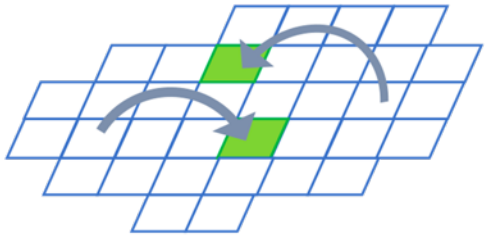
同濟大學 TONGJI UNIVERSITY

MAGIC



TransBigData

A Python Package for Transportation Spatiotemporal Big Data Processing, Analysis, and Visualization

Jian Yuan, Qing Yu

2022/3/12

TransBigData

Introduction

➢ Background

- TransBigData is a Python package developed for transportation spatiotemporal data **processing**, **analysis**, and **visualization**
- Target data includes vehicle GPS trajectories, bike sharing data, IC card data

➢ Features

- Provide **comprehensive methods** for each stage of spatiotemporal data analysis
  - Data Quality
  - Data Preprocess
  - Data Gridding
  - Data Aggregating
  - Data Visualization
  - Trajectory Processing
  - Base-map Loading

- Allowing **complex processing tasks** to be achieved **with concise code** (e.g., OD matrix extraction)
- Potential to implement combinatory analysis with other packages

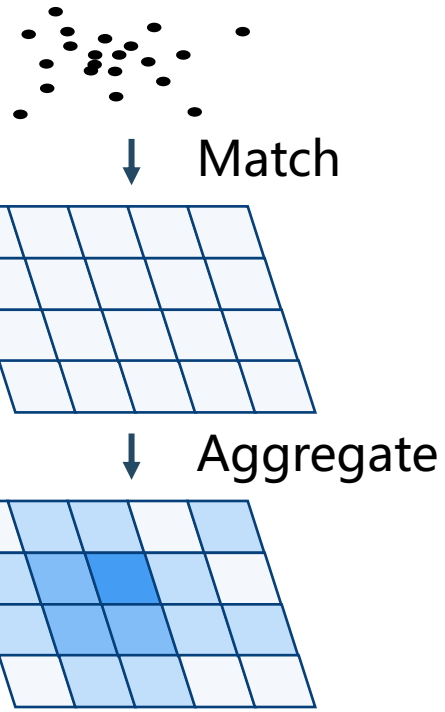
https://github.com/ni1o1/transbigdata

TransBigData

Why aggregate data to grids?

- **Discretization**
  Hard to analyze data in continuous space, but easy to analyze with discretized region
- **Comparable**
  All grids with same size, attributes are comparable under same standard
- **Controllable**
  Aggregation accuracy is controllable, higher resolution with smaller grids
- **Efficient**
  High computation speed for the matching between grids and GPS data.

- **GPS points**
- **Grids**
- **Data Distribution**

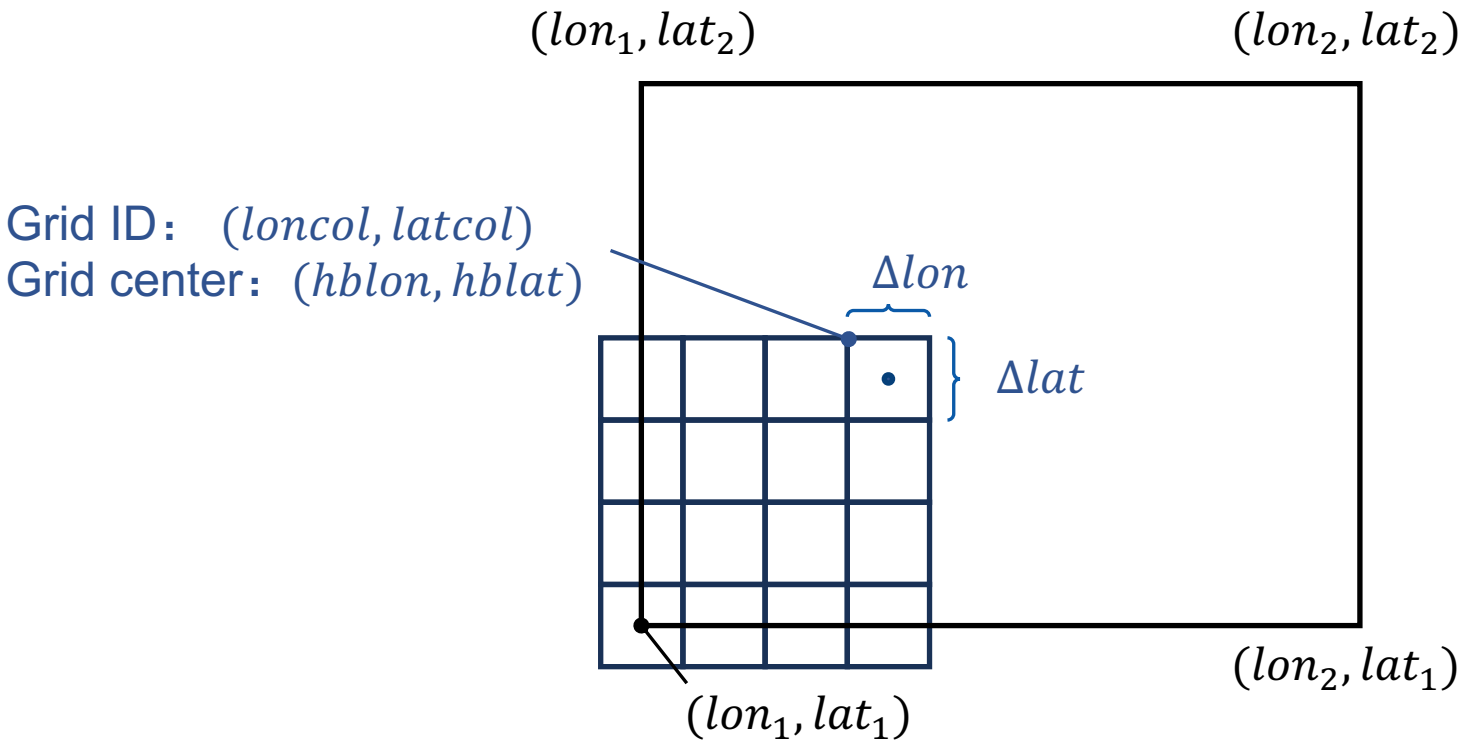


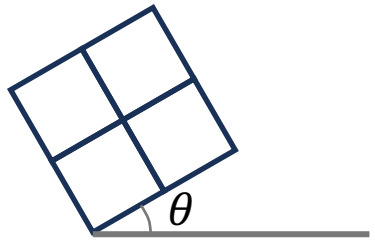
TransBigData offer a framework for grid processing

