Application of Artificial Intelligence in Road Traffic: Data Mining and Clustering

Author:
Zaid Khan – A3YI00

Supervisor:
Balázs Varga, PhD

MSc Transportation Engineering

Faculty of Transportation Engineering and Vehicle Engineering

Budapest University of Technology and Economics

zaid6064kp@gmail.com

Outline

- Introduction
- Data Mining and Clustering
- Data Sources
- Research Algorithm
- Data Pre-Processing
- Data Processing
- Data Analysis
- Application of Algorithm
- Future Potential

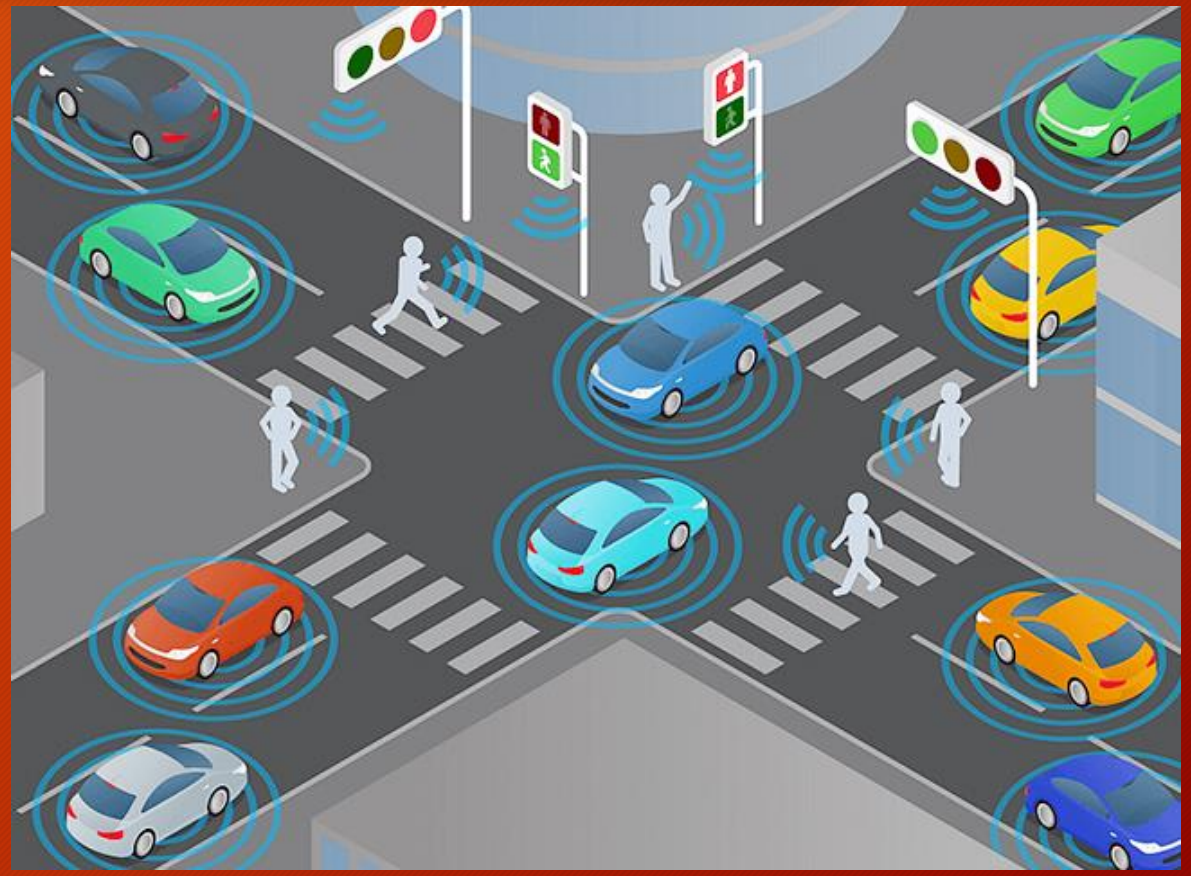
Introduction:
Artificial Intelligence

- Machines to replicating the human behavior and activities.
- Comes into place where the conventional computational techniques are failed.
- Adopted by businesses to enhance productivity.

Introduction:
Artificial Intelligence in Transportation

- Accessibility of massive amount of **quantitative** and **qualitative** data.

- Applications:
  - Travel demands forecast
  - CO2 Emission calculation
  - Safety enhancement
  - User experience enhancement
  - Transport infrastructure improvement
  - Resolve jams and congestion problems
  - Many other

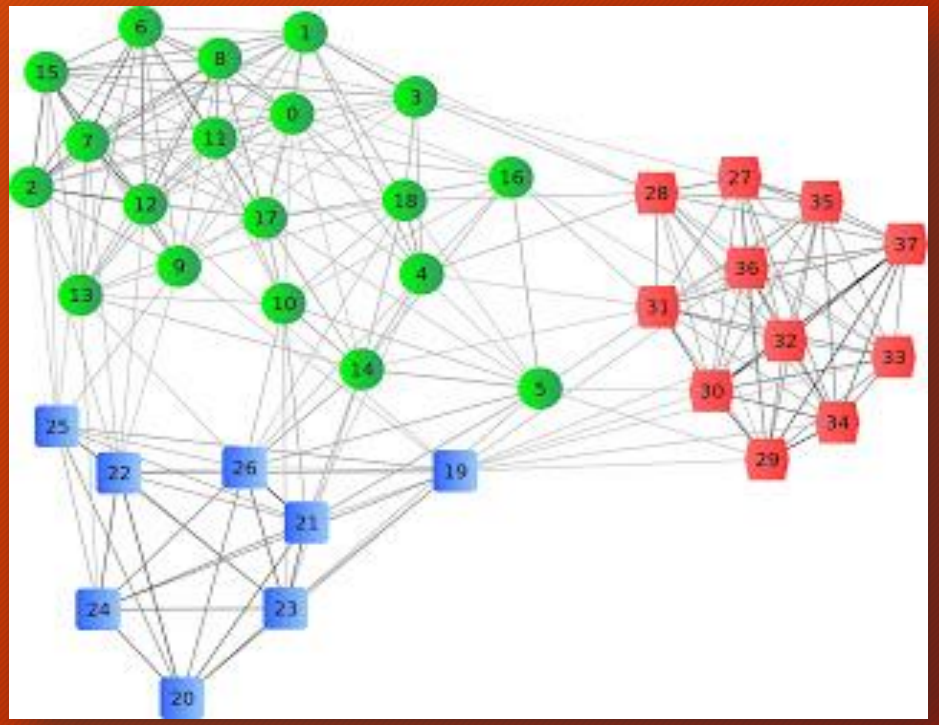


Data Mining and Clustering

Data Mining

It refers to extracting or “mining” knowledge from large amounts of data.

Data Clustering

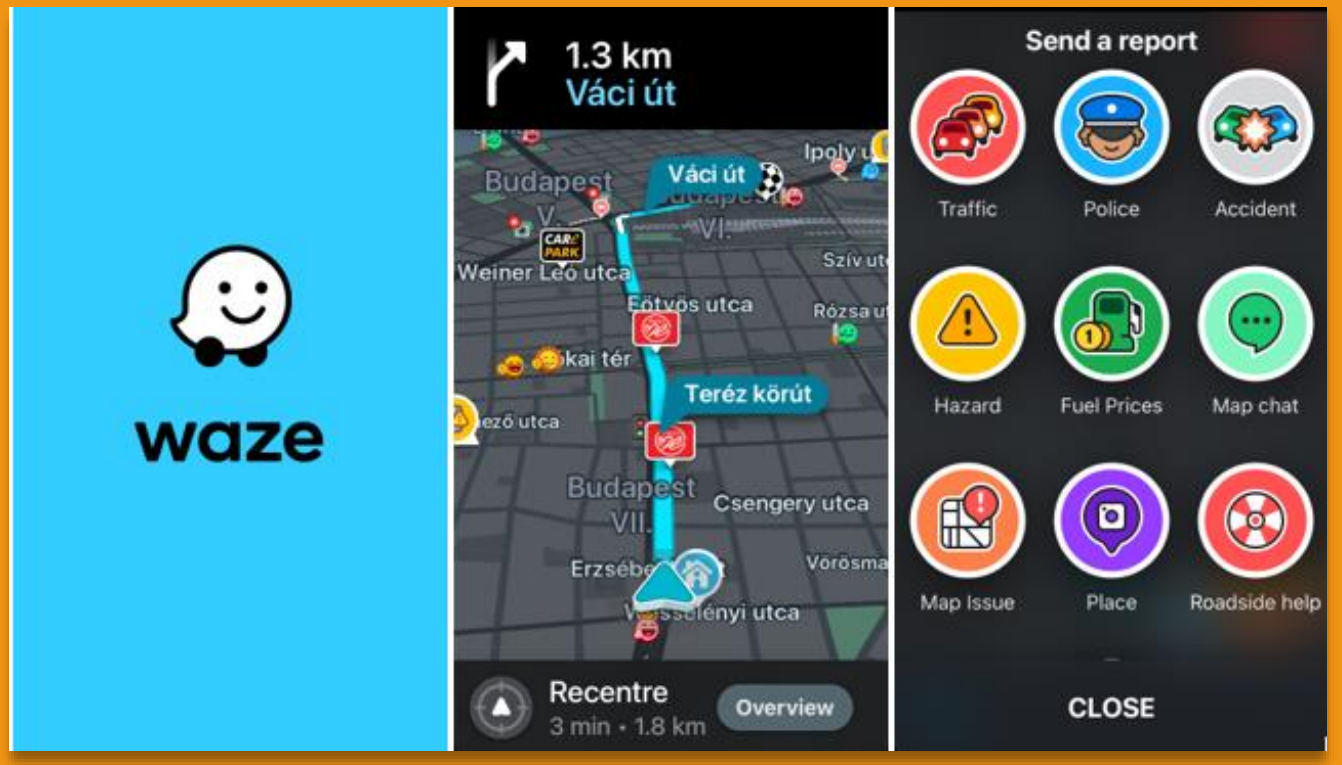


The goal is to segregate groups with similar characteristics and allocate them into clusters.

