pybdshadow

Release 0.3.4

Qing Yu

Jun 01, 2023

INSTALLATION AND DEPENDENCIES

pybdshadow

CHAPTER

ONE

INTRODUCTION

pybdshadow is a python package for generating, analyzing and visualizing building shadows from large scale building geographic data. *pybdshadow* support generate building shadows from both sun light and point light. *pybdshadow* provides an efficient and easy-to-use method to generate a new source of geospatial data with great application potential in urban study.

The latest stable release of the software can be installed via pip and full documentation can be found [here](https://pybdshadow.readthedocs.io/en/latest/).

CHAPTER

TWO

FUNCTIONALITY

Currently, *pybdshadow* mainly provides the following methods:

- **Generating building shadow from sun light**: With given location and time, the function in *pybdshadow* uses the properties of sun position obtained from *suncalc-py* and the building height to generate shadow geometry data.
- **Generating building shadow from point light**: *pybdshadow* can generate the building shadow with given location and height of the point light, which can be potentially useful for visual area analysis in urban environment.
- **Analysis**: *pybdshadow* integrated the analysing method based on the properties of sun movement to track the changing position of shadows within a fixed time interval. Based on the grid processing framework provided by *TransBigData*, *pybdshadow* is capable of calculating sunshine time on the ground and on the roof.
- **Visualization**: Built-in visualization capabilities leverage the visualization package *keplergl* to interactively visualize building and shadow data in Jupyter notebooks with simple code.

The target audience of *pybdshadow* includes data science researchers and data engineers in the field of BIM, GIS, energy, environment, and urban computing.

CHAPTER

THREE

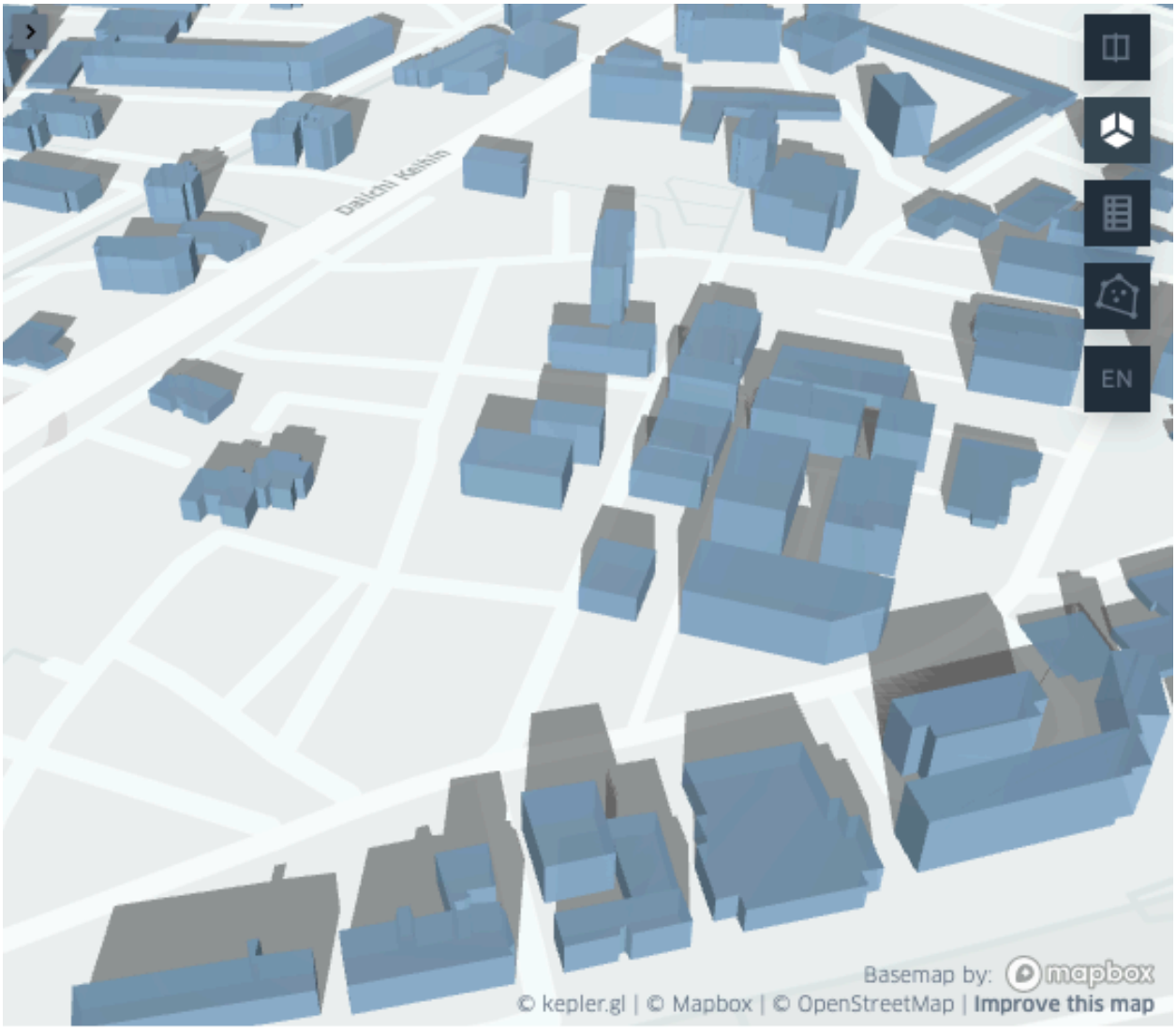
EXAMPLE

Given a building GeoDataFrame and UTC datetime, *pybdshadow* can calculate the building shadow based on the sun position obtained by *suncalc*

```
import pybdshadow
#Given UTC datetime
date = pd.to_datetime('2022-01-01 12:45:33.959797119')\
         .tz_localize('Asia/Shanghai')\
         .tz_convert('UTC')
#Calculate building shadow
shadows = pybdshadow.bdshadow_sunlight(buildings,date)
```

pybdshadow also provide visualization method supported by keplergl.

```
# visualize buildings and shadows
pybdshadow.show_bdshadow(buildings = buildings,shadows = shadows)
```