TransBigData

Griding

➢ **Griding coordinate system**





Optional: Support grids with rotation angle

➢ **Griding params: Define a griding coordinate system**

params=$(lon_1, lat_1, \Delta lon, \Delta lat, \theta)$

TransBigData

Griding methods offered by TransBigData

transbigdata.rect_grids(*location, accuracy=500, params='auto'*)

Generate the grids in given location: bounds or shape

transbigdata.grid_params(*bounds, accuracy=500*)

Generate gridding params from bounds

transbigdata.GPS_to_grids(*lon, lat, params*)

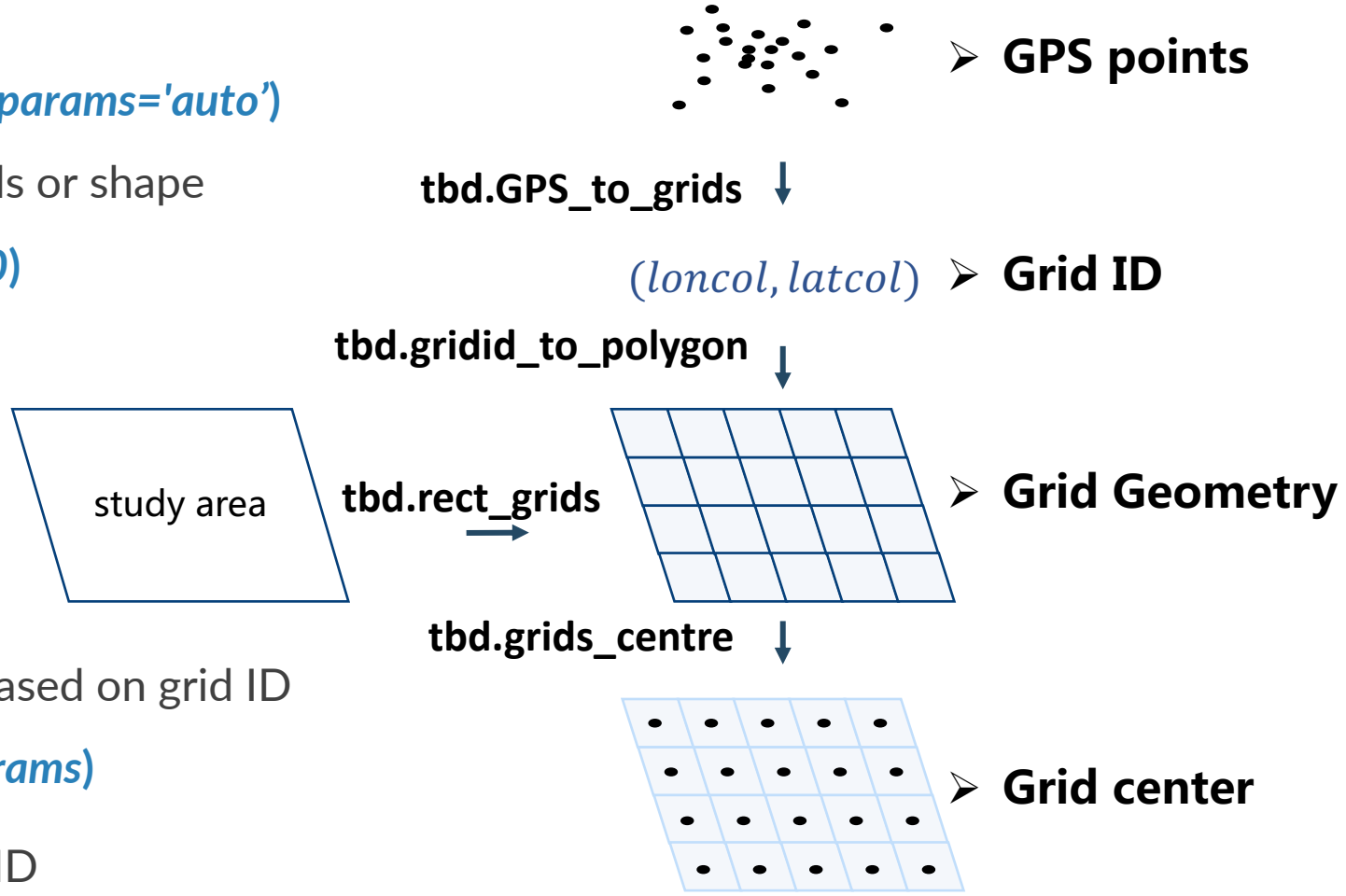
Match the GPS data to the grids

transbigdata.grids_centre(*loncol, latcol, params*)

Calculate the center location of the grids based on grid ID

transbigdata.gridid_to_polygon(*loncol, latcol, params*)