Data Sources:
Waze and Trafmine

- **WAZE:** is a GPS navigation software app owned by Google
- Uses crowd sourcing
- Real-time traffic information
- More sophisticated features

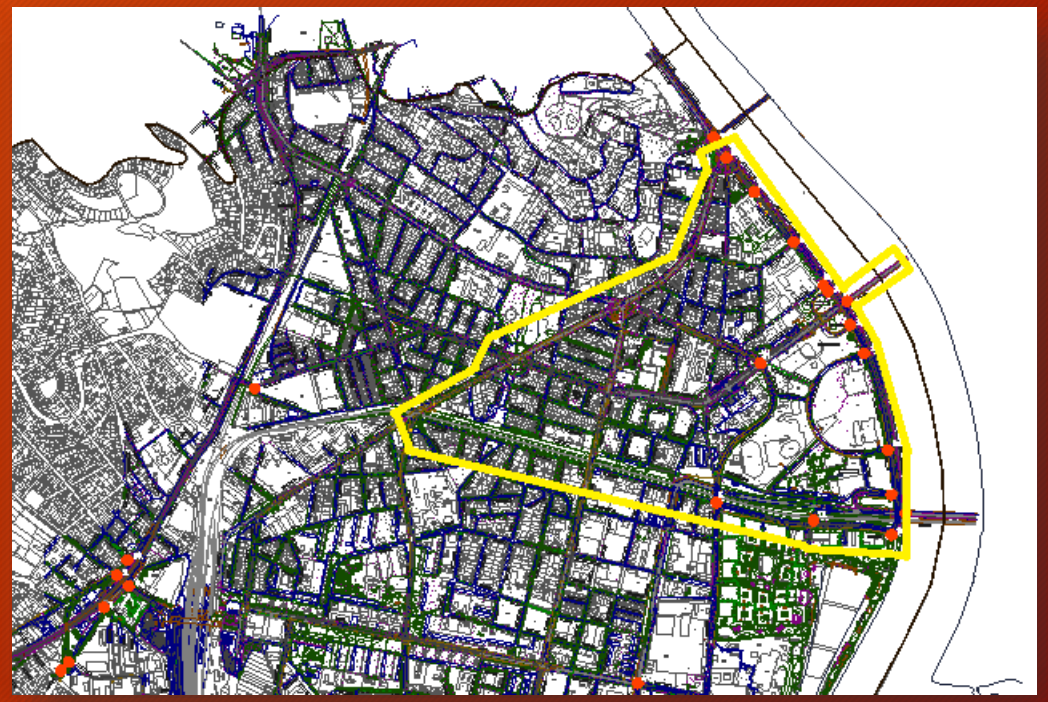
- **Trafmine:** is web tool which allows the subscribers to download the data that is collected by WAZE app



Data Sources:
Waze and Trafmine


```
{"alerts":[{"uuid":"79284b68-8437-3696-9ee8-
e349ff8e3b87","alert_type_id":3,"alert_subtype_id":null,"country":"HU","report_rating":2,"confidence":0,"
reliability":5,"magvar":285,"location":[19.035818,47.472594],"publish_date":"2019-01-
01T01:35:25+00:00","last_seen":"2019-01-01T02:05:16+00:00","city":"Budapest
XI.","road_type":1,"street":"Sárbogárdi
út","report_description":null,"thumbs_up":0,"report_by_partner":null,"jam_uuid":null},{"uuid":"3bc6bfbf-
807e-336b-a71c-
ef5c0727d441","alert_type_id":4,"alert_subtype_id":10,"country":"HU","report_rating":4,"confidence":0,"re
liability":6,"magvar":98,"location":[19.047134,47.473903],"publish_date":"2019-01-
01T08:12:03+00:00","last_seen":"2019-01-01T08:40:19+00:00","city":"Budapest
XI.","road_type":null,"street":"Október huszonharmadika
utca","report_description":null,"thumbs_up":0,"report_by_partner":null,"jam_uuid":null},
{"uuid":"9675c6ff-0cf8-3e67-88e7-
94c16d31e956","alert_type_id":3,"alert_subtype_id":3,"country":"HU","report_rating":0,"confidence":0,"rel
iability":5,"magvar":291,"location":[19.041656,47.473943],"publish_date":"2019-01-
01T09:21:49+00:00","last_seen":"2019-01-01T09:50:19+00:00","city":"Budapest
XI.","road_type":1,"street":"Ulászló
utca","report_description":null,"thumbs_up":0,"report_by_partner":null,"jam_uuid":null},
{"uuid":"000ec974-9ea1-3e46-9d41-
```


Part of Trafmine provided JSON data



Target Area:
Part of 11th District Budapest

Data Sources:
Open Street Map

- Data from Open Street Map official website

- Latitude Bounds:
- 47.46770 and 47.48430

- Longitude Bounds:
- 19.03310 and 19.06820

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.8.3 (896248
spike-08.openstreetmap.org)" copyright="OpenStreetMap and
contributors"
attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
 <bounds minlat="47.4677000" minlon="19.0331000"
maxlat="47.4843000" maxlon="19.0682000"/>
 <node id="277465" visible="true" version="7"
changeset="44669370" timestamp="2016-12-25T20:29:52Z"
user="urbalazs" uid="906236" lat="47.4820450" lon="19.0525080"/>
 <node id="277466" visible="true" version="7"
changeset="47110170" timestamp="2017-03-23T23:06:42Z"
user="BÃ¡thoryPÃ©ter" uid="408450" lat="47.4809680"
lon="19.0520464"/>
```

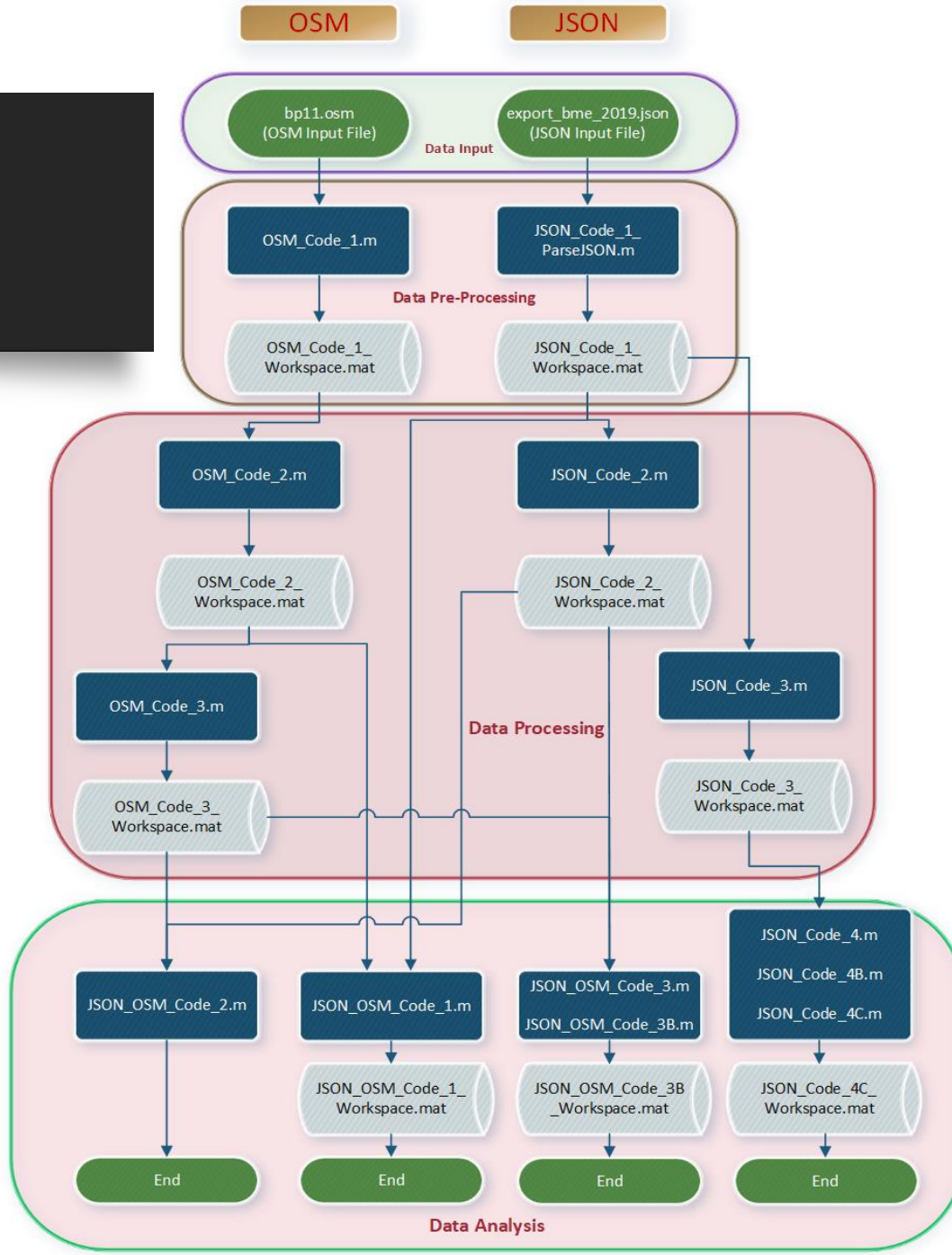
Part of XML-formatted OSM data

Research Algorithm

- Components
- Connections
- Workflow

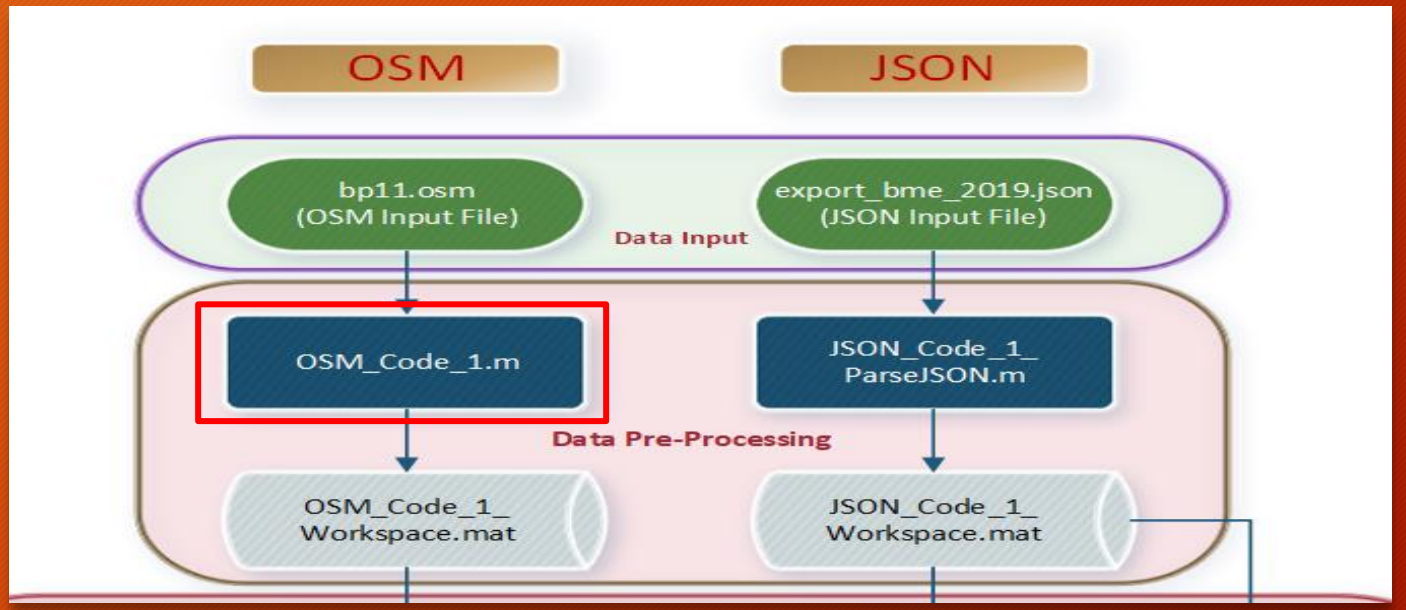
4 Parts

1. Data Input
2. Data Pre-Processing
3. Data Processing
4. Data Analysis



Data Pre-Processing:
OSM_Code_1

- To convert XML formatted data to MATLAB Structure.
- Extracts connectivity matrix.
- Plot routes on figures
- Main Variable ‘parsed_osm’

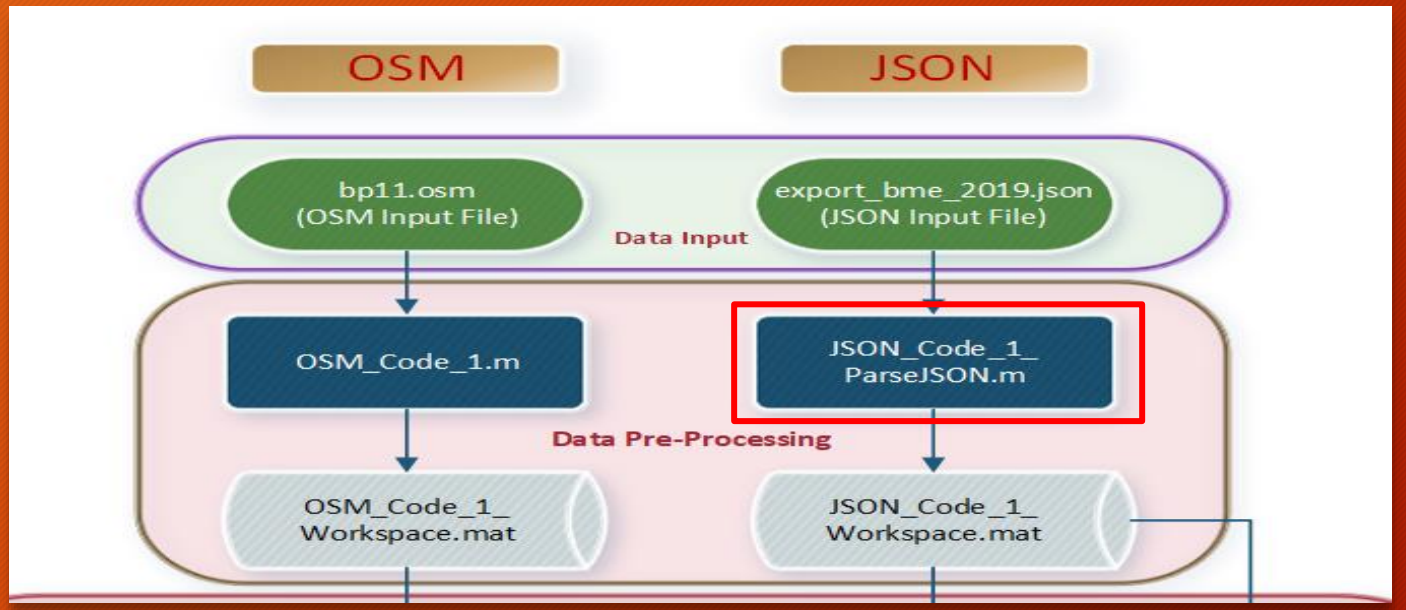


OSM_Code_1.m