3.1 Installation and dependencies

3.1.1 Installation

It is recommended to use *Python 3.7, 3.8, 3.9*.

pybdshadow can be installed by using *pip install*. Before installing *pybdshadow*, make sure that you have installed the available *geopandas* package: https://geopandas.org/en/stable/getting_started/install.html.

If you already have geopandas installed, run the following code directly from the command prompt to install *pybdshadow*:

```
pip install pybdshadow
```

3.1.2 Dependency

pybdshadow depends on the following packages

- *numpy*
- *pandas*
- *shapely*
- *rtree*
- *geopandas*
- *matplotlib*
- *suncalc*
- *keplergl*
- *TransBigData*

3.2 Building shadow analysis

Notebook for this example: here.

In this example, we will introduce how to use `pybdshadow` to generate, analyze and visualize the building shadow data

3.2.1 Building data preprocessing

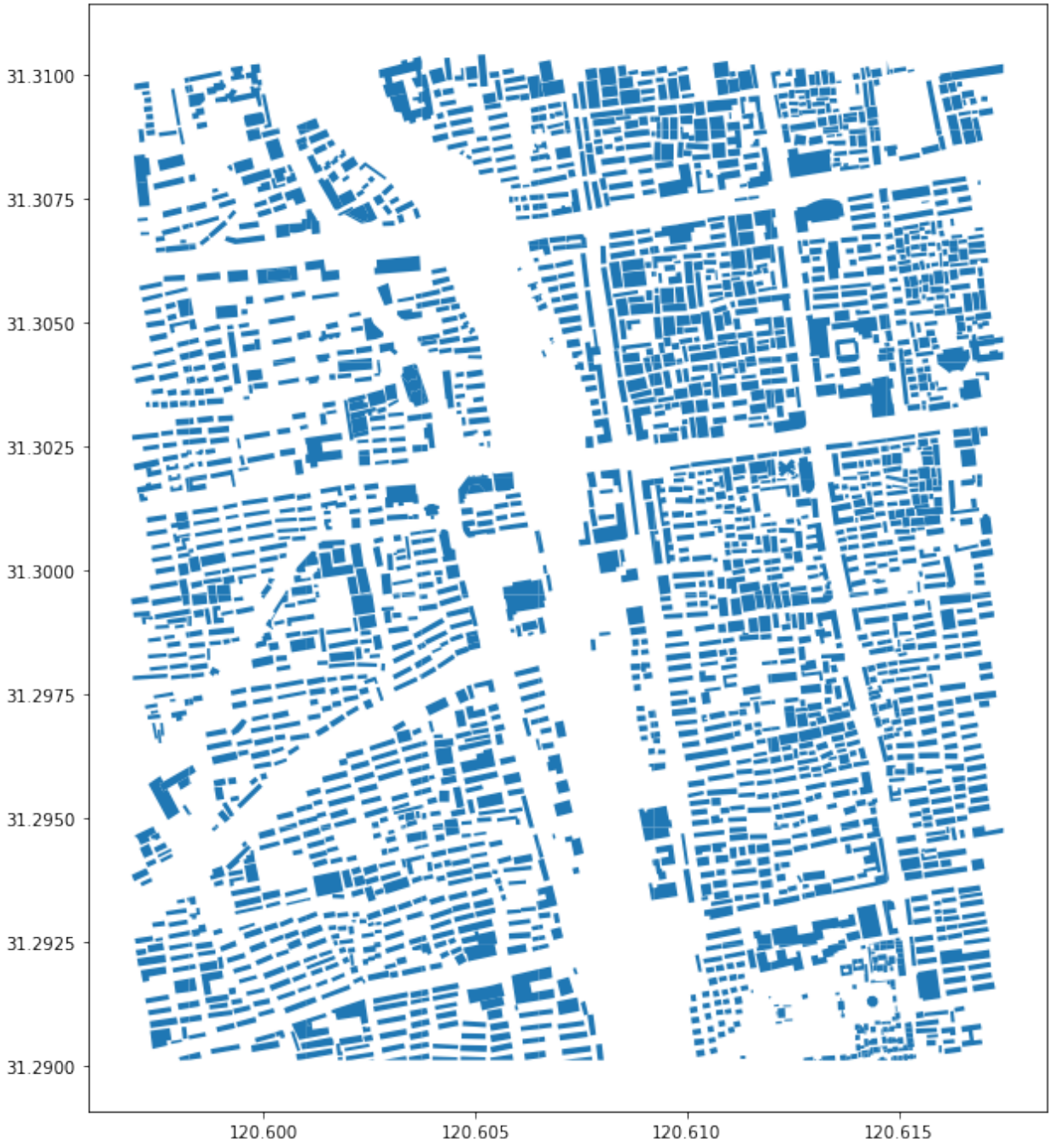
Building data can be obtain by Python package OSMnx from OpenStreetMap (Some of the buildings do not contain the height information).

