Practice Session N° 3: Spatial Data

This is the initial version of the document, shared in beginning of the lab session

In this tutorial we will do handle the spatial data using the following tools:

- **Numpy**
  - the fundamental package for [array](https://en.wikipedia.org/wiki/Array_data_structure) computing with Python.
  - [https://www.numpy.org](https://www.numpy.org)

- **Matplotlib**
  - Python plotting package
  - [https://matplotlib.org](https://matplotlib.org)

- **Pandas**
  - Powerful data structures for data analysis, [time series](https://en.wikipedia.org/wiki/Time_series), and statistics
  - [https://pandas.pydata.org](https://pandas.pydata.org)

- **SciPy**
  - Scientific Library for Python
  - [https://www.scipy.org](https://www.scipy.org)

- **RTree**
  - [R-Tree](https://en.wikipedia.org/wiki/R-tree) spatial index (https://en.wikipedia.org/wiki/Spatial_database#Spatial_index) for Python GIS
  - [https://github.com/Toblerity/rtree](https://github.com/Toblerity/rtree)

- **GDAL**
  - Geospatial Data Abstraction Library
  - [http://www.gdal.org](http://www.gdal.org)

- **Fiona**
  - Allows reading and writing different formats of geospatial data
  - [http://github.com/Toblerity/Fiona](http://github.com/Toblerity/Fiona)

- **Shapely**
  - Implements geometric objects, predicates, and operations
  - [https://github.com/Toblerity/Shapely](https://github.com/Toblerity/Shapely)

- **GeoPandas**
  - Geographic extensions for Pandas
  - [http://geopandas.org](http://geopandas.org)

- **PySAL**
  - Spatial Analysis Library for Python
  - [https://pysal.org](https://pysal.org)

- **Missingno**
  - Missing data visualization module for Python
  - [https://github.com/ResidentMario/missingno](https://github.com/ResidentMario/missingno)

Try to read the document and solve the exercises.

The exercises are highlighted in red blocks as follows:
**Task:** This is an example task block.

If you can not solve the task by yourself, join the live session the instructor will demonstrate the solutions.

For some of the tasks, the lab manual will suggest the verification blocks. The instructions in the verification block help to ensure the exercise is complete.

**Task verification:** This is an example verification block for the **Task**

For some of the tasks an extra details may be asked:

**Task Extra:** This an example of an extra question for the **Task**

---

**Installing the required packages**

First, we will recall the possible python-package management facilities we handled during the first lab.

**Task001:**

Make sure the following packages are available in your Python interpreter:

- numpy
- matplotlib
- pandas
- scipy
- rtree
- GDAL
- fiona
- shapely
- geopandas
- PySAL
- missingno

Install the missing packages!

**Task001 verification:** Execute the following cell; the task is complete if there was no error triggered.
Analyzing the synthetic trajectories data

We have prepared the trajectories data based on simulation of urban mobility. In particular, we run the microsimulation of urban traffic based on the road network of Tartu. Several vehicles are performing the navigation with the city given random origin and destination. The simulation recorded the readings as vehicles were progressing in navigation. The readings include vehicle:

- id
- position (x, y in meters)
- bearing (degrees)
- velocity (m/s)
- acceleration (m/s²)

For simplicity of the initial step, we will use meters of x and y in place of degrees for longitude and latitude. Moreover, we shrink our coordinate space, so that point (0, 0) is the left-most-bottom corner of our area of interest. The simple euclidean distance formula is therefore applicable to calculate the distances.

The CSV file is available here:

```
trajectories.csv (trajectories.csv)
```

Let's use our newly installed Python modules to answer a few simple questions:

- How many vehicles were there in the simulation?
- What was the average, maximal, minimal velocity overall?
- What was the average, maximal, minimal velocity per vehicle?
- What is the total traveled distance per vehicle?
- Were there any vehicles that did not move at all?
- Were there any vehicles that did not stop even once?