```
%% Name File
openstreetmap_filename = 'bp11.osm';

%% Convert XML to MATLAB Structure
[parsed_osm,osm_xml] = parse_openstreetmap(openstreetmap_filename);

%% Connectivity
[connectivity_matrix,intersection_node_indices] = extract_connectivity(parsed_osm);
intersection_nodes = get_unique_node_xy(parsed_osm,intersection_node_indices);

%% Plan a Route without assuming one way roads
start = 1 ;
target = 9 ;
dg = or(connectivity_matrix,connectivity_matrix.'); %symmetry
[route,dist] = route_planner(dg,start,target);

```

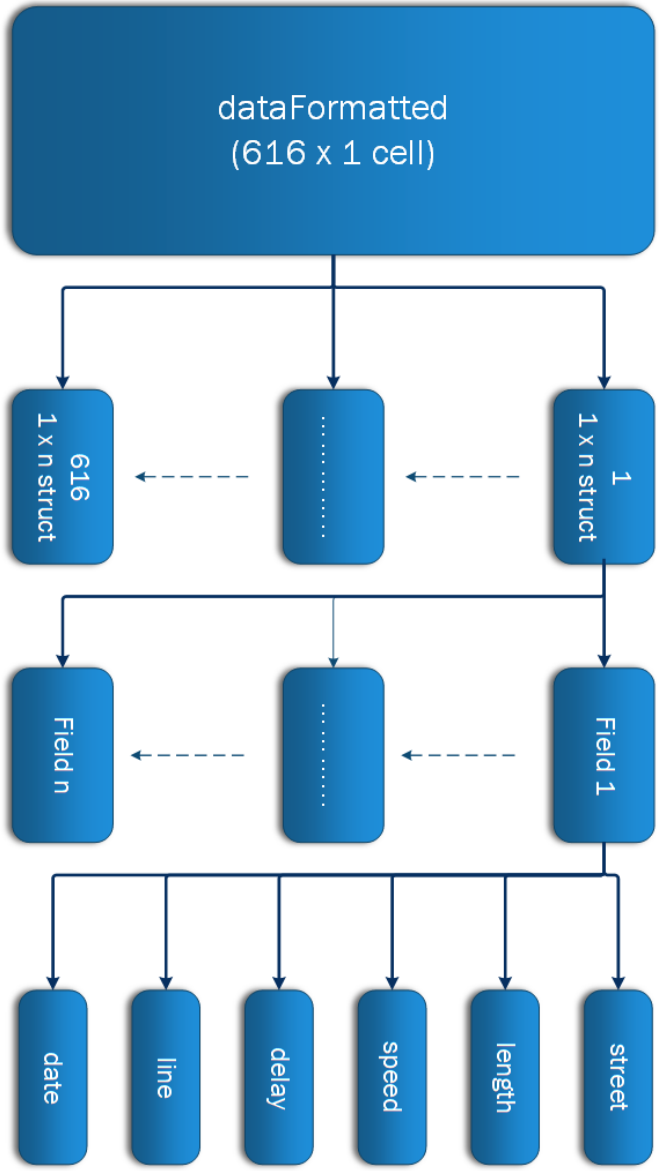
Data Pre-Processing:
JSON_Code_1_ParseJSON

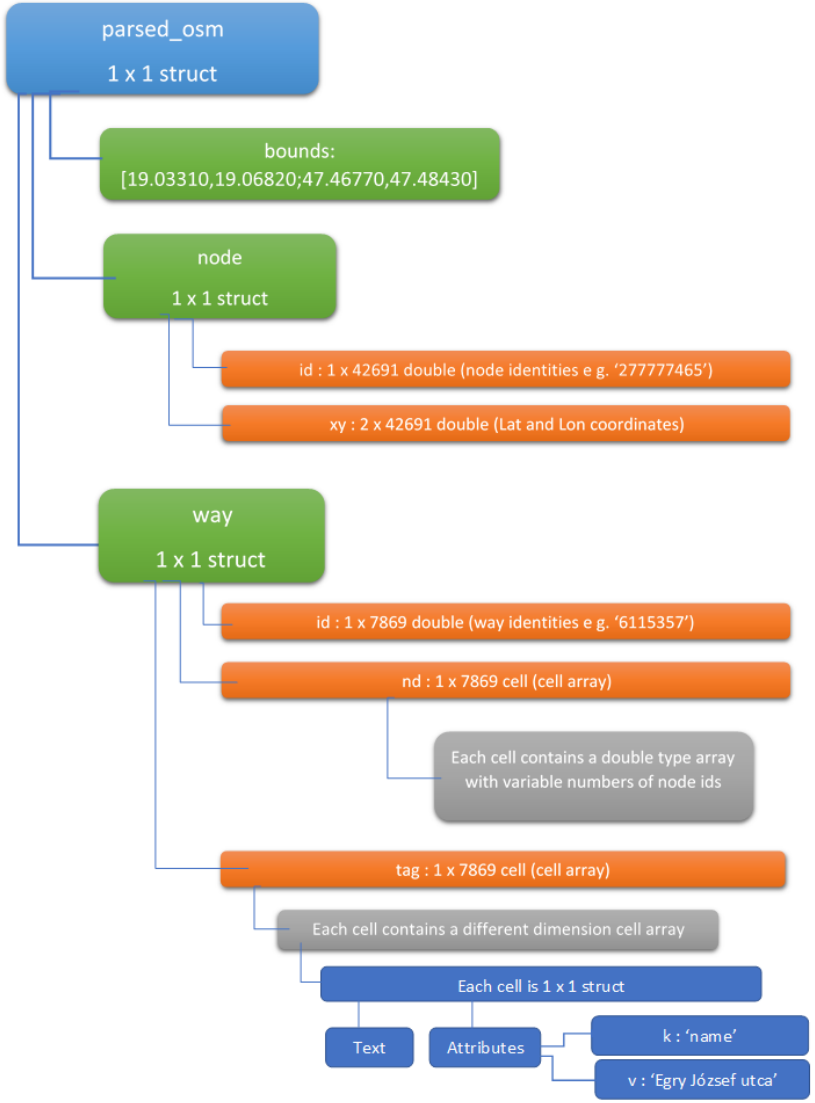
- To convert JSON data into understandable structure
- Data about Alerts and Jams
- Main variable ‘dataFormatted’



ParseJSON.m

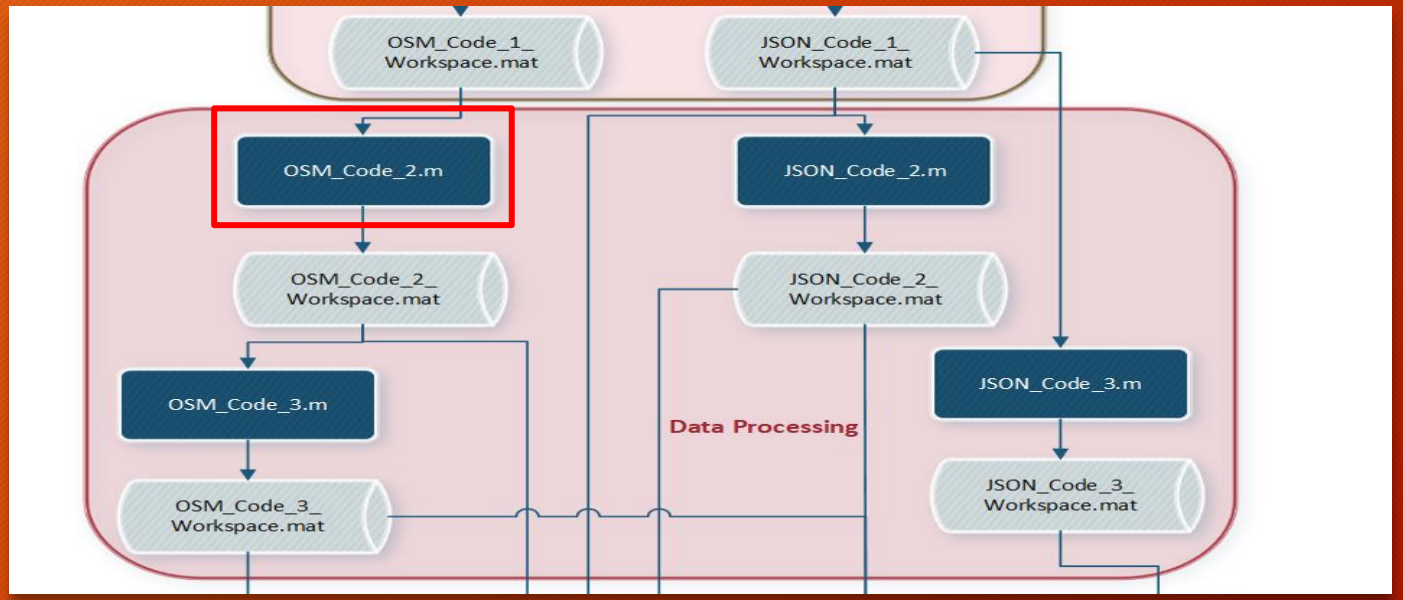
```
21 -    clear dataFormatted;
22 -    k = 0;
23 -    dataFormatted = {};
24 -    for i = 1:length(data.jams)
25
26 -        ctr = 0;
27 -        for l = 1:length(dataFormatted)
28
29 -            if strcmp(dataFormatted{l}(1).street, data.jams{i,1}.street)
30 -                n = size(dataFormatted{l},2);
31
32 -                try
33 -                for j = 1:length(data.jams{i,1}.revisions)
34 -                    dataFormatted{l,1}(n+j).street = char(data.jams{i,1}.street);
35 -                    dataFormatted{l,1}(n+j).length = data.jams{i,1}.revisions(j).length;
36 -                    dataFormatted{l,1}(n+j).speed = data.jams{i,1}.revisions(j).speed;
37 -                    dataFormatted{l,1}(n+j).delay = data.jams{i,1}.revisions(j).delay;
38 -                    dataFormatted{l,1}(n+j).line = data.jams{i,1}.revisions(j).line;