The buildings are usually store in the data as the form of Polygon object with `height` column. Here, we provide a demo building data store as GeoJSON file to demonstrate the functionality of `pybdshadow`

```
import pandas as pd
import geopandas as gpd
import pybdshadow
#Read building data
buildings = gpd.read_file(r'../example/data/bd_demo_2.json')
buildings.head(5)
```

The input building data must be a `GeoDataFrame` with the `height` column storing the building height information and the `geometry` column storing the geometry polygon information of building outline.

```
#Plot the buildings
buildings.plot(figsize=(12,12))
```



Before analysing buildings, make sure to preprocess building data using *pybdshadow.bd_preprocess()* before calculate shadow. It will remove empty polygons, convert multipolygons into polygons and generate `building_id` for each building.

```
buildings = pybdshadow.bd_preprocess(buildings)
buildings.head(5)
```

3.2.2 Generate building shadows

Shadow generated by Sun light

Given a building GeoDataFrame and UTC datetime, `pybdshadow` can calculate the building shadow based on the sun position obtained by `suncalc`

```
#Given UTC time
date = pd.to_datetime('2022-01-01 12:45:33.959797119')\
    .tz_localize('Asia/Shanghai')\
    .tz_convert('UTC')
#Calculate shadows
shadows = pybdshadow.bdshadow_sunlight(buildings,date,roof=True,include_building =␣
↪False)
shadows
```

The generated shadow data is store as another `GeoDataFrame`. It contains both rooftop shadow(with height over 0) and ground shadow(with height equal to 0).

```
# Visualize buildings and shadows using matplotlib
import matplotlib.pyplot as plt
fig = plt.figure(1, (12, 12))
ax = plt.subplot(111)

# plot buildings
buildings.plot(ax=ax)

# plot shadows
shadows.plot(ax=ax, alpha=0.7,
             column='type',
             categorical=True,
             cmap='Set1_r',
             legend=True)

plt.show()
```

pybdshadow also provide 3D visualization method supported by keplergl.

```
#Visualize using keplergl
pybdshadow.show_bdshadow(buildings = buildings,shadows = shadows)
```

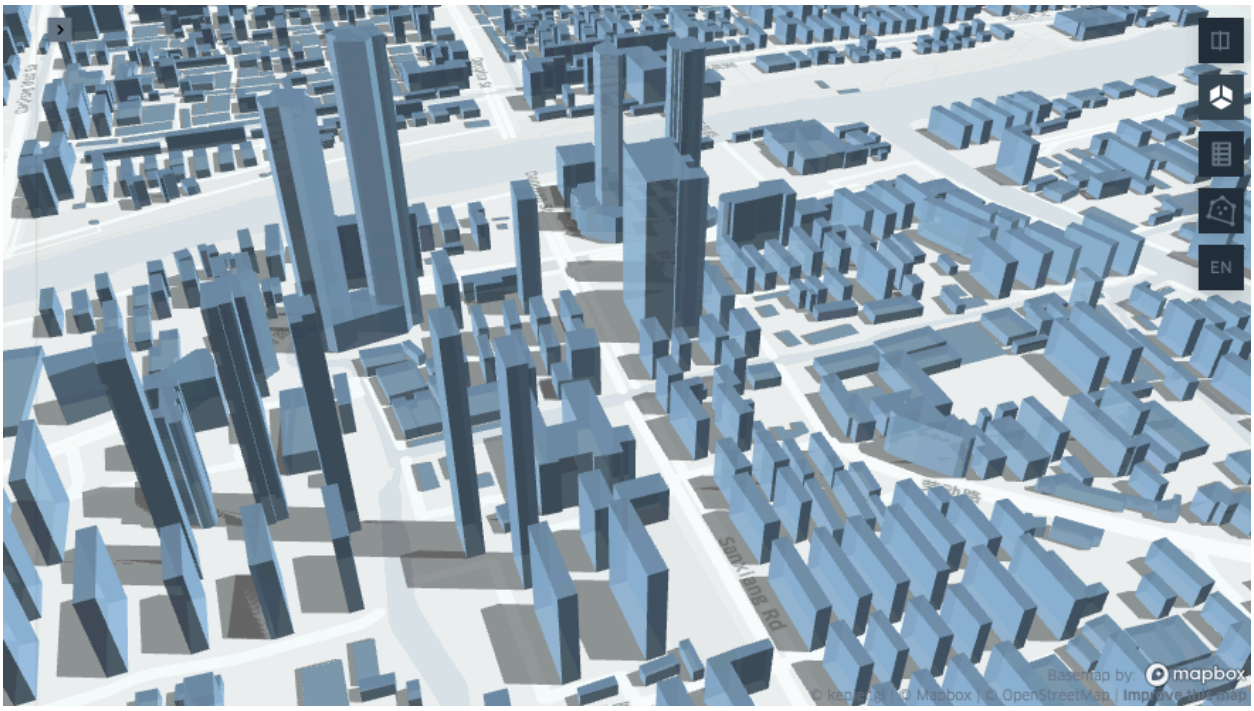


Fig. 1: 1649161376291.png

Shadow generated by Point light

pybdshadow can generate the building shadow generated by point light, which can be potentially useful for visual area analysis in urban environment. Given coordinates and height of the point light:

```
#Define the position and the height of the point light
pointlon,pointlat,pointheight = [120.60820619503946,31.300141884245672,100]
#Calculate building shadow for point light
shadows = pybdshadow.bdshadow_pointlight(buildings,pointlon,pointlat,pointheight)
#Visualize buildings and shadows
pybdshadow.show_bdshadow(buildings = buildings,shadows = shadows)
```