Let's do a bit of warmup and start with Pandas, Numpy and Matplotlib

Those packages were partially covered in the lectures as well as in the first lab, therefore we expect your participation here ;)

**Task002**: Using Pandas, load the trajectories.csv (trajectories.csv) into a dataframe

In [33]:
```
# TODO: finish the line
df =
```

**Task002 verification**: Execute the following cell; the task is complete if you can see the data

In [34]:
df

Out[34]:
```
   id    x      y    bearing  velocity   acceleration
0   26  563.48  870.36     71.95      0.00            0.00
1   26  565.41  870.99     71.95      2.03            2.03
2   26  566.88  871.47     71.95      1.55           -0.49
3   26  568.10  871.87     71.95      1.29           -0.26
4   26  570.28  872.58     71.95      2.29            1.00
      ...   ...    ...     ...        ...            ...
1501042  993  2630.11 1096.75    66.01      0.00            0.00
1501043  996  1588.54  710.53     38.40      0.00            0.00
1501044  997  1759.05  1718.66   157.70     0.00            0.00
1501045  998  1660.51  812.22     33.16      0.00            0.00
1501046  999  1583.89  704.66     38.40      0.00            0.00
```

1501047 rows × 6 columns

**Task 002 Extra**: Can we count many records were there per vehicle?
In [58]:
# There result should look as follows

Out[58]:

    id  
26   2357
67   2316
81   2302
82   2301
99   2283
...
1589  794
1588  793
1590  793
1591  792
1593  790
Name: id, Length: 1000, dtype: int64

Task003: How many vehicles were there in the simulation?

In [36]:
# TODO: Finish the line
vehicles_count =

Task003 verification: Execute the following cell; the task is complete if you can see 1000

In [38]:
vehicles_count

Out[38]:
1000

Task004: What was the average, maximal, minimal velocity overall? Try to reproduce the output

In [ ]:
# TODO: fill in the cell

Task004 verification: The correct values are as follows:
In [46]:

# Values in m/s

Out[46]:

mean   0.778503
min    0.000000
max    29.060000
Name: velocity, dtype: float64

Task004 Extra: Can we report in km/h instead of m/s?

In [53]:

# Values in km/h

Out[53]:

mean   2.802610
min    0.000000
max    104.616000
Name: velocity, dtype: float64

Task005: What was the average, maximal, minimal velocity per vehicle?

In [ ]:

# TODO: fill in the cell

Task005 verification: The correct values are as follows:
In [54]:

# Values in km/h

Out[54]:

<table>
<thead>
<tr>
<th>id</th>
<th>mean</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>4.149989</td>
<td>0.0</td>
<td>69.660</td>
</tr>
<tr>
<td>67</td>
<td>4.426819</td>
<td>0.0</td>
<td>70.380</td>
</tr>
<tr>
<td>80</td>
<td>13.806473</td>
<td>0.0</td>
<td>81.900</td>
</tr>
<tr>
<td>81</td>
<td>4.757614</td>
<td>0.0</td>
<td>78.516</td>
</tr>
<tr>
<td>82</td>
<td>5.300073</td>
<td>0.0</td>
<td>80.532</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1588</td>
<td>1.915082</td>
<td>0.0</td>
<td>83.160</td>
</tr>
<tr>
<td>1589</td>
<td>4.501360</td>
<td>0.0</td>
<td>50.760</td>
</tr>
<tr>
<td>1590</td>
<td>1.993301</td>
<td>0.0</td>
<td>52.308</td>
</tr>
<tr>
<td>1591</td>
<td>8.302636</td>
<td>0.0</td>
<td>87.336</td>
</tr>
<tr>
<td>1593</td>
<td>8.220167</td>
<td>0.0</td>
<td>86.724</td>
</tr>
</tbody>
</table>

1000 rows × 3 columns

Task005 Extra: Can we sort the values by mean speed in ascending order?

In [56]:

# Values in km/h, sorted by mean in ascending order

Out[56]:

<table>
<thead>
<tr>
<th>id</th>
<th>mean</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>243</td>
<td>4.420435</td>
<td>0.0</td>
<td>28.20</td>
</tr>
<tr>
<td>218</td>
<td>4.011988</td>
<td>0.0</td>
<td>15.82</td>
</tr>
<tr>
<td>100</td>
<td>3.893157</td>
<td>0.0</td>
<td>23.38</td>
</tr>
<tr>
<td>477</td>
<td>3.874957</td>
<td>0.0</td>
<td>24.70</td>
</tr>
<tr>
<td>80</td>
<td>3.835131</td>
<td>0.0</td>
<td>22.75</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1150</td>
<td>0.000000</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>380</td>
<td>0.000000</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>1117</td>
<td>0.000000</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>993</td>
<td>0.000000</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>1345</td>
<td>0.000000</td>
<td>0.0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

1000 rows × 3 columns
**Task006:** What is the total traveled distance per vehicle?