Generate the grid geometry based on grid ID



TransBigData

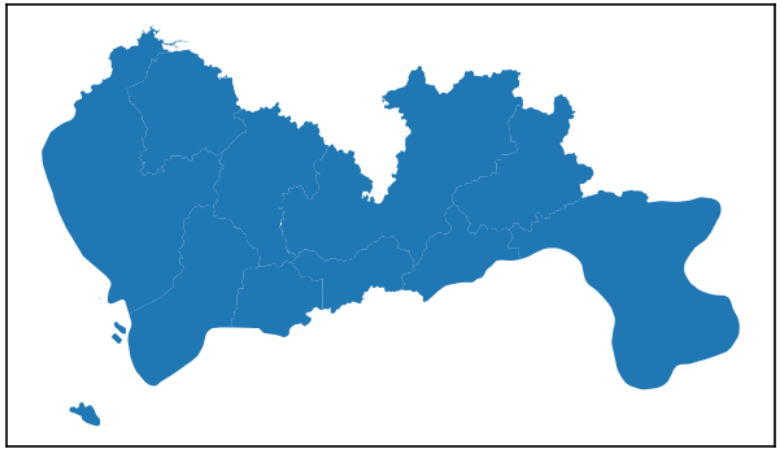
Examples 1: Taxi GPS Data Processing

➢ Data Gridding and Visualization

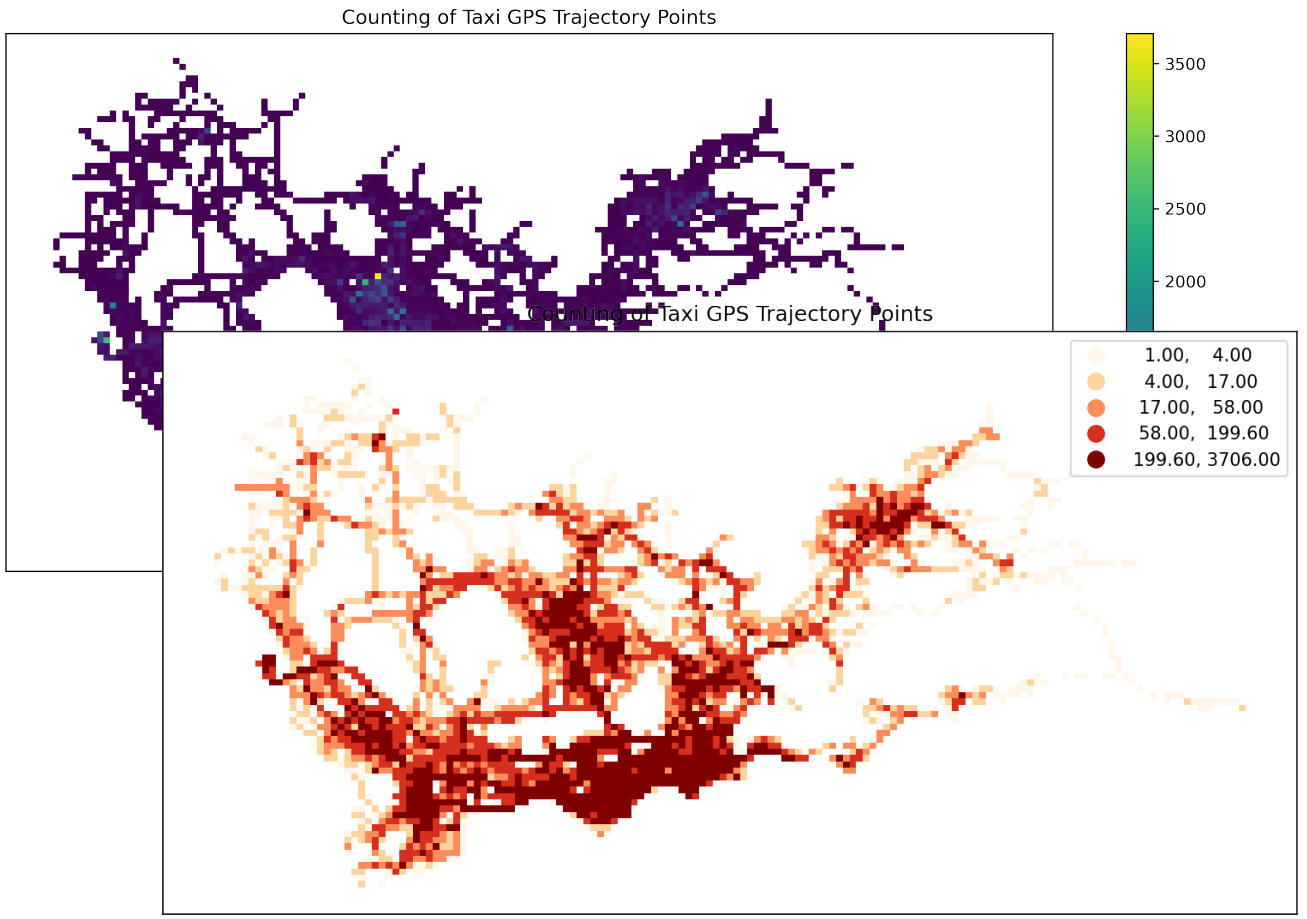
	VehicleNum	Time	Lng	Lat	OpenStatus	Speed
0	34745	20:27:43	113.806847	22.623249	1	27
1	34745	20:24:07	113.809898	22.627399	0	0
2	34745	20:24:27	113.809898	22.627399	0	0
3	34745	20:22:07	113.811348	22.628067	0	0
4	34745	20:10:06	113.819885	22.647800	0	54

Taxi GPS Trajectory Data

+



Geometric Data



Counts of Trajectory Points in Each Grid (Could be any)

TransBigData

Examples 1: Taxi GPS Data Processing

➢ Data Gridding and Visualization

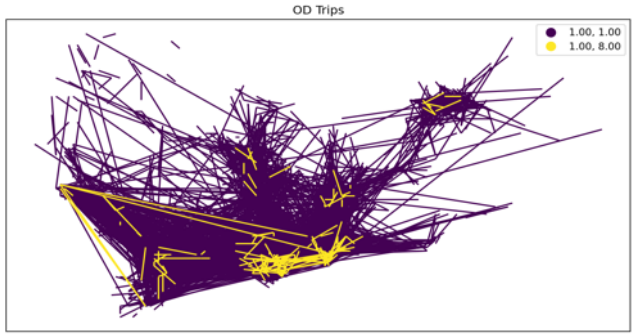
➢ Used Methods

- 剔除界外数据 `transbigdata.clean_outofshape(data, shape, col=['Lng', 'Lat'], accuracy=500)`
- 修正载客状态 `transbigdata.clean_taxi_status(data, col=['VehicleNum', 'Time', 'OpenStatus'], timelimit=None)`
- 生成栅格参数 `tbd.grid_params(bounds, accuracy=500)`
- 轨迹匹配栅格 `transbigdata.GPS_to_grids(lon, lat, params)`
- 栅格地理信息 `transbigdata.gridid_to_polygon(loncol, latcol, params)`