3.2.3 Shadow coverage analysis

To demonstrate the analysis function of pybdshadow, here we select a smaller area for detail analysis of shadow coverage.

```
#define analysis area
bounds = [120.603,31.303,120.605,31.305]
#filter the buildings
buildings['x'] = buildings.centroid.x
buildings['y'] = buildings.centroid.y
```

(continues on next page)

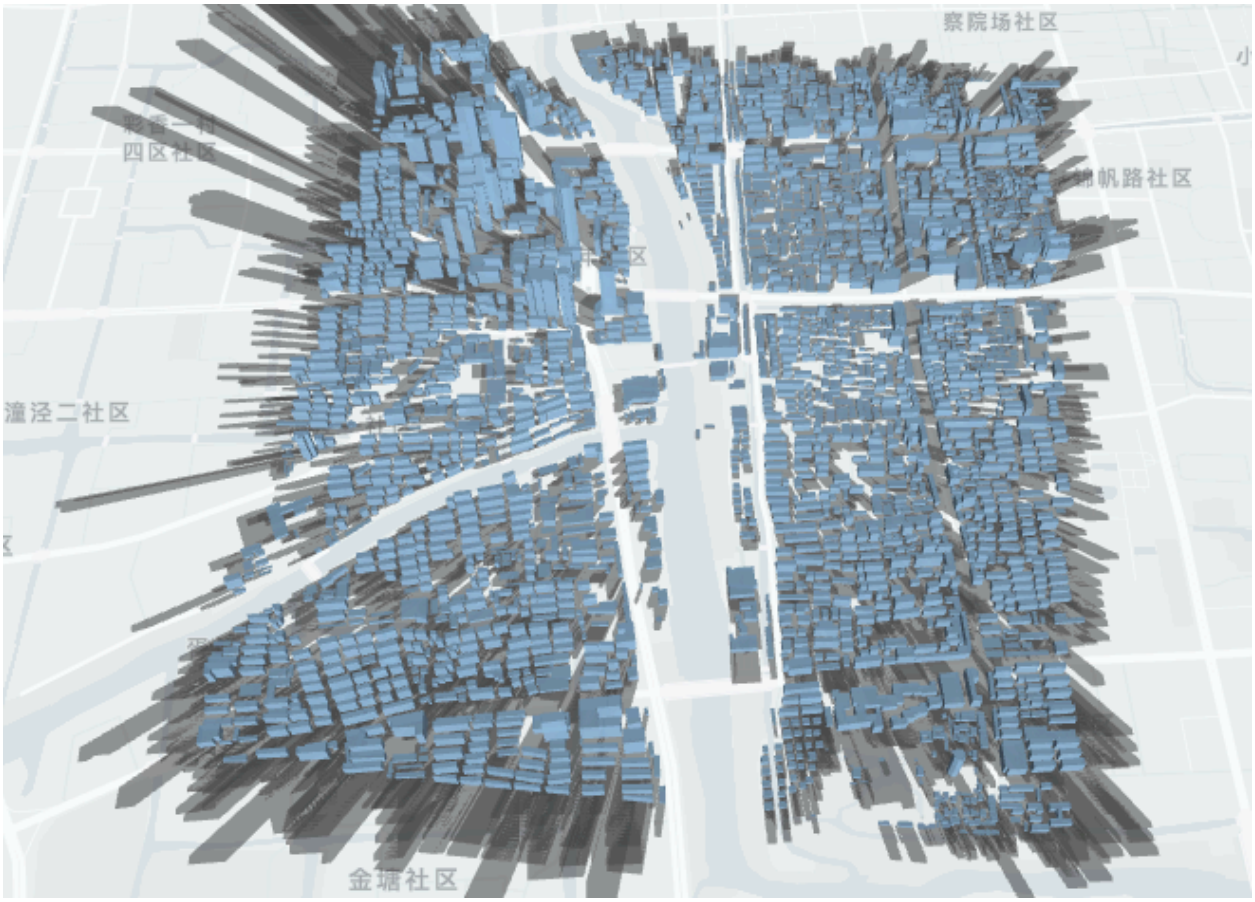
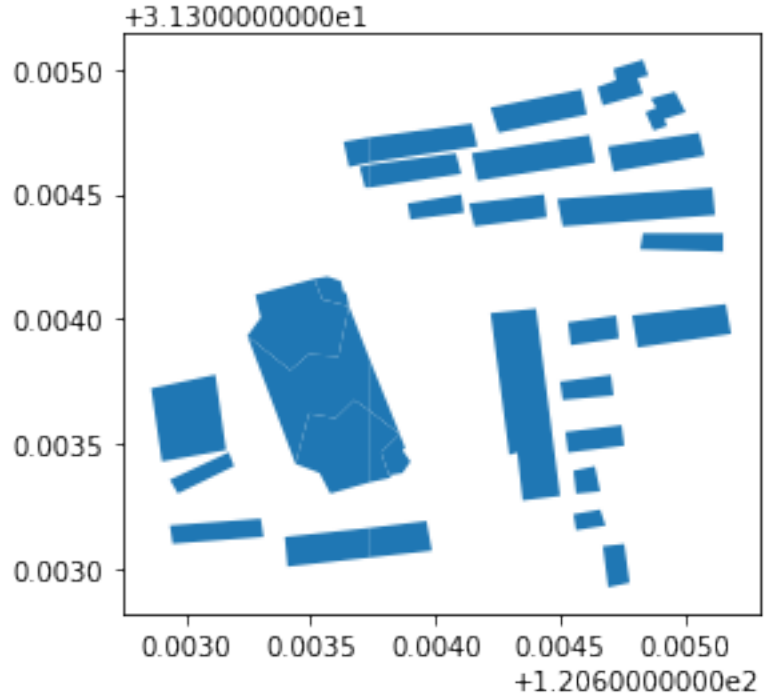