In order to answer this question we have to first calculate the length of trajectory for each vehicle. Our $x$ and $y$ coordinates are given in meters on Euclidean plane.

*hint:* you can reuse your distance formula from the first lab.

In [64]:

```python
# TODO: fill in the cell
id_traj_d =
```  

**Task006 verification:** The correct values are as follows:

In [71]:

```python
id_traj_d.sort_values(ascending=False)
```

Out[71]:

<table>
<thead>
<tr>
<th>id</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>243</td>
<td>4059.622209</td>
</tr>
<tr>
<td>115</td>
<td>3682.987899</td>
</tr>
<tr>
<td>82</td>
<td>3377.952576</td>
</tr>
<tr>
<td>222</td>
<td>3365.887032</td>
</tr>
<tr>
<td>218</td>
<td>3322.392993</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>618</td>
<td>0.000000</td>
</tr>
<tr>
<td>796</td>
<td>0.000000</td>
</tr>
<tr>
<td>1345</td>
<td>0.000000</td>
</tr>
<tr>
<td>1421</td>
<td>0.000000</td>
</tr>
<tr>
<td>1117</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Name: id, Length: 1000, dtype: float64

**Task007:** Were there any vehicles that did not move at all?

In [76]:

```python
# TODO: fill in the cell
id_not_moved =
```

**Task007 verification:** The correct values are as follows:
In [79]:

len(id_not_moved)

Out[79]:

12

In [80]:

id_not_moved.index.values

Out[80]:

array([ 380,  618,  687,  694,  796,  896,  993, 1117, 1127, 1150, 1345, 1421])

**Task007 Extra:** Those vehicles did not travel any distance, but how many records were there for those vehicles?

In [91]:

```python
# TODO: fill in the cell
id_not_moved_records =
```

In [92]:

```
id_not_moved_records

Out[92]:

    id
  id  1844
  380  1765
  618  1696
  687  1689
  694  1587
  796  1487
  896  1390
  993  1266
 1117  1256
 1127  1222
 1150  1038
 1345   962
 1421

Name: id, dtype: int64
```

**Task008:** Were there any vehicles that did not stop even once?

In [104]:

```python
# TODO: improvise the cell
```
**Task008 verification**: All vehicles did stop at least once, in fact all vehicles stopped once destination was reached (the readings however were still recorded till the end of the).

Next let’s recall the plotting using matplotlib

```python
In [110]:
from matplotlib import pylab as plt
```

**Task009**: Plot the histogram of events distribution per vehicle

**Task009 verification**: Histogram is drawn below

```python
In [ ]:
# Improvise the cell
```
Task010: Consider the average speed and traveled distance per vehicle, draw the boxplots

Task010 verification: Histograms are drawn below

In [ ]:

# TODO improvise the cell
Task010 Extra: Can we add another boxplot illustrating travel time per vehicle?

Task011: Draw the longest trajectory, indicate moment velocity using color schema

Task011 verification: Histogram is drawn below

In []:

# Improvise the cell
Task011 Extra: Can we draw trajectories of 10 fastest vehicles onto 5x2 grid of figures
In [151]:

![Graphs showing time series data with multiple lines and axes.](image-url)
**Task012:** Draw the 10 longest trajectories overlayed on one plot

**In [164]:**

```python
# Improvise the cell
```

**In [163]:**

```python

```

**Out[163]:**

<matplotlib.legend.Legend at 0x7fba9c3c1ba8>

---

**Task012 Extra:** Can we draw it in 3d to give better overview of overlaying segments
Let's conclude with the warmup, and proceed to more sophisticated modules

The exercises we have solved before were mostly solvable using Numpy, Pandas and Matplotlib. Next we are going to try out the other modules from our list.

SciPy
In [178]:

    import scipy as sc
    sc.__version__

Out[178]:

    '1.6.1'

In [194]:

    # Will be shown during the lab

In [196]:

    # Quick pairwise distance map and distance based clustering
    from scipy.spatial.distance import cdist

In [201]:

Out[201]:

    <matplotlib.colorbar.Colorbar at 0x7fbaa2ae5438>