TransBigData

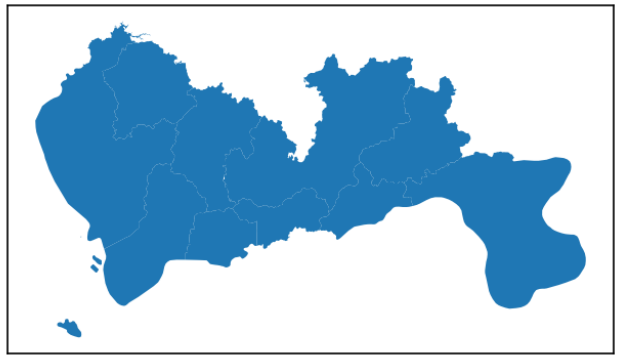
Examples 1: Taxi GPS Data Processing

➢ Individual Trip Extraction, OD Flow Aggregation, Base-map Loading

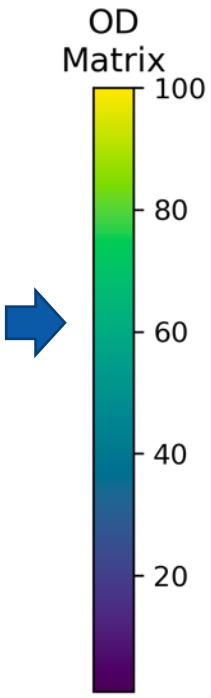


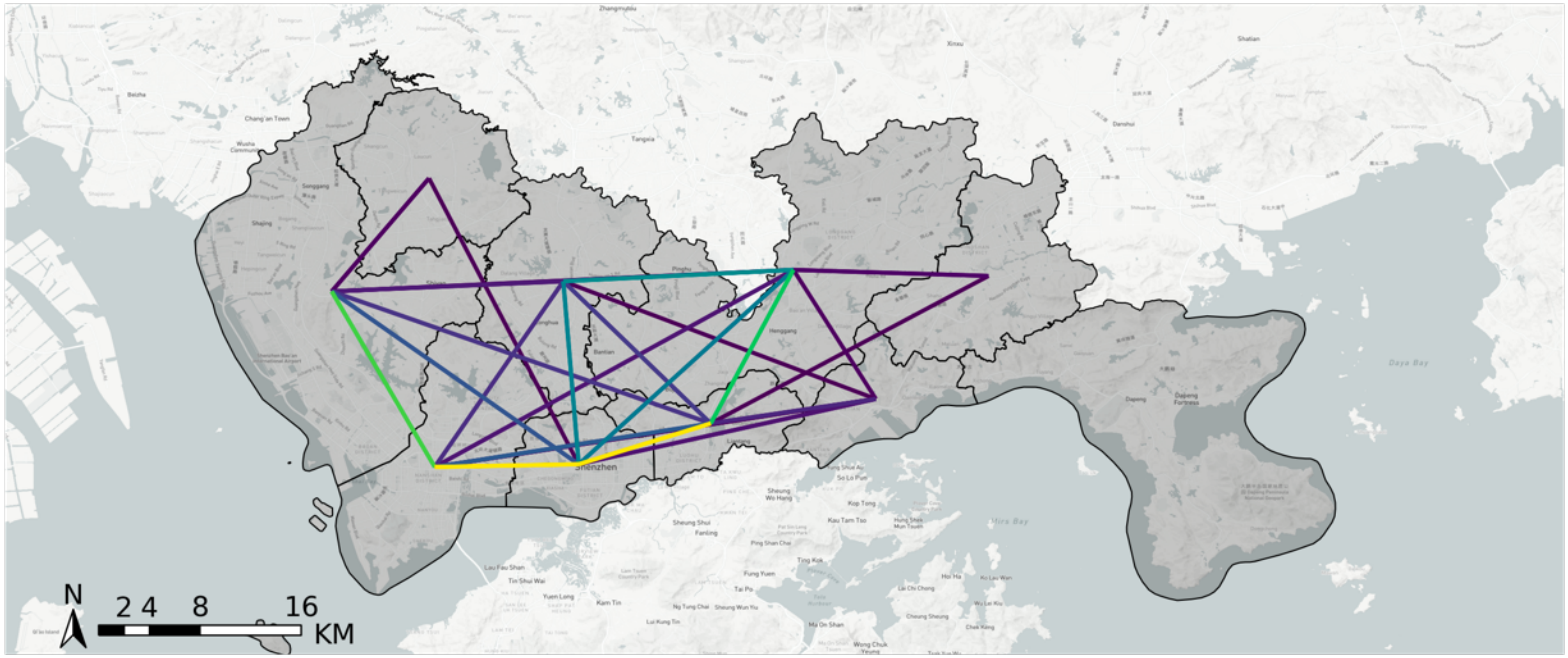
Grid-Level OD Flow

+



Geometric Data
(Traffic Zone Division)





Traffic-zone-level OD Flow, Base-map Loading

TransBigData

Examples 1: Taxi GPS Data Processing

➢ OD Extraction, OD Aggregation, Base-map Loading, Visualization

➢ Used Methods

OD Extraction

- 提取个体OD

```
transbigdata.taxigps_to_od(data, col=['VehicleNum', 'Stime', 'Lng', 'Lat', 'OpenStatus'])
```

- 栅格集计获取栅格OD

```
transbigdata.odagg_grid(oddata, params, col=['slon', 'slat', 'elon', 'elat'], arrow=False)
```

- 区域集计获取区域OD

```
transbigdata.odagg_shape(oddata, shape, col=['slon', 'slat', 'elon', 'elat'], params=None,
round_accuracy=6, arrow=False)
```

Visualization

- 地图底图加载

```
transbigdata.plot_map(plt, bounds, zoom='auto', style=4, printlog=False, styleid='dark')
```

Others

- 载客空驶点位

```
transbigdata.taxigps_traj_point(data, oddata, col=['Vehicleid', 'Time', 'Lng', 'Lat', 'OpenStatus'])
```

TransBigData

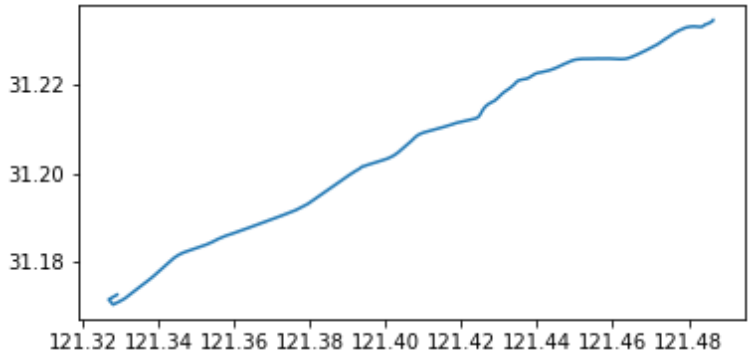
Examples 2: Bus GPS Data Processing

➢ Identification of Bus Arrival/Departure Time

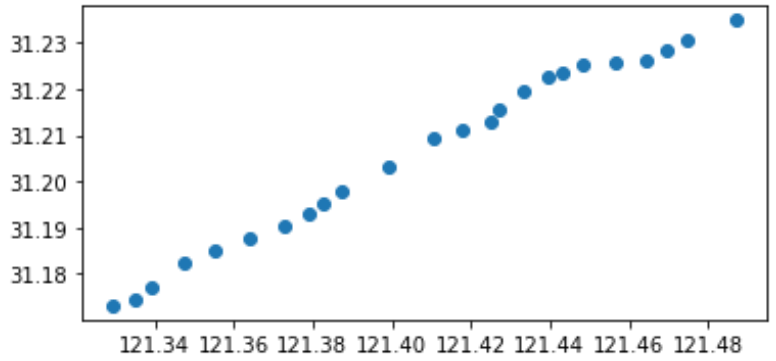
when　　where　　who

	GPSDateTime	LineId	LineName	NextLevel	PrevLevel	StrIatIon	ToDir	VehicleId	VehicleNo	unknow
0	2019-01-16 23:59:59	7100	71	2	1	121.335413,31.173188	1	沪D-R7103	Z5A-0021	1
1	2019-01-17 00:00:00	7100	71	2	1	121.334616,31.172271	1	沪D-R1273	Z5A-0002	1
2	2019-01-17 00:00:00	7100	71	24	23	121.339955,31.173025	0	沪D-R5257	Z5A-0020	1
3	2019-01-17 00:00:01	7100	71	14	13	121.409491,31.20433	0	沪D-R5192	Z5A-0013	1
4	2019-01-17 00:00:03	7100	71	15	14	121.398615,31.200253	0	沪D-T0951	Z5A-0022	1