Fig. 2: 1649405838683.png

(continued from previous page)

```
buildings_analysis = buildings[(buildings['x'] > bounds[0]) &
                        (buildings['x'] <  bounds[2]) &
                        (buildings['y'] >  bounds[1]) &
                        (buildings['y'] <  bounds[3])]
buildings_analysis.plot()
```



Use *pybdshadow.cal_sunshine()* to analyse shadow coverage and sunshine time. Here, we select `2022-01-01` as the date, set the spatial resolution of 1 meter*1 meter grids, and 900 s as the time interval.

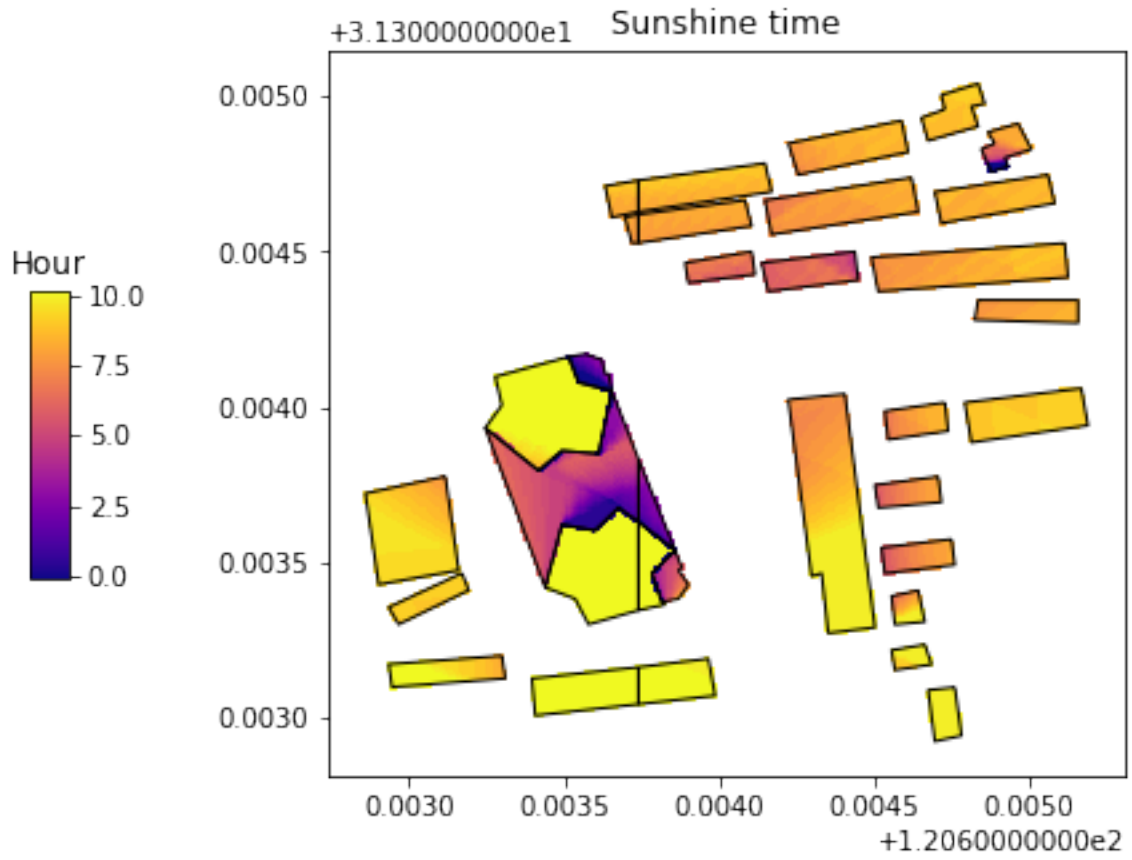
```
#calculate sunshine time on the building roof
sunshine = pybdshadow.cal_sunshine(buildings_analysis,
                                   day='2022-01-01',
                                   roof=True,
                                   accuracy=1,
                                   precision=900)
```

```
#Visualize buildings and sunshine time using matplotlib
import matplotlib.pyplot as plt
fig = plt.figure(1,(10,5))
ax = plt.subplot(111)
#define colorbar
cax = plt.axes([0.15, 0.33, 0.02, 0.3])
plt.title('Hour')
#plot the sunshine time
```

(continues on next page)

(continued from previous page)

```
sunshine.plot(ax = ax,cmap = 'plasma',column ='Hour',alpha = 1,legend = True,cax =␣
↪cax,)
#Buildings
buildings_analysis.plot(ax = ax,edgecolor='k',facecolor=(0,0,0,0))
plt.sca(ax)
plt.title('Sunshine time')
plt.show()
```