39 -                    dataFormatted{l,1}(n+j).date = data.jams{i,1}.revisions(j).created_at;
40 -                end
```


dataFormatted
(616 x 1 cell)
616
1 x n struct
1
1 x n struct
Field n
Field 1
date
line
delay
speed
length
street


parsed_osm
1 x 1 struct
bounds:
[19.03310,19.06820;47.46770,47.48430]
node
1 x 1 struct
id : 1 x 42691 double (node identities e g. '277777465')
xy : 2 x 42691 double (Lat and Lon coordinates)
way
1 x 1 struct
id : 1 x 7869 double (way identities e g. '6115357')
nd : 1 x 7869 cell (cell array)
Each cell contains a double type array with variable numbers of node ids
tag : 1 x 7869 cell (cell array)
Each cell contains a different dimension cell array
Each cell is 1 x 1 struct
Text
Attributes
k : 'name'
v : 'Egry József utca'

Data Processing:
OSM_Code_2

- Input Variable: ‘parsed_osm’
- Operations:
  - Extract Nodes’ Data
  - Extract Ways’ Data
  - Plot Ways
- Output Variable: ‘WayData’

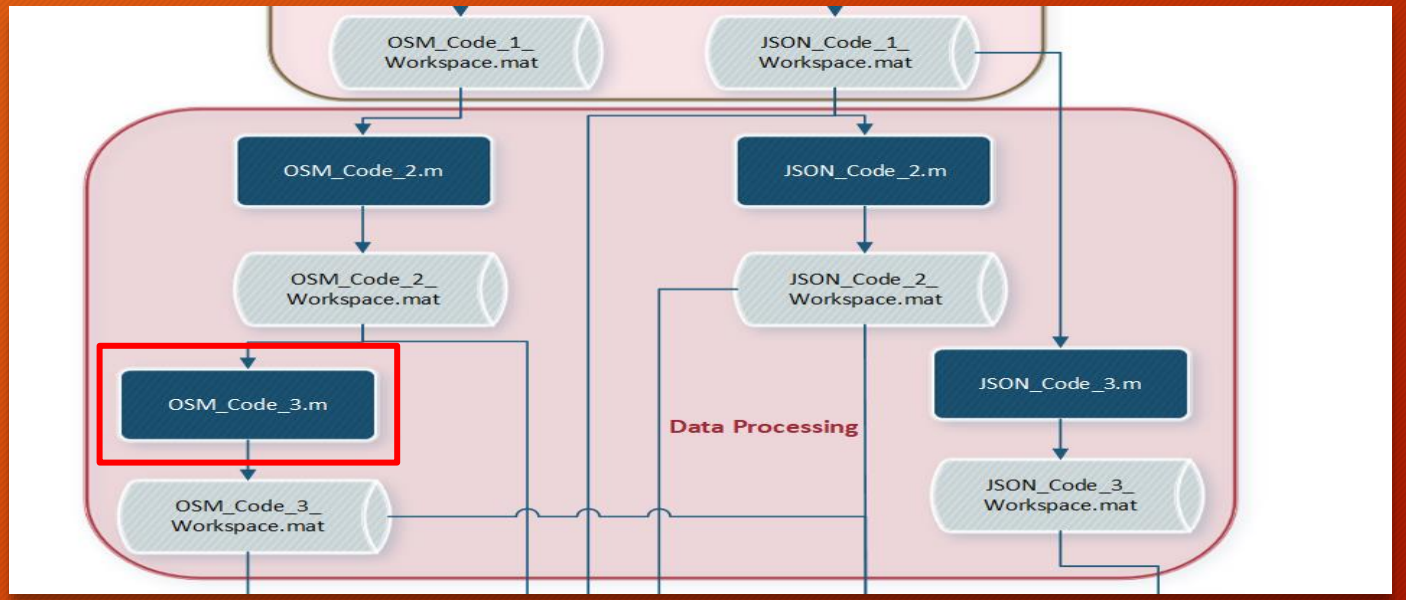


OSM_Code_2.m

```
%% Loading Variables from OSM_Code_1_Workspace
load OSM_Code_1_Workspace.mat parsed_osm

%% Extracting Nodes
nodeIds = (parsed_osm.node.id)';
nodeCoords = (parsed_osm.node.xy)';
nodeData = [nodeIds , nodeCoords];

%% Extracting Ways:
% 1)Way IDs ; 2)Node IDs corresponding Each Way ; 3)Coordinates ;
% 4)Attributes ; 5)Street Names ; 6)Plot ; 7) Master Cell Array

WayData = [];
figure
for WayNo = 1:length(parsed_osm.way.id)

    %Way IDs
    wayId = (parsed_osm.way.id(WayNo))';
```

Data Processing:
OSM_Code_3

- Extended operation on OSM_Code_2
- Input Variable: ‘WayData’
- Operation: Gathers all coordinates in single matrix by vertical concatenation
- Output Variable: ‘All_OSM_Coordinates’

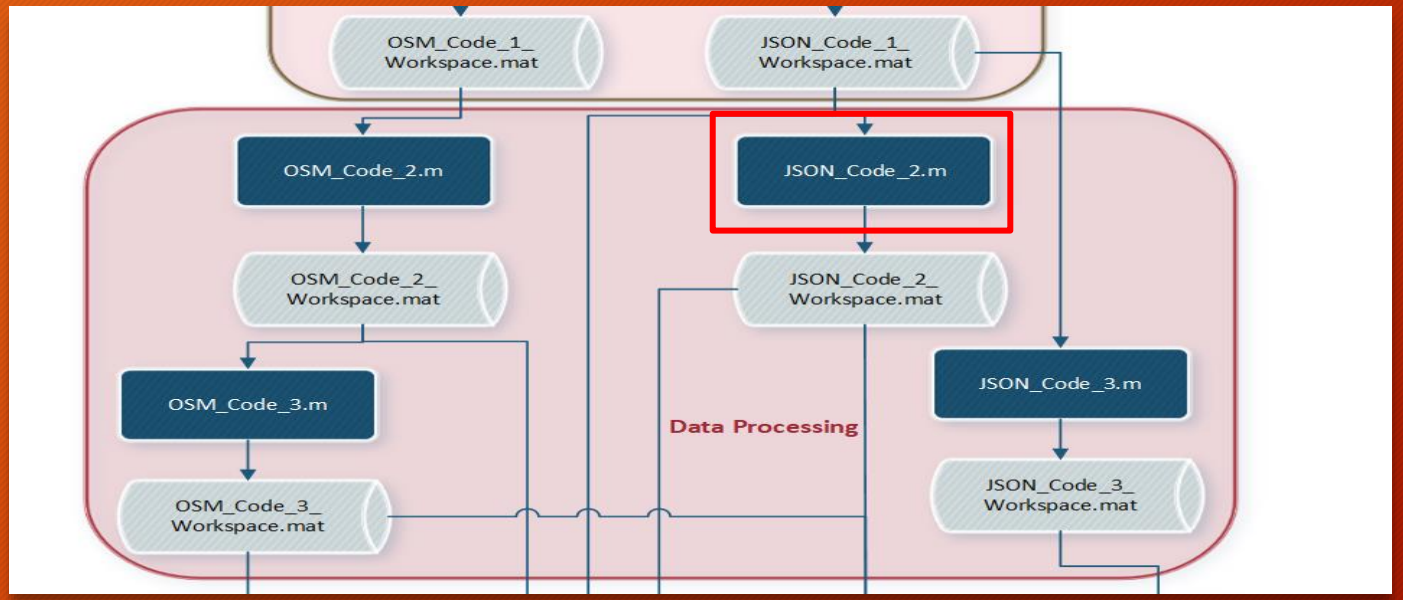


OSM_Code_3.m