Column Information of Bus Trajectory Data



Geometric Data of the Bus Line



Geometric Data of the Bus Station (One-way)

TransBigData

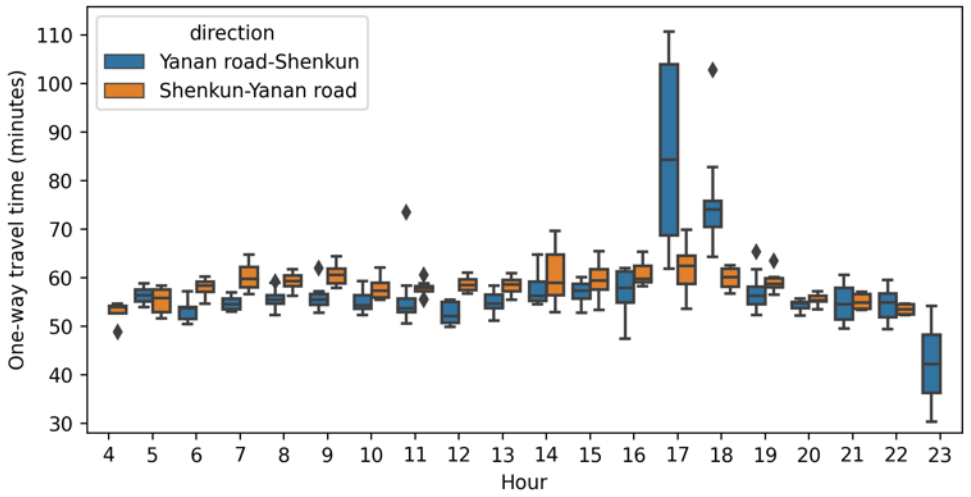
Examples 2: Bus GPS Data Processing

➢ Identification of Bus Arrival/Departure Time

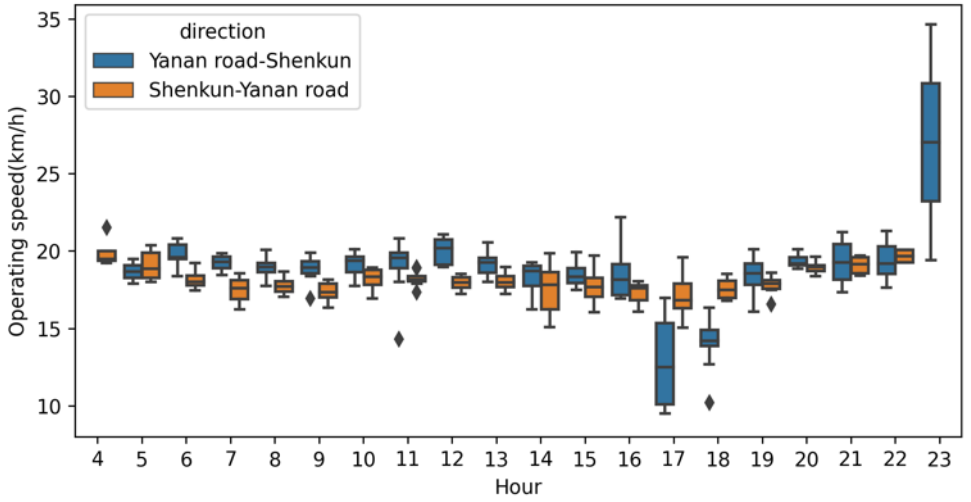
- 提取车辆到离站时刻

```
transbigdata.busgps_onewaytime(arrive_info, start, end,
col=['VehicleId', 'stopname', 'arrivetime', 'leavetime'])
```

	arrivetime	leavetime	stopname	VehicleId
0	2019-01-17 07:19:42	2019-01-17 07:31:14	延安东路外滩	1
1	2019-01-17 09:53:08	2019-01-17 10:09:34	延安东路外滩	1
0	2019-01-17 07:13:23	2019-01-17 07:15:45	西藏中路	1
1	2019-01-17 07:34:24	2019-01-17 07:35:38	西藏中路	1
2	2019-01-17 09:47:03	2019-01-17 09:50:22	西藏中路	1
...	...	...	...	...
2	2019-01-17 16:35:52	2019-01-17 16:36:49	吴宝路	148
3	2019-01-17 19:21:09	2019-01-17 19:23:44	吴宝路	148
0	2019-01-17 13:36:26	2019-01-17 13:45:04	申昆路枢纽站	148
1	2019-01-17 15:52:26	2019-01-17 16:32:46	申昆路枢纽站	148
2	2019-01-17 19:24:54	2019-01-17 19:25:55	申昆路枢纽站	148