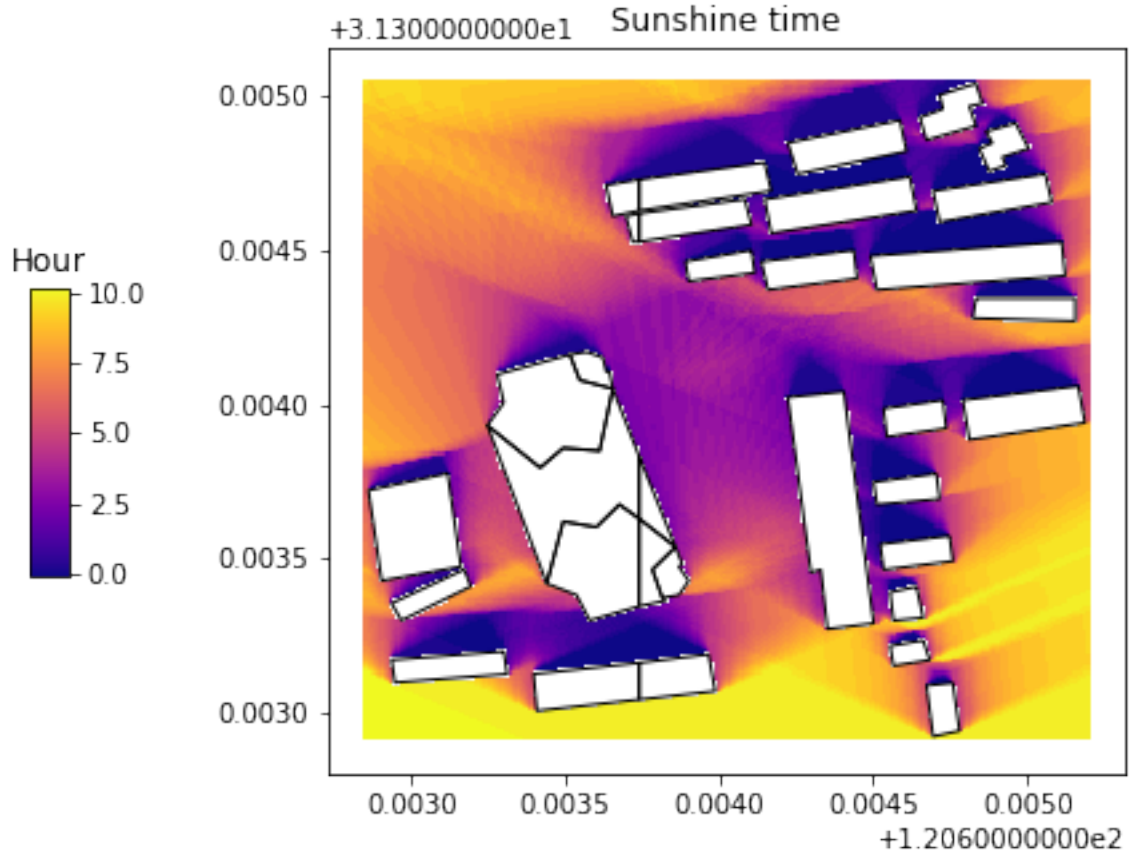

```
#calculate sunshine time on the ground (set the roof to False)
sunshine = pybdshadow.cal_sunshine(buildings_analysis,
                                   day='2022-01-01',
                                   roof=False,
                                   accuracy=1,
                                   precision=900)
```

```
#Visualize buildings and sunshine time using matplotlib
import matplotlib.pyplot as plt
fig = plt.figure(1,(10,5))
ax = plt.subplot(111)
```

(continues on next page)

(continued from previous page)

```
#define colorbar
cax = plt.axes([0.15, 0.33, 0.02, 0.3])
plt.title('Hour')
#plot the sunshine time
sunshine.plot(ax = ax,cmap = 'plasma',column ='Hour',alpha = 1,legend = True,cax =␣
↪cax,)
#Buildings
buildings_analysis.plot(ax = ax,edgecolor='k',facecolor=(0,0,0,0))
plt.sca(ax)
plt.title('Sunshine time')
plt.show()
```