```
%% Gathering All OSM Coordinates in 1 Matrix
%% load Data
%OSM Data Load
load OSM_Code_2_Workspace . WayData

%% All OSM Coordinates Matrix

All_OSM_Coordinates = []
for x = 1:length(WayData)

    All_OSM_Coordinates_0 = WayData(x).WayCoordinates;
    All_OSM_Coordinates = [All_OSM_Coordinates ; All_OSM_Coordinates_0];

end
```

Data Processing:
JSON_Code_2

- Input Variable: ‘dataFormatted’
- Operations: To convert timestamps from character to MATLABs’ datetime format
- Important for timely analysis

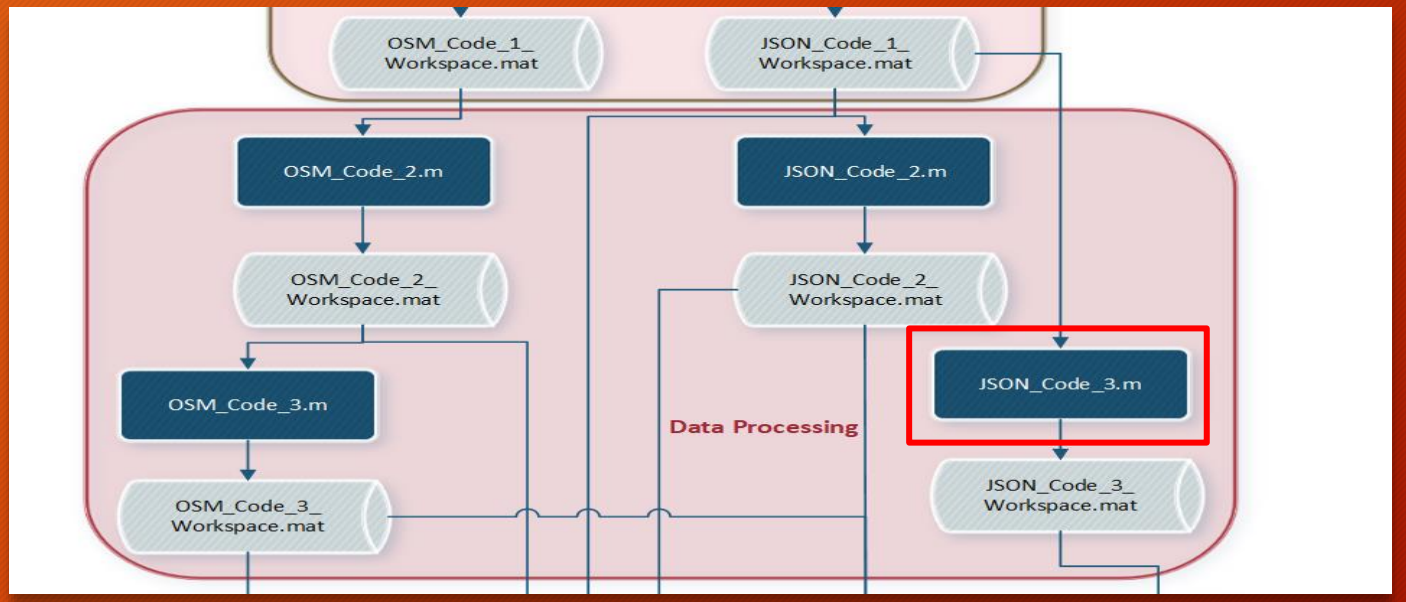


JSON_Code_2.m

```
%% Converting Date from ch to datetime Format
% For doing further Calculations

%% load Data

load JSON_Code_1_Workspace dataFormatted

%% Changing ch to datetime

for street = 1:length(dataFormatted)
    for field = 1:length(dataFormatted{street, 1})
        time = datetime(dataFormatted{street, 1}(field).date,'InputFormat
        dataFormatted{street, 1}(field).date = time;
    end
end
```

Data Processing:
JSON_Code_3

- Input Variable: ‘dataFormatted’
- Operation: Segregation of Jam direction on two-way streets
- Output Variable: ‘dataFormattedDir’



dataFormatted

616x1 cell

	1
1	1x10828 struct
2	1x4046 struct
3	1x3243 struct
4	1x28641 struct
5	1x5665 struct
6	1x26004 struct
7	1x26048 struct
8	1x15282 struct
9	1x10256 struct
10	1x1547 struct
11	1x883 struct

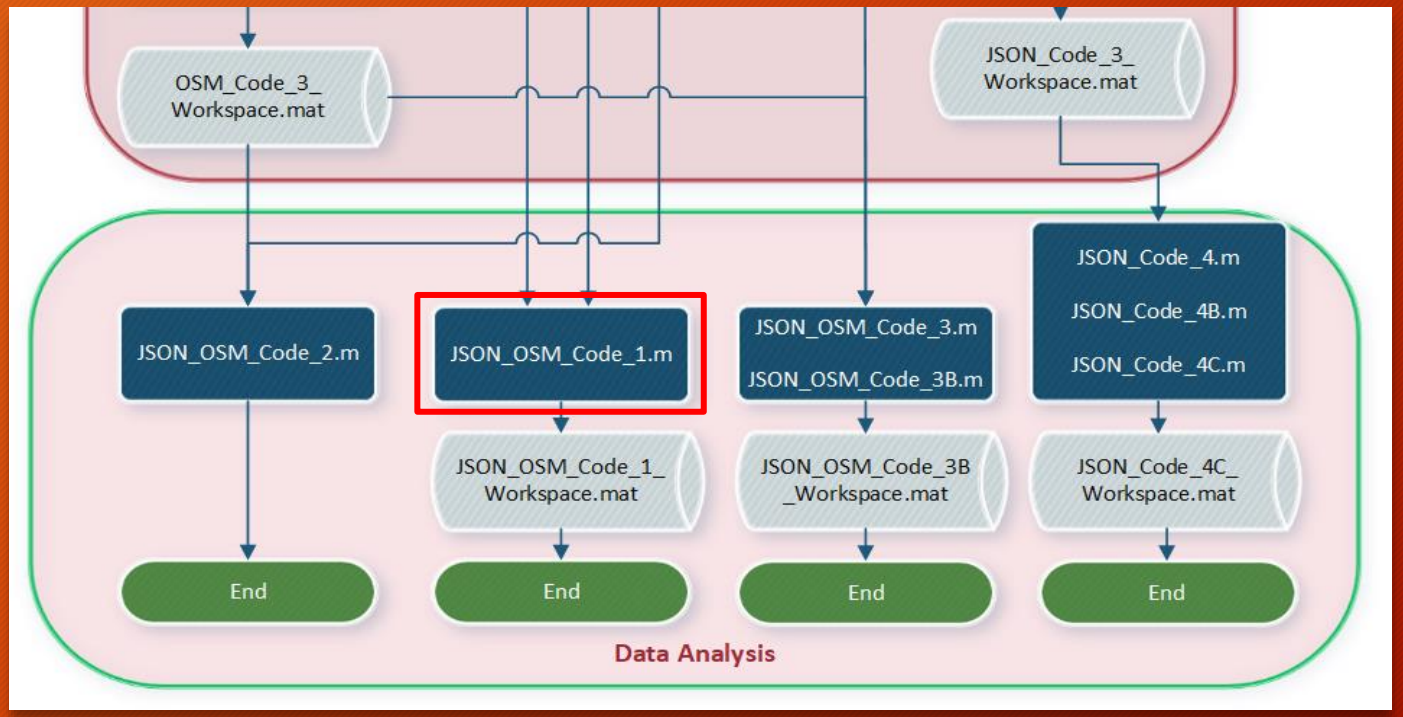
dataFormattedDir

616x2 cell

	1	2
1	1x4904 struct	1x5924 struct
2	1x290 struct	1x3756 struct
3	1x927 struct	1x2316 struct
4	1x14206 str...	1x14435 str...
5	1x1755 struct	1x3910 struct
6	1x3310 struct	1x22694 str...
7	1x25864 str...	1x184 struct
8	1x4150 struct	1x11132 str...
9	1x7192 struct	1x3064 struct
10	1x683 struct	1x864 struct
11	1x192 struct	1x691 struct

Data Analysis:
JSON_OSM_Code_1

- Input variables: WayData & dataFormatted
- Operation:
  - Inegrates 2 datasets
  - By assigning OSM Coordinates data to Trafmine data through **comparing street names.**
- Output: Comparison_Matrix
- Some drawbacks
  - Missing names
  - Spelling difference

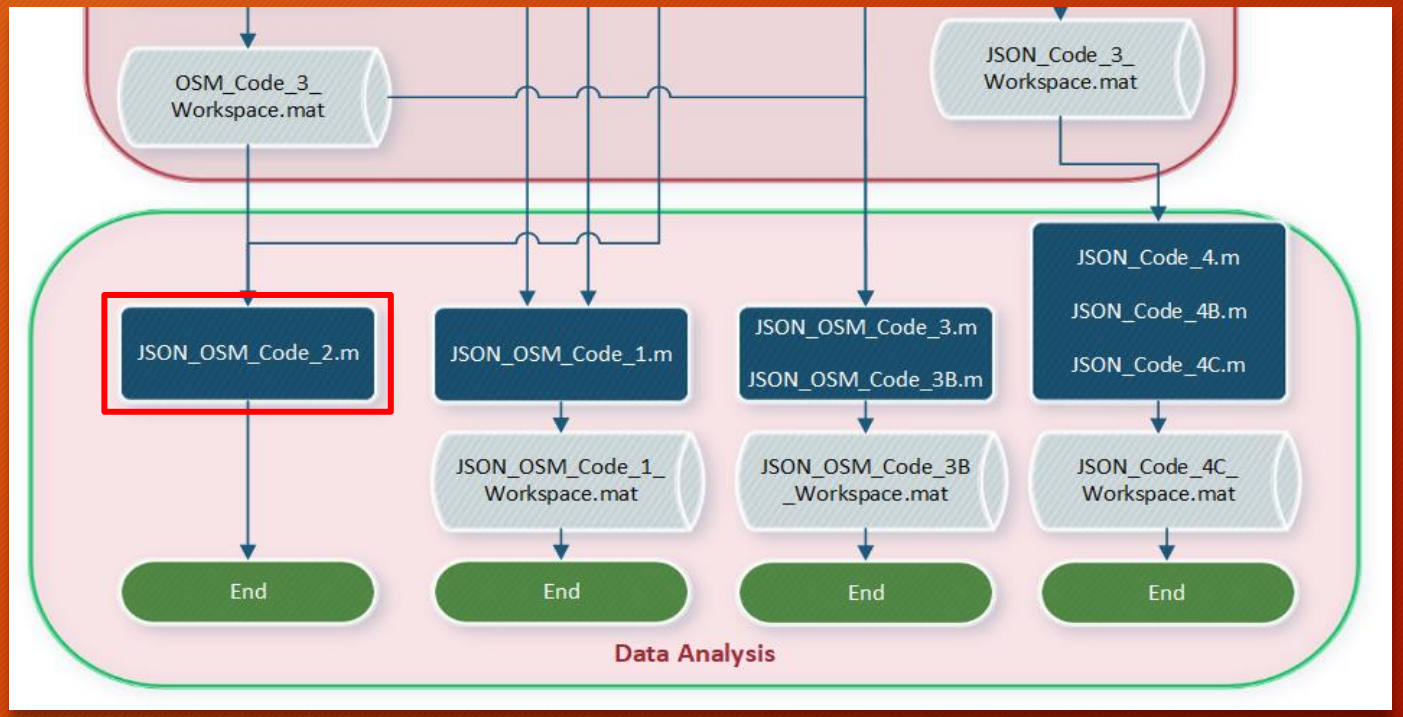


JSON_OSM_Code_1.m