One-Way Travel Time

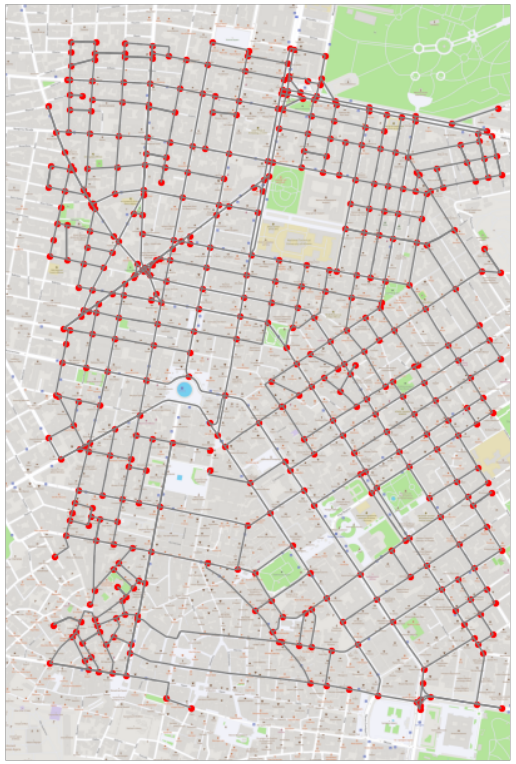


Operation Speed

TransBigData

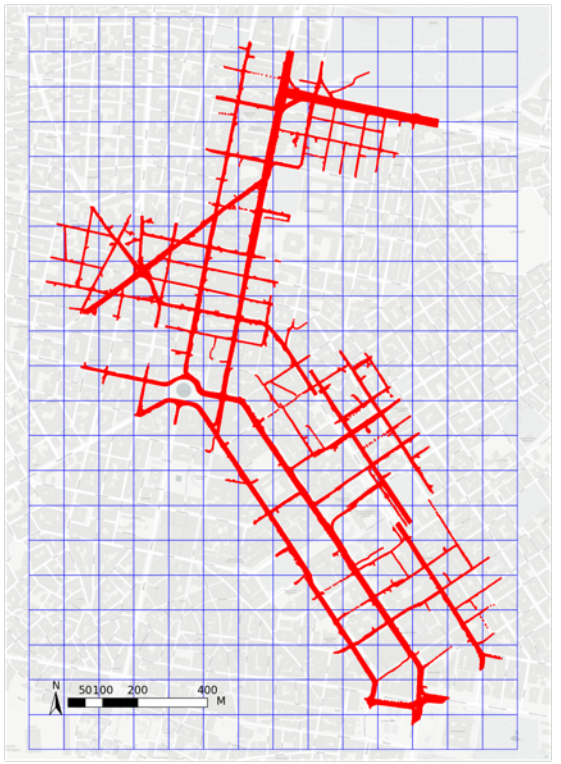
Examples 3: OD/Route Flow Extraction based on the pNEUMA Dataset

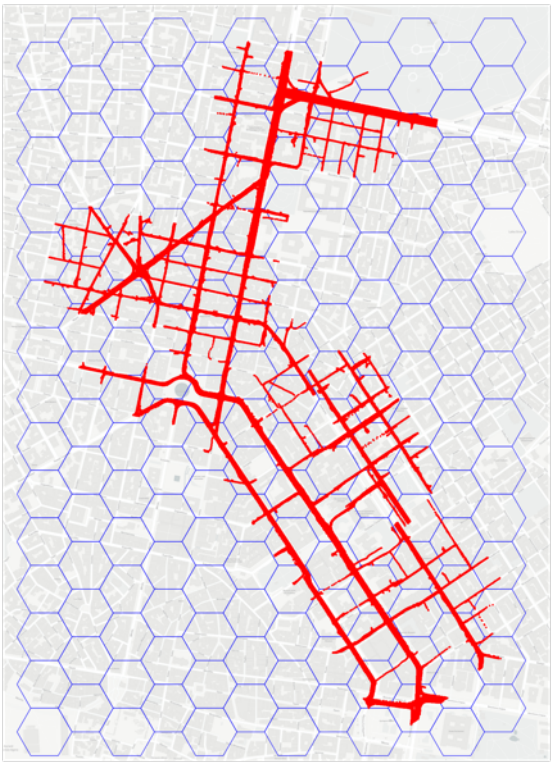
➢ Data Preparation



Base-map Loading (Different Visualization Styles)

(Network topology: from OSMNX)





Data Gridding (Rectangular/Hexagon Approach)

pNEUMA: https://open-traffic.epfl.ch/

TransBigData

Examples 3: OD/Route Flow Extraction based on the pNEUMA Dataset

➢ Data Compression (1/2)

- 采样频率调整 `transbigdata.traj_sparsify(data, col=['Vehicleid', 'Time', 'Lng', 'Lat'], timegap=15)`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 581244 entries, 0 to 581243
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   track_id  581244 non-null  int64
 1   lon       581244 non-null  float64
 2   lat       581244 non-null  float64
 3   speed     581244 non-null  float64
 4   time      581244 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(1)
memory usage: 22.2 MB
None
```

	track_id	lon	lat	speed	time
0	128	23.730362	37.990046	12.5845	1970-01-01 00:00:00.000
1	128	23.730364	37.990045	12.4935	1970-01-01 00:00:00.040
2	128	23.730366	37.990045	12.3965	1970-01-01 00:00:00.080
3	128	23.730367	37.990045	12.2949	1970-01-01 00:00:00.120
4	128	23.730369	37.990044	12.1910	1970-01-01 00:00:00.160

The Origin Sampling Frequency is 0.04 sec

```
# change the sampling frequency to 0.4 second
data_sparsify = tbd.traj_sparsify(data,
                                  col=['track_id', 'time', 'lon', 'lat'],
                                  timegap=0.4,
                                  method='subsample') # do not using interpolate method
print('The number of rows of the de-sampled data: \n')
data_sparsify.info()

# check-points (please do not activate the following code)
# data_sparsify.to_csv('data/data_sparsify.csv', index=None)
```

executed in 1.48s, finished 11:52:55 2022-03-09

```
The number of rows of the de-sampled data:

<class 'pandas.core.frame.DataFrame'>
Int64Index: 23293 entries, 0 to 581229
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   track_id  23293 non-null  int64
 1   lon       23293 non-null  float64
 2   lat       23293 non-null  float64
 3   speed     23293 non-null  float64
 4   time      23293 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(1)
memory usage: 1.1 MB
```

Set the Sampling Frequency to 0.4 sec

TransBigData

Examples 3: OD/Route Flow Extraction based on the pNEUMA Dataset

➢ Data Compression (2/2)

- 去除中间不动点 `transbigdata.clean_same(data, col=['VehicleNum', 'Time', 'Lng', 'Lat'])`

```
# change the sampling frequency to 0.4 second
data_sparsify = tbd.traj_sparsify(data,
                                  col=['track_id', 'time', 'lon', 'lat'],
                                  timegap=0.4,
                                  method='subsample') # do not using interpolate method
print('The number of rows of the de-sampled data: \n')
data_sparsify.info()

# check-points (please do not activate the following code)
# data_sparsify.to_csv('data/data_sparsify.csv', index=None)
```

executed in 1.48s, finished 11:52:55 2022-03-09

```
The number of rows of the de-sampled data:

<class 'pandas.core.frame.DataFrame'>
Int64Index: 23293 entries, 0 to 581229
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   track_id  23293 non-null  int64
 1   lon       23293 non-null  float64
 2   lat       23293 non-null  float64
 3   speed     23293 non-null  float64
 4   time      23293 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(1)
memory usage: 1.1 MB
```

Before

```
# using TBD to clean the stopping points
data_sparsify_clean = tbd.clean_same(data_sparsify, col=['track_id', 'time', 'lon', 'lat'])
print('The number of rows of the compressed data:', len(data_sparsify_clean))

data_sparsify_clean.info()
data_sparsify_clean.head()
```

executed in 87ms, finished 12:22:02 2022-03-09