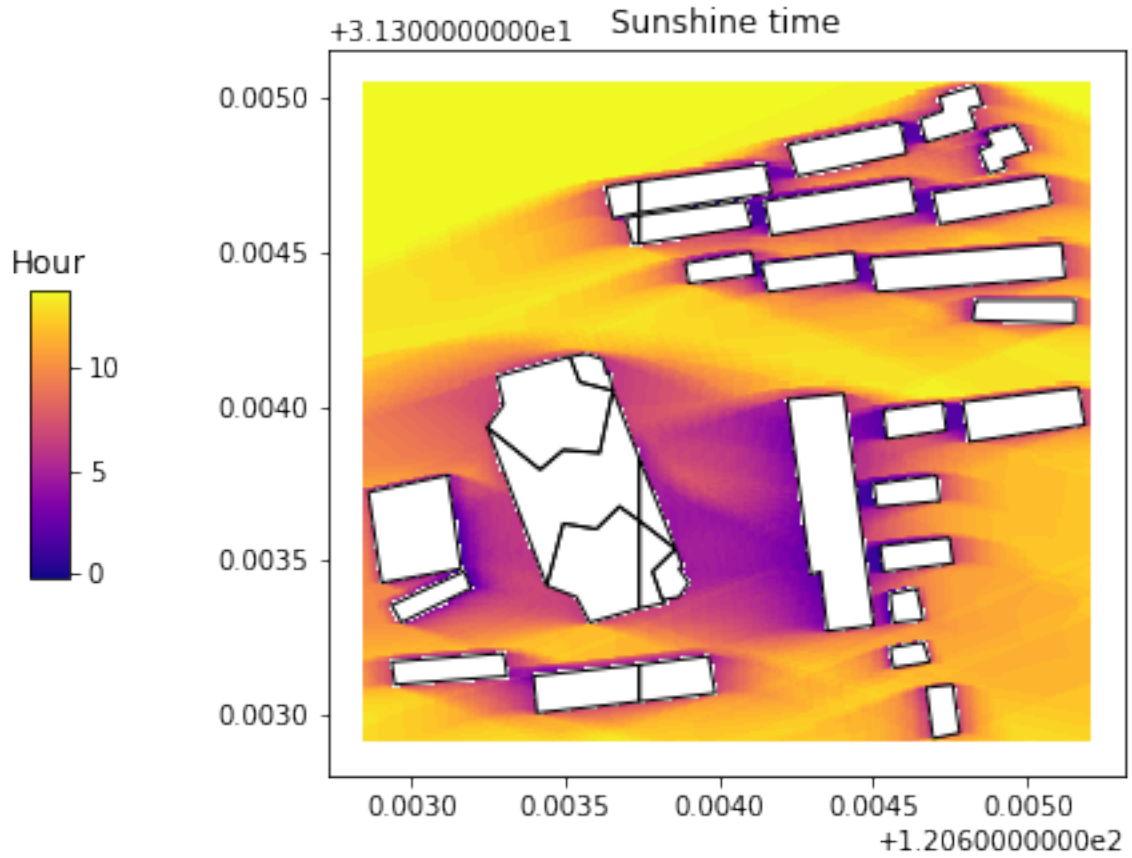
We can change the date to see if it has different result:

```
#calculate sunshine time on the ground (set the roof to False)
sunshine = pybdshadow.cal_sunshine(buildings_analysis,
                                   day='2022-07-15',
                                   roof=False,
                                   accuracy=1,
                                   precision=900)
```

(continues on next page)

(continued from previous page)

```
#Visualize buildings and sunshine time using matplotlib
import matplotlib.pyplot as plt
fig = plt.figure(1,(10,5))
ax = plt.subplot(111)
#define colorbar
cax = plt.axes([0.15, 0.33, 0.02, 0.3])
plt.title('Hour')
#plot the sunshine time
sunshine.plot(ax = ax,cmap = 'plasma',column ='Hour',alpha = 1,legend = True,cax =␣
↪cax,)
#Buildings
buildings_analysis.plot(ax = ax,edgecolor='k',facecolor=(0,0,0,0))
plt.sca(ax)
plt.title('Sunshine time')
plt.show()
```



3.3 Building Preprocess

3.3.1 Building preprocess

pybdshadow.**bd_preprocess**(*buildings*, *height=''*)

Preprocess building data, so that we can perform shadow calculation. Remove empty polygons and convert multi-polygons into polygons.

Parameters

- **buildings** (*GeoDataFrame*) –Buildings.
- **height** (*string*) –Column name of building height(meter).

Returns

allbds –Polygon buildings

Return type

GeoDataFrame

3.4 Building shadow

3.4.1 Shadow from sunlight

pybdshadow.**bdshadow_sunlight**(*buildings*, *date*, *height='height'*, *roof=False*, *include_building=True*, *ground=0*)

Calculate the sunlight shadow of the buildings.

Parameters

- **buildings** (*GeoDataFrame*) –Buildings. coordinate system should be WGS84
- **date** (*datetime*) –Datetime
- **height** (*string*) –Column name of building height(meter).
- **roof** (*bool*) –Whether to calculate the roof shadows.
- **include_building** (*bool*) –Whether the shadow include building outline.
- **ground** (*number*) –Height of the ground(meter).

Returns

shadows –Building shadow

Return type

GeoDataFrame

3.4.2 Shadow from pointlight

pybdshadow.**bdshadow_pointlight**(*buildings*, *pointlon*, *pointlat*, *pointheight*, *merge=True*, *height='height'*, *ground=0*)

Calculate the sunlight shadow of the buildings.

Parameters

- **buildings** (*GeoDataFrame*) –Buildings. coordinate system should be WGS84
- **pointlon** (*float*) –Point light coordinates and height(meter).
- **pointlat** (*float*) –Point light coordinates and height(meter).
- **pointheight** (*float*) –Point light coordinates and height(meter).
- **date** (*datetime*) –Datetime
- **merge** (*bool*) –Whether to merge the wall shadows into the building shadows
- **height** (*string*) –Column name of building height(meter).
- **ground** (*number*) –Height of the ground

Returns

shadows –Building shadow

Return type

GeoDataFrame

3.5 Shadow coverage

3.5.1 Shadow coverage

pybdshadow.**cal_sunshine**(*buildings*, *day='2022-01-01'*, *roof=False*, *grids=Empty GeoDataFrame Columns: [] Index: []*, *accuracy=1*, *precision=3600*, *padding=1800*)

Calculate the sunshine time in given date.

Parameters

- **buildings** (*GeoDataFrame*) –Buildings. coordinate system should be WGS84
- **day** (*str*) –the day to calculate the sunshine
- **roof** (*bool*) –whether to calculate roof shadow.
- **grids** (*GeoDataFrame*) –grids generated by TransBigData in study area
- **precision** (*number*) –time precision(s)
- **padding** (*number*) –padding time before and after sunrise and sunset

- **accuracy** (*number*) –size of grids. Produce vector polygons if set as *vector*

Returns

grids –grids generated by TransBigData in study area, each grids have a *time* column store the sunshine time

Return type

GeoDataFrame

pybdshadow.**cal_sunshadows**(*buildings*, *cityname='somecity'*, *dates=['2022-01-01']*, *precision=3600*, *padding=1800*, *roof=True*, *include_building=True*, *save_shadows=False*, *printlog=False*)

Calculate the sunlight shadow in different date with given time precision.