```
10
11      %% Extracting all streets names from OSM data
12      %Street Names Full Matrix
13 -    StreetName_OSM = [];
14 -    for x = 1:length(WayData)
15 -        if isempty(WayData(x).StreetNames)
16 -            StreetName_OSM_0 = [x];
17 -        else
18 -            StreetName_OSM_0 = WayData(x).StreetNames;
19 -        end
20 -        StreetName_OSM = [StreetName_OSM ; StreetName_OSM_0];
21 -    end
22      %% Comparison Matrix for Both Data
23
24 -    Comparison_Matrix = [];
25 -    for StreetNo = 1:length(dataFormatted)
26
27          %Street Name Trafmine
28 -        StreetName_TM = dataFormatted{StreetNo, 1}(1).street;
29
30          %Street Data Trafmine
31 -        StreetData_TM = [];
32 -        for field = 1:length(dataFormatted{StreetNo, 1})
```

Data Analysis:
JSON_OSM_Code_2

- Input variables: All_OSM_Coordinates & dataFormatted
- Operations:
  - Integrates 2 datasets
  - By replacing Trafmine Coordinates with OSM Coordinates through **comparing closest coordinate set.**
- Large processing time
- High complexity

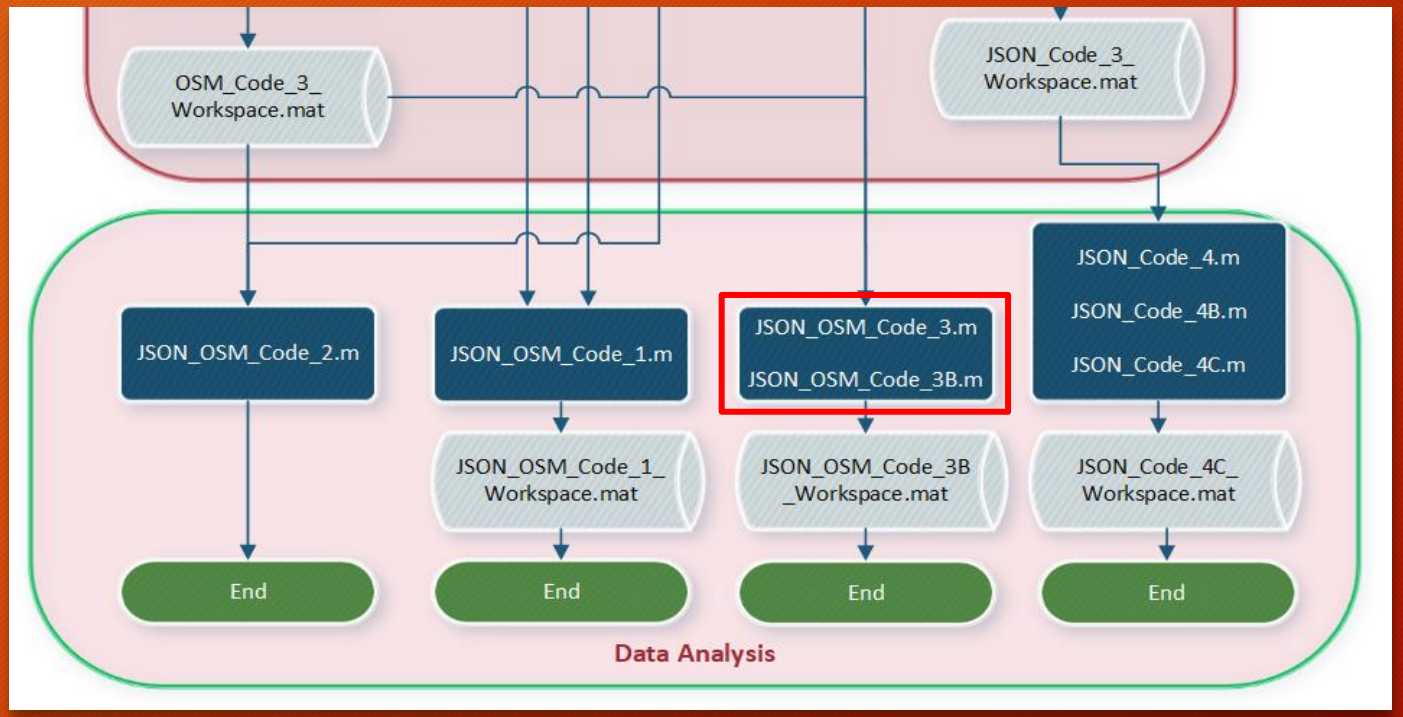


JSON_OSM_Code_2.m

```
%% Replacing Trafmine Coordinates with OSM Coordinates

for x = 1:length(dataFormatted)
    for x2 = 1:length(dataFormatted{x, 1})
        for x3 = 1:length(dataFormatted{x, 1}(x2).line)
            latlon1 = [dataFormatted{x, 1}(x2).line(x3,2) dataFormatted{x, 1}(x2).line(x3,1)];
            dist = [];
            for x4 = 1:length(All_OSM_Coordinates)
                latlon2 = [All_OSM_Coordinates(x4 , 2) All_OSM_Coordinates(x4 , 1)];
                [d1km, d2km] = distance(latlon1 , latlon2);
                dist0 = [d1km d2km];
                dist = [dist ; dist0];
            end
            [min_dist idx] = min(dist(:,1));
            log = [x , x2 , x3 , x4];
            new_lat = All_OSM_Coordinates(idx , 2);
            new_lon = All_OSM_Coordinates(idx , 1);

            dataFormatted{x, 1}(x2).line(x3,2) = new_lat;
            dataFormatted{x, 1}(x2).line(x3,1) = new_lon;
        end
    end
end
```

Data Analysis:
JSON_OSM_Code_3

- Input variables: All_OSM_Coordinates & dataFormatted
- Compact version of JSON_OSM_Code_2
- Operation: Replace coordinates of top-50 jam fields only

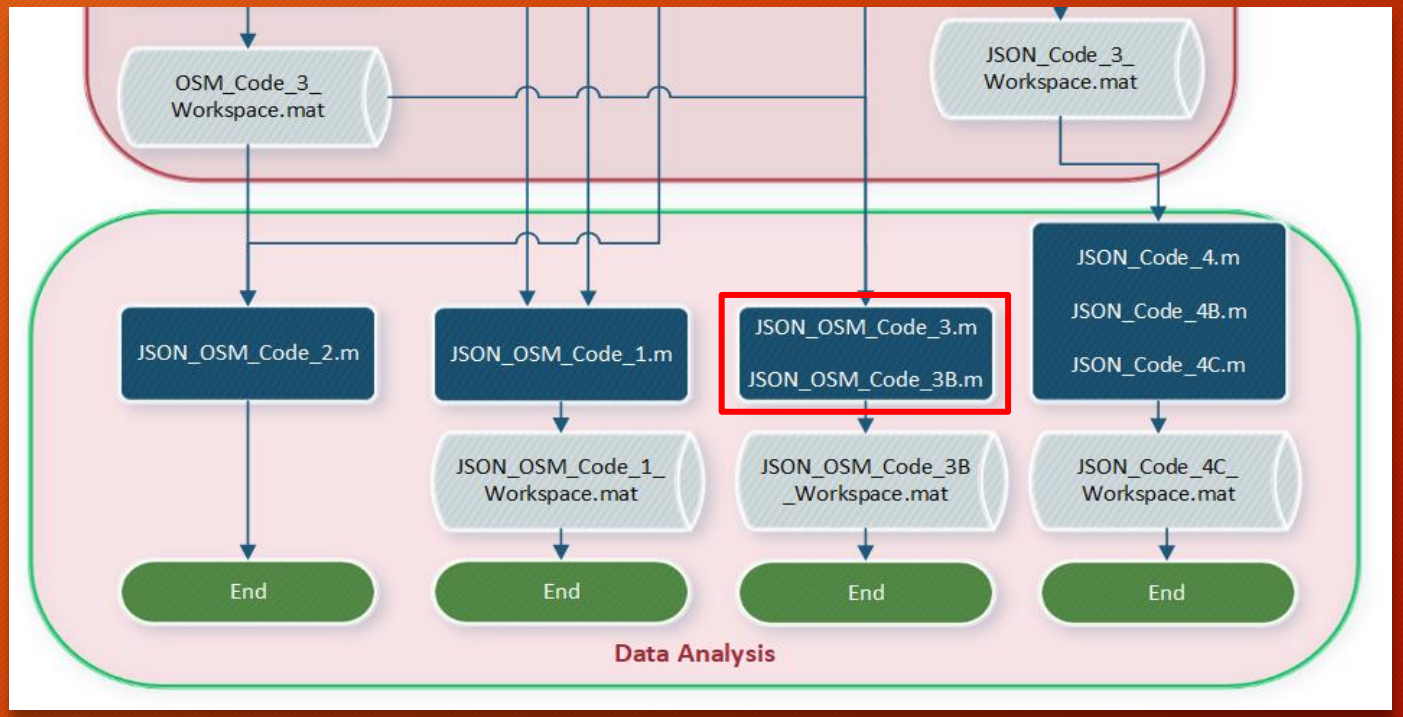


JSON_OSM_Code_3.m