```
The number of rows of the compressed data: 10674
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10674 entries, 0 to 23292
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   track_id  10674 non-null  int64
 1   lon       10674 non-null  float64
 2   lat       10674 non-null  float64
 3   speed     10674 non-null  float64
 4   time      10674 non-null  object
dtypes: float64(3), int64(1), object(1)
memory usage: 500.3+ KB
```

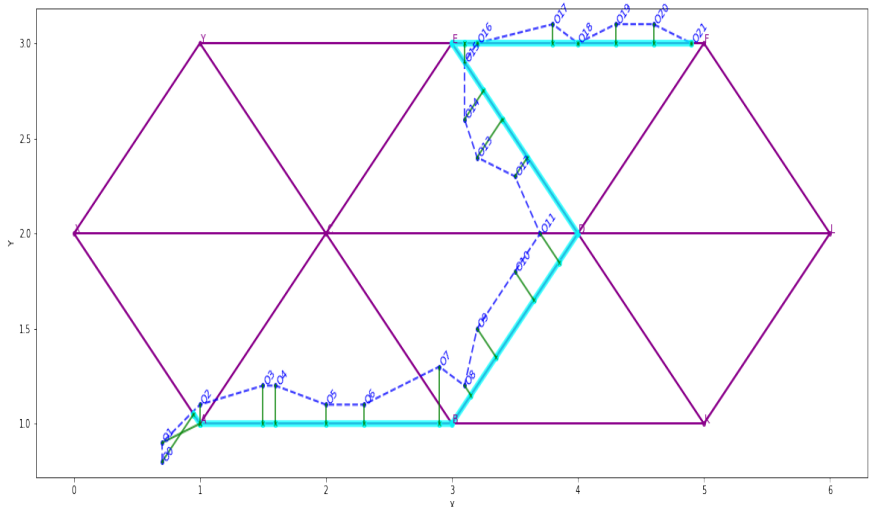
	track_id	lon	lat	speed	time
0	128	23.730362	37.990046	12.5845	1970-01-01 00:00:00.000
1	128	23.730399	37.990040	10.6835	1970-01-01 00:00:01.000
2	128	23.730429	37.990036	7.8580	1970-01-01 00:00:02.000
3	128	23.730443	37.990033	1.2661	1970-01-01 00:00:03.000
71	128	23.730443	37.990033	0.0027	1970-01-01 00:01:11.000

After

TransBigData

Examples 3: OD/Route Flow Extraction based on the pNEUMA Dataset

- **Analysis of Route Choice Behavior** (Combined with map-matching and shortest path searching



Map-matching
using leuvenmapmatching

An Individual Trip

The Shortest Path

TransBigData

Examples 4: Community Detection Based on the Bike Sharing Data

➢ **Data**

	BIKE_ID	DATA_TIME	LOCK_STATUS	LONGITUDE	LATITUDE
0	5	2018-09-01 0:00:36	1	121.363566	31.259615
1	6	2018-09-01 0:00:50	0	121.406226	31.214436
2	6	2018-09-01 0:03:01	1	121.409402	31.215259
3	6	2018-09-01 0:24:53	0	121.409228	31.214427
4	6	2018-09-01 0:26:38	1	121.409771	31.214406

The Origin Form of the Bike Sharing Data
(Only generate records when "LOCK_STATUS" changes)

Geometric Data

- 骑行/静止信息

