – path of the clip wallheight (from *calculate_wallheight_wall_aspect*)
- **path_wallaspect_clip** (*path-like*) – path of the clip wallaspect (from *calculate_wallheight_wall_aspect*)
- **path_average_meteorological_file** (*path-like*) – path of the average txt meteorological file (obtained in *obtain_average_meteorological_files*)
- **path_output_SEBE_temp_folder** (*path-like*) – path of the folder to save output of the calculation
- **path_sky_irradiance** (*path-like*) – path where to save the sky irradiance distribution
- **path_roof_irradiance** (*path-like*) – path where to save the roof irradiance raster
- **bool_global** (*bool, optional*) – boolean value to calculate or not the diffuse and direct irradiance value from global irradiance value, by default True
- **utc** (*int, optional*) – value of utc time, by default 1 from -12 to 12, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation
- **bool_save_sky_irradiance** (*bool, optional*) – boolean to save or not the sky irradiance data, by default True
- **albedo** (*float, optional*) – value of the albedo, by default 0.15 from 0 to 1, see umep:Solar Radiation: Solar Energy of Builing Envelopes (SEBE) documentation

Raises

- **TypeError** – albedo must be a float (or a string of numbers)
- **TypeError** – uct must be a int (or a string of numbers)
- **AssertionError** – albedo must be between 0 and 1
- **AssertionError** – utc must be between -12 and 12

2.5 toscana.utils package

2.5.1 Module contents

toscana.utils.**calculate_histogram_and_johnsonsu_fit**(*path_irradiation_csv*, *bool_buffer=True*)

Calculate the histogram/distribution from a csv file and calculate the fitted Johnson's SU distribution.

It is especially made to calculate the histogram for the distribution of the average annual irradiation received by building rooftops, with x value representing irradiation values and y value representing probability values. Two methods are tested to fit the Johnson's SU distribution : a classic method (init) and a method by setting floc = xmax (xmax, setting the location parameter to the maximum value of x value of the real distribution). The fitting parameters (a,b,c,d) with the best R2 coefficient value are kept.

Parameters

- **path_irradiation_csv** (*pathlib.Path*) – path of the csv file with irradiation value (path_irradiation_csv obtained in generate_irradiation_csv_file)

- **bool_buffer** (*bool, optional*) – boolean value to specify or not if the shapefile with building footprints used to obtain the irradiation value has a buffer and then na value need to be removed, by default True

Returns

- **data** (*DataFrame*) – dataframe with irradiation values
- **nb_bins** (*int*) – number of bins in the histogram
- **fitting_parameters** (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter
- **x** (*list*) – x values used to calculate the histogram (middle value of the bar), with two more values added (that will be used for the display of Johnson's SU fit)
- **R2** (*float*) – best R2 coefficient calculated between the real distribution and the fitted Johnson's SU distribution between the two methods (classic (init) or floc=xmax(xmax))
- **R2_init** (*float*) – R2 coefficient calculated between the real distribution and the fitted Johnson's SU distribution with classic method
- **R2_xmax** (*float*) – R2 coefficient calculated between the real distribution and the fitted Johnson's SU distribution by setting floc = xmax (maximum value of the x value of the real distribution)
- **method** (*str*) – name of the method with the best R2 coefficient between the classic method or by setting floc = xmax (maximum value of the x value of the real distribution) (init for initial, or xmax for floc =xmax)

toscana.utils.**calculate_r2_johnsonsu**(*fitting_parameters*, *x_obs*, *y_obs*)

Calculate the R2 coefficient between the real distribution (`x_obs` and `y_obs`) and the Johnson's SU distribution from *fitting_parameters* ` a,b,c,d.

Parameters

- **fitting_parameters** (*tuple*) – float tuple with the fitting parameters of the Johnson's SU distribution (a,b,c,d), c is the location parameter and d is the scale parameter
- **x_obs** (*list*) – list of x value of the real distribution (irradiation)
- **y_obs** (*list*) – list of y value of the real distribution (probability)

Returns

r2_johnsonsu – R2 coefficient calculated between real distribution and Johnson's SU distribution.

Return type

float

toscana.utils.**clip_raster**(*path_mask_shapefiles*, *path_input_raster*, *path_clip_raster*)

Clip a raster based on a shapefile mask layer (a grid tile for example).

Parameters

- **path_mask_shapefiles** (*pathlib.Path*) – path of the shapefile used as mask (clip to the shapefile extent) (for example path_clip_grid obtained in *clip_grid* or path_reproject_municipality_footprint obtained in *reproject_municipality_footprint*)
- **path_input_raster** (*pathlib.Path*) – path of the raster that need to be clipped (for example DSM, DHM)
- **path_clip_raster** (*pathlib.Path*) – path where to save the clip raster layer (raster tiles for example)

toscana.utils.**create_centroid**(*path_shapefile*, *path_centroid*)

Create centroids of a shapefile (grid, municipality footprint for example).

Parameters

- **path_shapefile** (*pathlib.Path*) – path of the shapefile for which the centroids are wanted (for example path_grid obtained in *define_grid*, or path_reproject_municipality_footprint obtained in *reproject_municipality_footprint*)
- **path_centroid** (*pathlib.Path*) – path where to save the created shapefile with the centroids

Returns
gdf_centroid – geopandas file with the centroids

Return type
GeoDataFrame

toscana.utils.**create_csv_coordinates**(*path_shapefile*, *path_shapefile_coordinates*, *path_csv_file*)

Obtain the coordinates (longitude, latitude) of some points (centroids for example) in a shapefile and transform it into a csv file.

Parameters

- **path_shapefile** (*pathlib.Path*) – path of the shapefile with the points for which the coordinates are wanted
- **path_shapefile_coordinates** (*pathlib.Path*) – path where to save the shapefile with the coordinates of the points
- **path_csv_file** (*pathlib.Path*) – path where to save the csv file with the coordinates of the points

Returns
df_points – dataframe with coordinates of the points.

Return type
DataFrame

toscana.utils.**zonal_statistics**(*path_shapefile*, *path_raster*, *path_zonal_statistics*, *bool_count=True*, *bool_sum=True*, *bool_mean=True*, *column_prefix='_'*)

Obtain statistics for each shape of a shapefile according to a raster file (count the number of pixel (default :True), sum the pixel values (default : True), average of the pixel values (default : True)). `column_prefix` could be used to define the name of the column where will be stored the statistics (linked with the raster data that are used to calculate the statistics for example).

Parameters

- **path_shapefile** (*pathlib.Path*) – path of the shapefile containing the shapes for which statistics are wanted (path_buildings_buffer obtained in *create_buffer* for example)
- **path_raster** (*pathlib.Path*) – path of the raster containing the value on which statistics will be calculated (path_merge_SEBE_raster obtained in *merge_SEBE_raster* for example)
- **path_zonal_statistics** (*pathlib.Path*) – path where to save the layer with zonal statistics
- **bool_count** (*bool, optional*) – boolean value to obtain or not the number of pixel inside each shape, by default True
- **bool_sum** (*bool, optional*) – boolean value to obtain or not the sum of the pixel values inside each shape, by default True

- **bool_mean** (*bool, optional*) – boolean value to obtain or not the average of the pixel values inside each shape, by default True
- **column_prefix** (*str, optional*) – prefix name of the columns that will be created to store the different statistics, by default '_'

Returns

column_prefix – prefix name of the columns that are created to store the different statistics

Return type

str

CHAPTER

THREE

INDICES AND TABLES

- modindex

PYTHON MODULE INDEX

t

INDEX

G

I

J

L

M

O

P

R

S

T

Z