```
%% Extracting top 50 delay fields from each street data

dataFormatted_50 = [];
for street = 1:length(dataFormatted)
    delay = [];
    for field = 1:length(dataFormatted{street, 1})
        delay0 = dataFormatted{street, 1}(field).delay;
        delay = [delay ; delay0];
    end
    [top_50_delay , idx] = maxk(delay,50);
    idx = sort(idx);
    top_50_fields = [];
    for field2 = 1:length(idx)
        top_50_fields0 = [dataFormatted{street, 1}(idx(field2))];
        top_50_fields = [top_50_fields ; top_50_fields0];
    end
    dataFormatted_50 = [dataFormatted_50 ; top_50_fields];
end
```

Data Analysis:
JSON_OSM_Code_3B

- Performs additional operation on JSON_OSM_Code_3
- Timely analysis
- Three additional input parameters:
  - Street Name
  - Start Time
  - End Time
- Output: delay, queue length and speed are plotted against time.
- Code can be modified to accept multiple street inputs

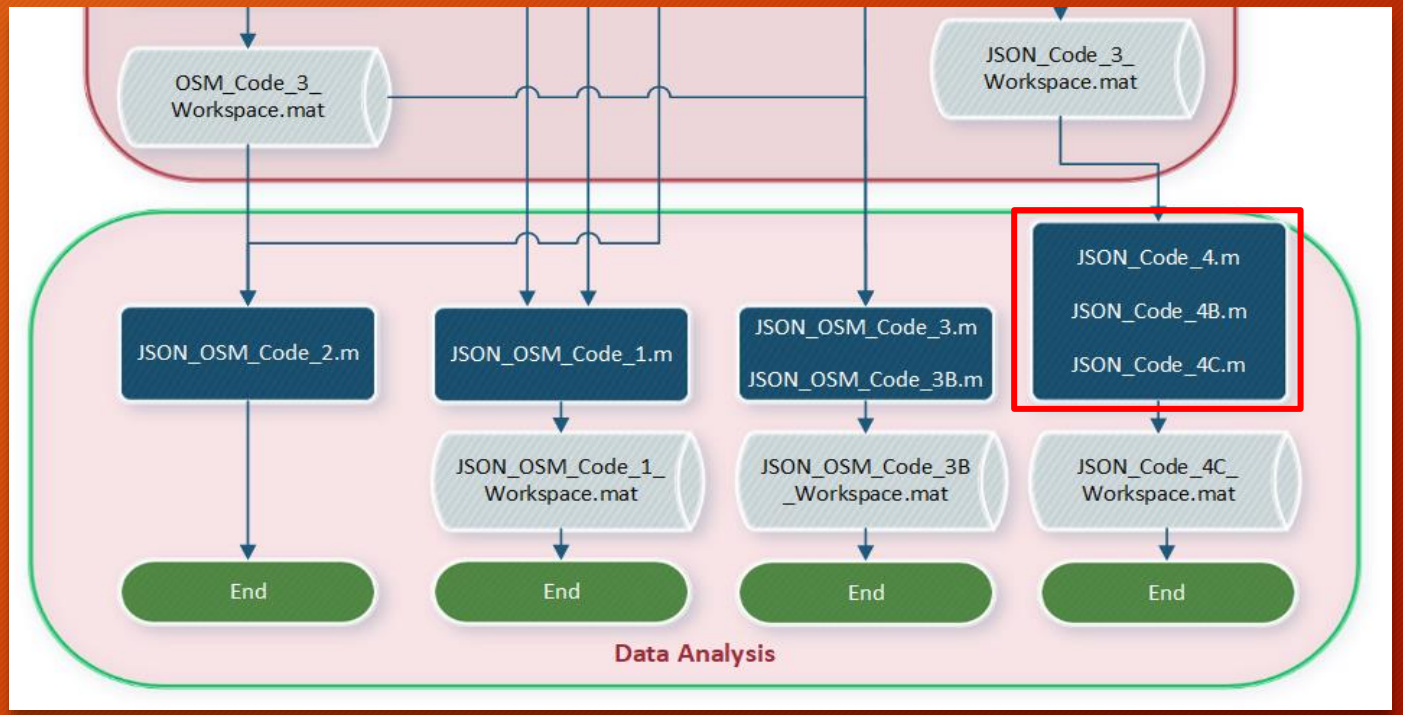


JSON_OSM_Code_3B.m

```
%% Plot Street Data with Time
% Top 50 Delays Vs Time

%Enter Street Name
street_name = "Egry József utca";
%Enter Start Date
start_time = datetime('2019-02-01T00:00:01+00:00','InputFormat','yyyy-MM-dd''T''HH:mm:ssXXX', 'Ti
%Enter End Date
end_time = datetime('2019-07-31T23:59:59+00:00','InputFormat','yyyy-MM-dd''T''HH:mm:ssXXX', 'Time

%% Data Collection
date = [];
delay = [];
q_length = [];
speed =[];
for field = 1:length(dataFormatted_50)
    if (dataFormatted_50(field).street == street_name & dataFormatted_50(field).date>=start_time
        % Timestamps
        date0 = dataFormatted_50(field).date;
        date = [date ; date0];
        % Delays
        delay0 = dataFormatted_50(field).delay;
        delay = [delay ; delay0];
        % Queue Lengths
```

Data Analysis:
JSON_Code_4

- Input variable: dataFormattedDir
- Operation: To convert timestamps from character to MATLABs' datetime format
- Enable timely analysis

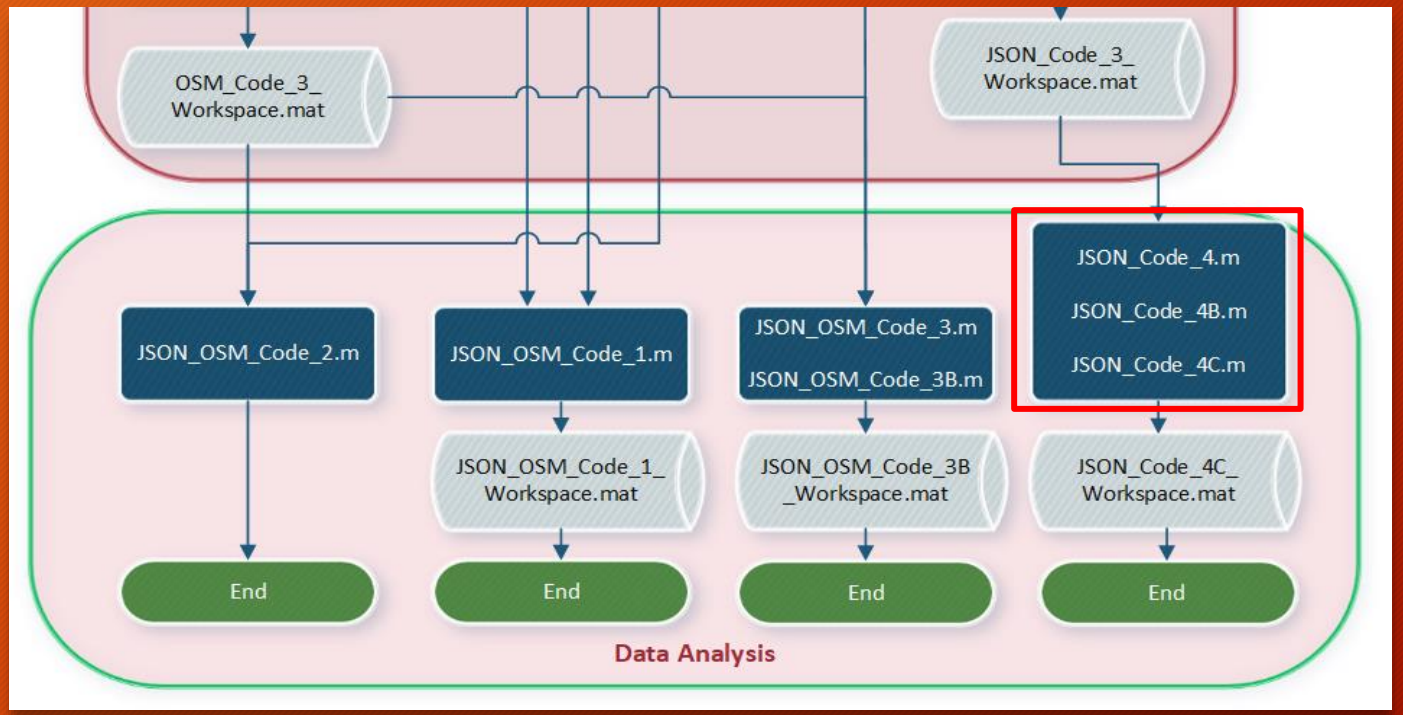


JSON_Code_4.m

```
%% Load Data
load DataForClustering.mat dataFormattedDir

%% Changing ch to datetime

for street = 1:length(dataFormattedDir)
    for dir = 1:2
     for field = 1:length(dataFormattedDir{street, dir})
         time = datetime(dataFormattedDir{street, dir}(field).date, 'InputFormat'
         dataFormattedDir{street, dir}(field).date = time;
     end
    end
end
```

Data Analysis:
JSON_Code_4B

- Further operation on JSON_Code_4
- Operation: to sort all the Jam entries in dataFormattedDir variable in ascending order with respect to time.



JSON_Code_4B.m

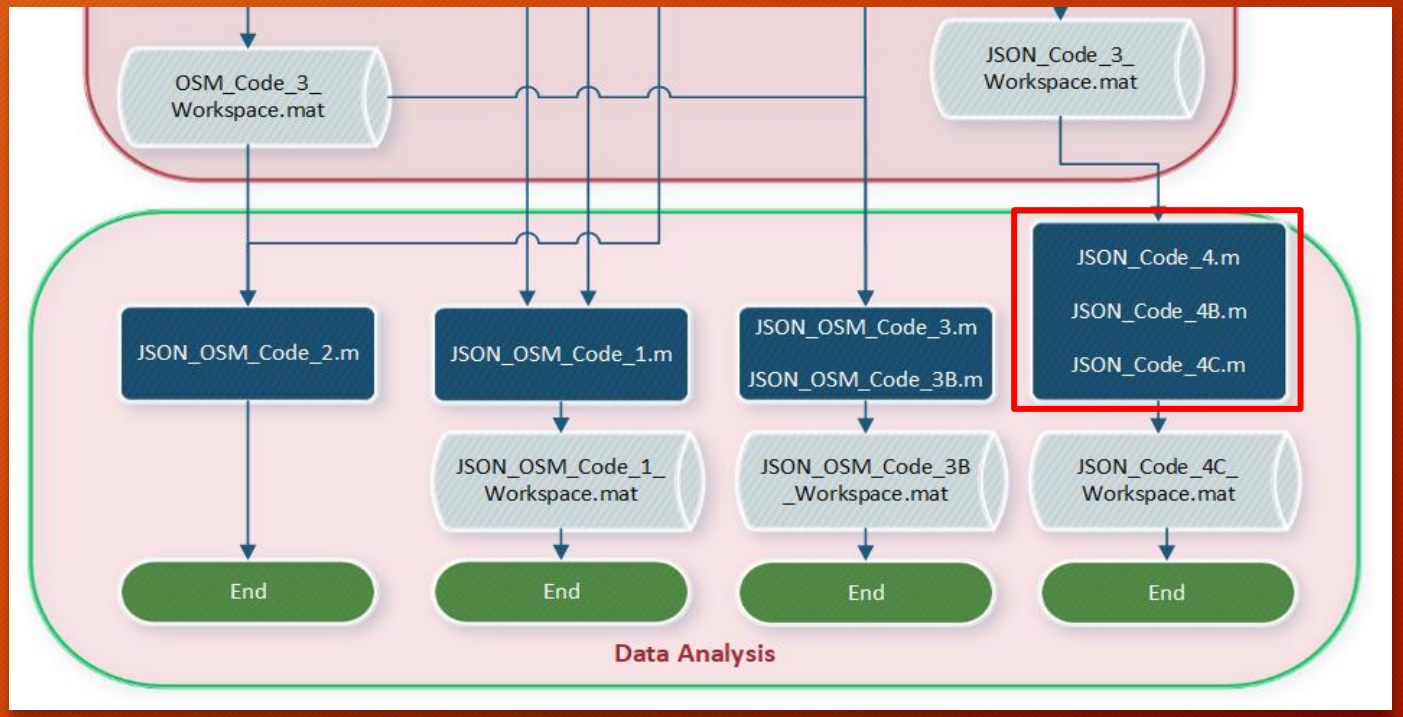
```
%% Gathering timestamps and addresses
%% Load Data
load JSON_Code_4_Workspace.mat dataFormattedDir