```
transbigdata.bikedata_to_od(data, col=['BIKE_ID', 'DATA_TIME', 'LONGITUDE', 'LATITUDE', 'LOCK_STATUS'], startend=None)
```

	BIKE_ID	stime	slon	slat	etime	elon	elat
96	6	2018-09-01 0:00:50	121.406226	31.214436	2018-09-01 0:03:01	121.409402	31.215259
561	6	2018-09-01 0:24:53	121.409228	31.214427	2018-09-01 0:26:38	121.409771	31.214406
564	6	2018-09-01 0:50:16	121.409727	31.214403	2018-09-01 0:52:14	121.412610	31.214905
784	6	2018-09-01 0:53:38	121.413333	31.214951	2018-09-01 0:55:38	121.412656	31.217051
1028	6	2018-09-01 11:35:01	121.419261	31.213414	2018-09-01 11:35:13	121.419518	31.213657

Moving Trips (for OD extraction and community detection)

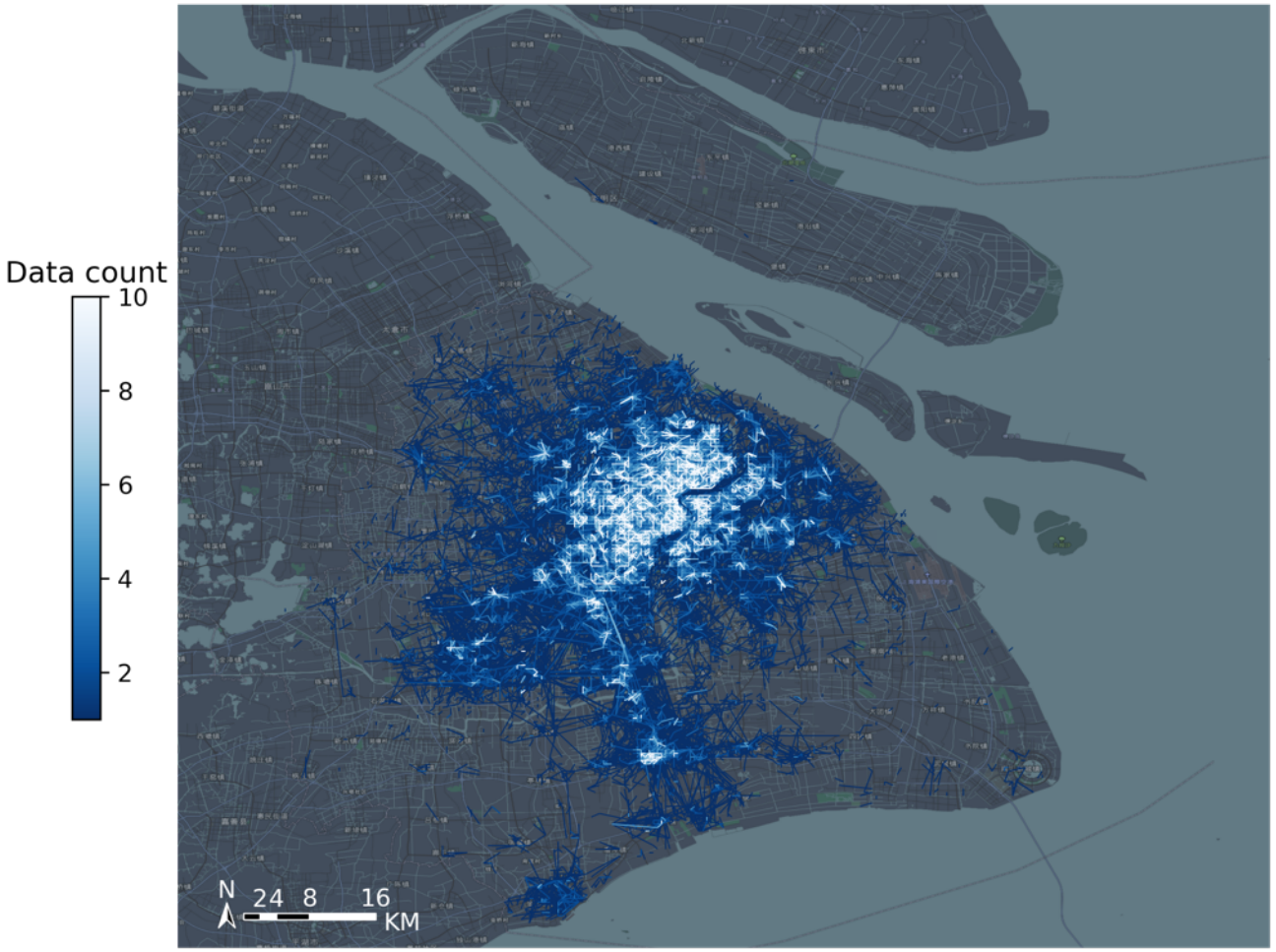
	BIKE_ID	stime	slon	slat	etime	elon	elat
272	6	2018-09-01 0:03:01	121.409402	31.215259	2018-09-01 0:24:53	121.409228	31.214427
562	6	2018-09-01 0:26:38	121.409771	31.214406	2018-09-01 0:50:16	121.409727	31.214403
563	6	2018-09-01 0:52:14	121.412610	31.214905	2018-09-01 0:53:38	121.413333	31.214951
273	6	2018-09-01 0:55:38	121.412656	31.217051	2018-09-01 11:35:01	121.419261	31.213414
1027	6	2018-09-01 11:35:13	121.419518	31.213657	2018-09-01 12:26:22	121.419505	31.213727

Locked Records (for operation analysis and optimization)

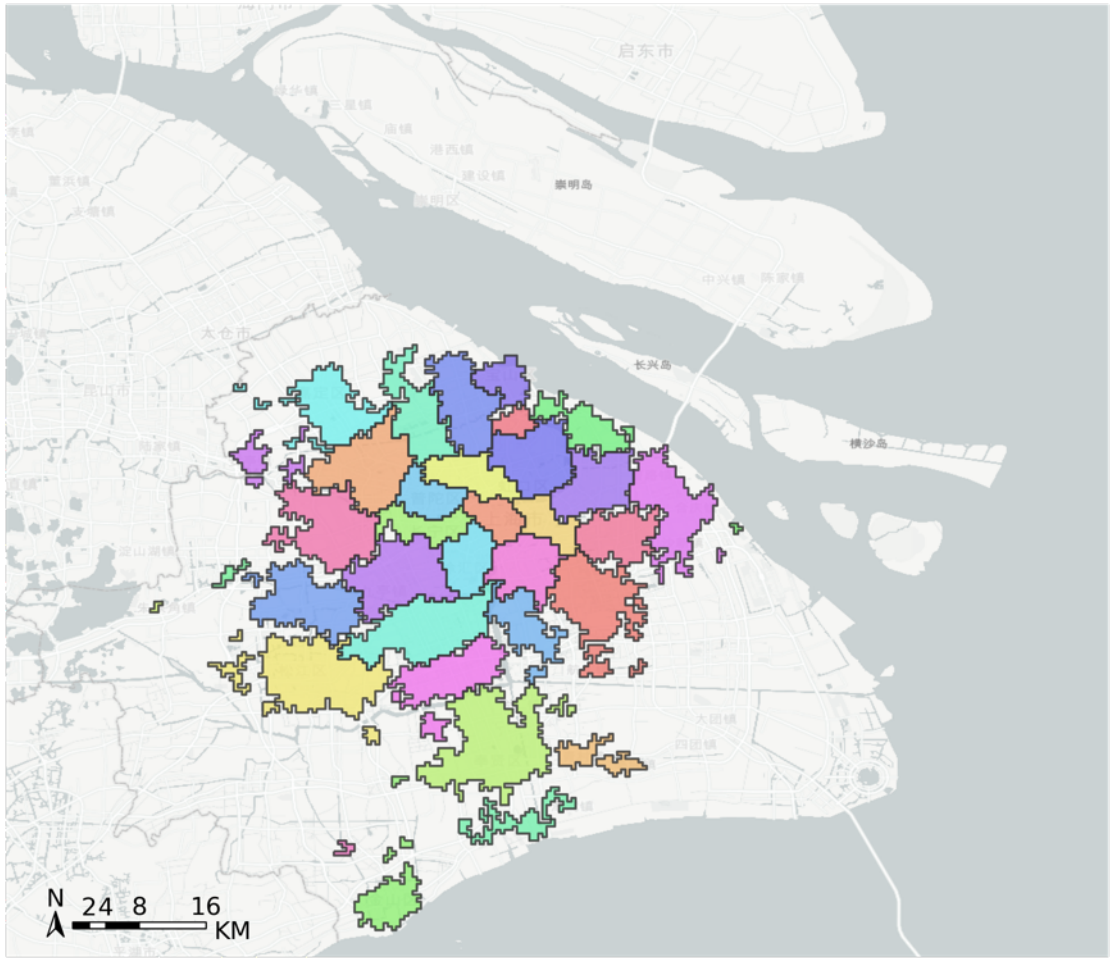
TransBigData

Examples 4: Community Detection Based on the Bike Sharing Data

➢ OD Extraction



➢ Community Detection (combined with iGraph)



https://github.com/ni1o1/transbigdata/tree/main/example

TransBigData

Examples 4: Community Detection Based on the Bike Sharing Data

➢ Used Methods

OD Extraction

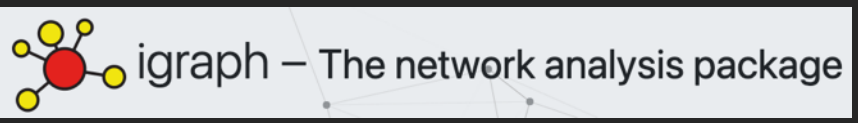
- 生成栅格数据 — tbd.grid_params(bounds, accuracy=500)
- 栅格集计获取栅格OD — transbigdata.odagg_grid(oddata, params, col=['slon', 'slat', 'elon', 'elat'], arrow=False)

Visualization

- 地图底图加载 — transbigdata.plot_map(plt, bounds, zoom='auto', style=4, printlog=False, styleid='dark')
- 指北针比例尺 — transbigdata.plotscale(ax, bounds, textcolor='k', textsize=8, compasssize=1, accuracy='auto', rect=[0.1, 0.1], unit='KM', style=1, **kwargs)

Others

- 计算相对距离 — transbigdata.getdistance(lon1, lat1, lon2, lat2)



同濟大學 TONGJI UNIVERSITY

MAGIC

谢谢，请赐教！

Jian Yuan, Qing Yu

2022/3/12