Parameters

- **buildings** (*GeoDataFrame*) –Buildings. coordinate system should be WGS84
- **cityname** (*string*) –Cityname. If save_shadows, this function will create *result/cityname* folder to save the shadows
- **dates** (*list*) –List of dates
- **precision** (*number*) –Time precision(s)
- **padding** (*number*) –Padding time (second) before and after sunrise and sunset. Should be over 1800s to avoid sun altitude under 0
- **roof** (*bool*) –whether to calculate roof shadow.
- **include_building** (*bool*) –whether the shadow include building outline
- **save_shadows** (*bool*) –whether to save calculated shadows
- **printlog** (*bool*) –whether to print log

Returns

allshadow –All building shadows calculated

Return type

GeoDataFrame

pybdshadow.**cal_shadowcoverage**(*shadows_input*, *buildings*, *grids=Empty GeoDataFrame Columns: [] Index: []*, *roof=True*, *precision=3600*, *accuracy=1*)

Calculate the sunlight shadow coverage time for given area.

Parameters

- **shadows_input** (*GeoDataFrame*) –All building shadows calculated
- **buildings** (*GeoDataFrame*) –Buildings. coordinate system should be WGS84
- **grids** (*GeoDataFrame*) –grids generated by TransBigData in study area

- **roof** (*bool*) –If true roof shadow, false then ground shadow
- **precision** (*number*) –time precision(s), which is for calculation of coverage time
- **accuracy** (*number*) –size of grids.

Returns

grids –grids generated by TransBigData in study area, each grids have a *time* column store the shadow coverage time

Return type

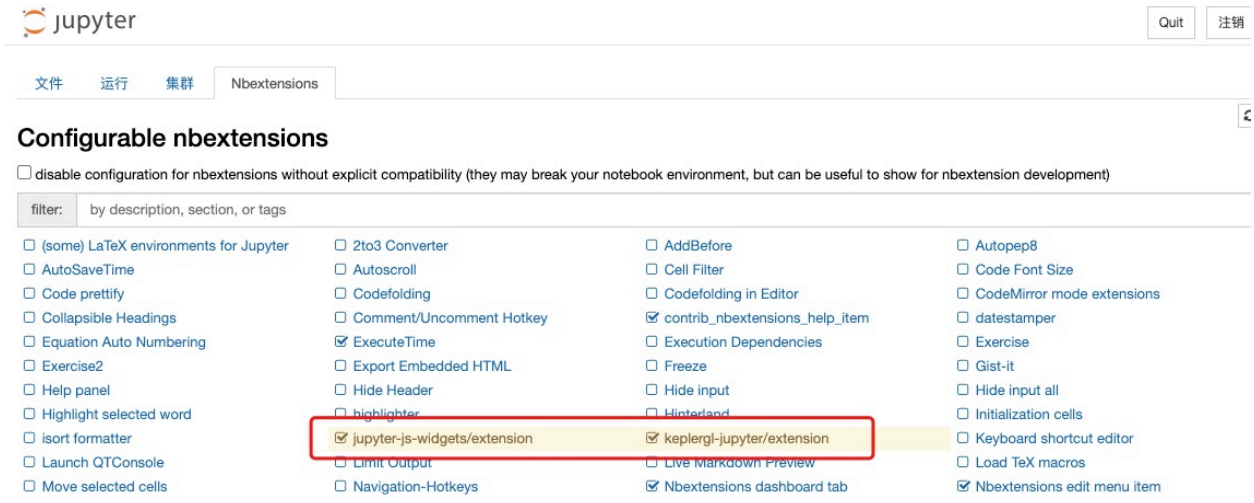
GeoDataFrame

3.6 Visualization

3.6.1 Visualization Settings in Jupyter

The *pybdshadow`* package provide visualization methods based on the visualization plugin provided by *kepler.gl*.

If you want to display the visualization results in jupyter notebook, you need to check the jupyter-js-widgets (which may need to be installed separately) and keplergl-jupyter plugins



3.6.2 Visualization

pybdshadow.**show_bdshadow**(*buildings=Empty GeoDataFrame Columns: [] Index: []*, *shadows=Empty GeoDataFrame Columns: [] Index: []*, *ad=Empty GeoDataFrame Columns: [] Index: []*, *ad_visualArea=Empty GeoDataFrame Columns: [] Index: []*, *height='height'*, *zoom='auto'*)

Visualize the building and shadow with keplergl.

Parameters

- **buildings** (*GeoDataFrame*) –Buildings. coordinate system should be WGS84
- **shadows** (*GeoDataFrame*) –Building shadows. coordinate system should be WGS84
- **ad** (*GeoDataFrame*) –Advertisment. coordinate system should be WGS84
- **ad_visualArea** (*GeoDataFrame*) –Visualarea of Advertisment. coordinate system should be WGS84
- **height** (*string*) –Column name of building height
- **zoom** (*number*) –Zoom level of the map

Returns

vmap –Visualizations provided by keplergl

Return type

keplergl.keplergl.KeplerGl

INDEX

B

C

S