%% Getting timestamps
time = [];
for street = 1:length(dataFormattedDir)
    for dir = 1:2
        for field = 1:length(dataFormattedDir{street, dir})
            time0 = dataFormattedDir{street, dir}(field).date;
            time = [time ; time0];
            log = [street dir field];
        end
    end
end

%% Getting corresponding addresses
addr = [];
```

Data Analysis:
JSON_Code_4C

- Further operation on JSON_Code_4B

- Operations
  - Visualization of Jams on map
  - Creation of Statistics table

- How traffic evolved over time



JSON_Code_4C.m

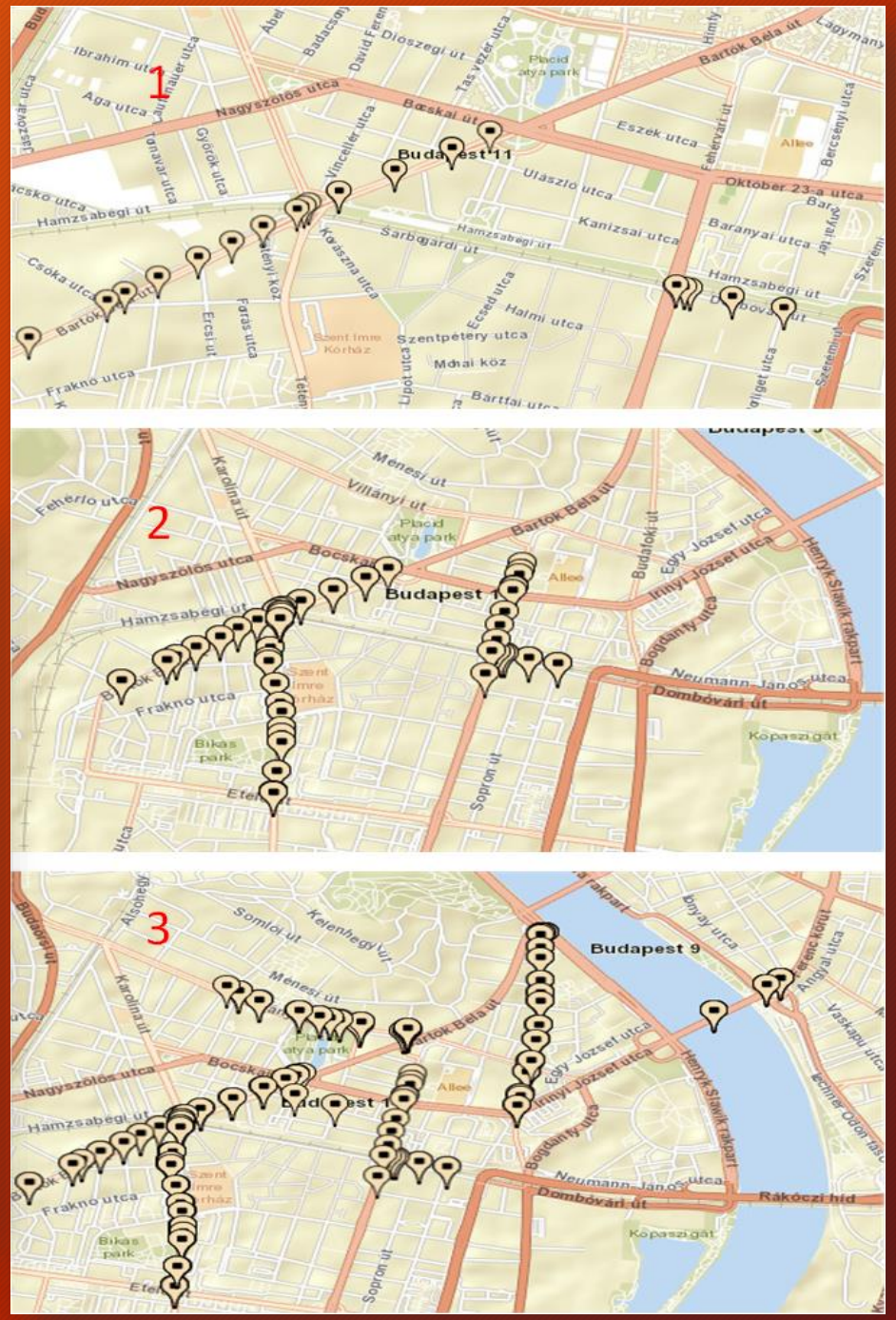
```
%% Visualizing the jams and obtaining statistics
%% Load data
load JSON_Code_4B_Workspace.mat

%% Plot
start_time = '01-Jan-2019 07:00:00';
end_time = '01-Jan-2019 10:00:00';
info = [];

for x = 1:length(sort_time)
    if (sort_time(x)>=start_time & sort_time(x)<=end_time)
        y = idx(x);
        address = [addr(y,1) , addr(y,2) , addr(y,3)];

        %Information
        line = dataFormattedDir{address(1), address(2)}(address(3)).line;
        name = string(dataFormattedDir{address(1), address(2)}(address(3)).street);
        direction = address(2);
        q_length = dataFormattedDir{address(1), address(2)}(address(3)).length;
        speed = dataFormattedDir{address(1), address(2)}(address(3)).speed;
        delay = dataFormattedDir{address(1), address(2)}(address(3)).delay;
```

Data Analysis:
JSON_Code_4C

Statistics

10x8 table

	1 Streert Name	2 Direction	3 Entries	4 Max Jam Length(m)	5 Avg Jam Length(m)	6 Min Speed(km/h)	7 Max Delay(s)	8 Avg Delay(s)
1	"Bartók Béla út"	"1"	"1"	"558"	"558"	"4.1889"	"83"	"83"
2	"Bartók Béla út"	"2"	"2"	"1253"	"1201.5"	"7.0306"	"78"	"74"
3	"Budafoki út"	"1"	"1"	"705"	"705"	"4.8472"	"61"	"61"
4	"Budafoki út"	"2"	"4"	"1088"	"1017.5"	"4.425"	"105"	"102.5"
5	"Dombóvári út"	"1"	"2"	"259"	"259"	"1.8778"	"109"	"108"
6	"Fehérvári út"	"1"	"1"	"650"	"650"	"4.6889"	"63"	"63"
7	"Petőfi híd"	"1"	"12"	"344"	"344"	"1.6639"	"186"	"138.3333"
8	"Tétényi út"	"1"	"6"	"1050"	"988"	"6.9472"	"75"	"70.5"
9	"Ulászló utca"	"1"	"1"	"177"	"177"	"0.87778"	"171"	"171"
10	"Villányi út"	"2"	"3"	"1025"	"987.6667"	"4.8417"	"79"	"71"



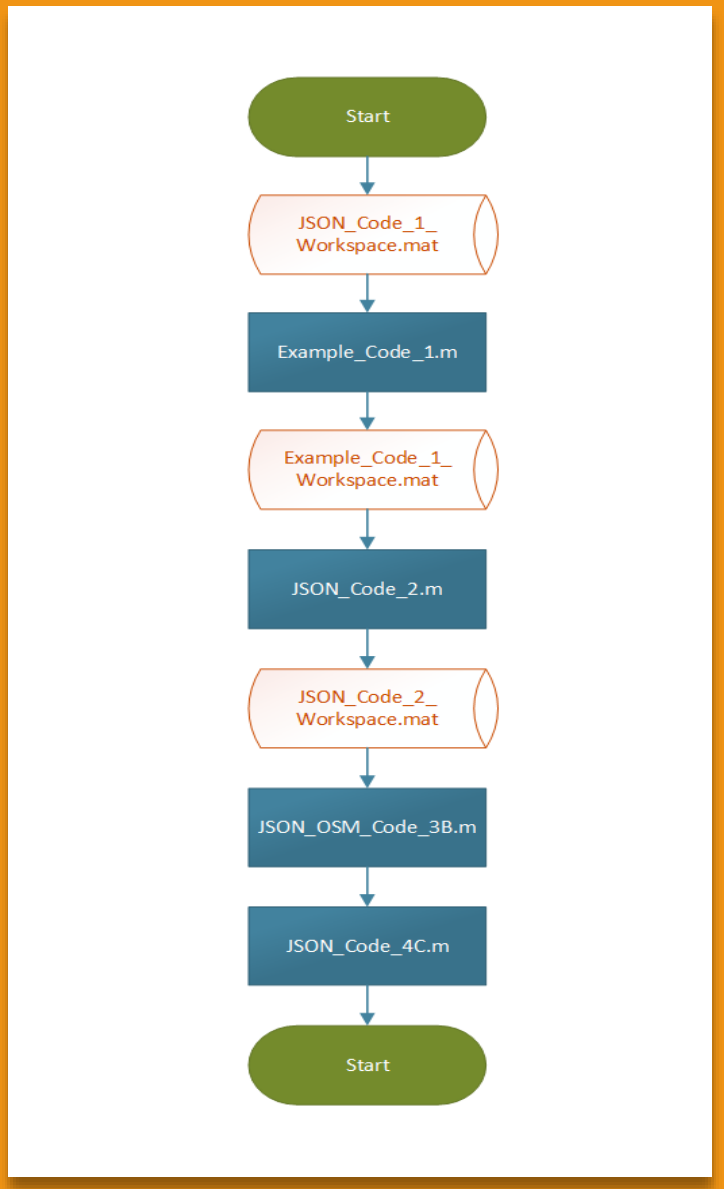
Application of Algorithm

Case:

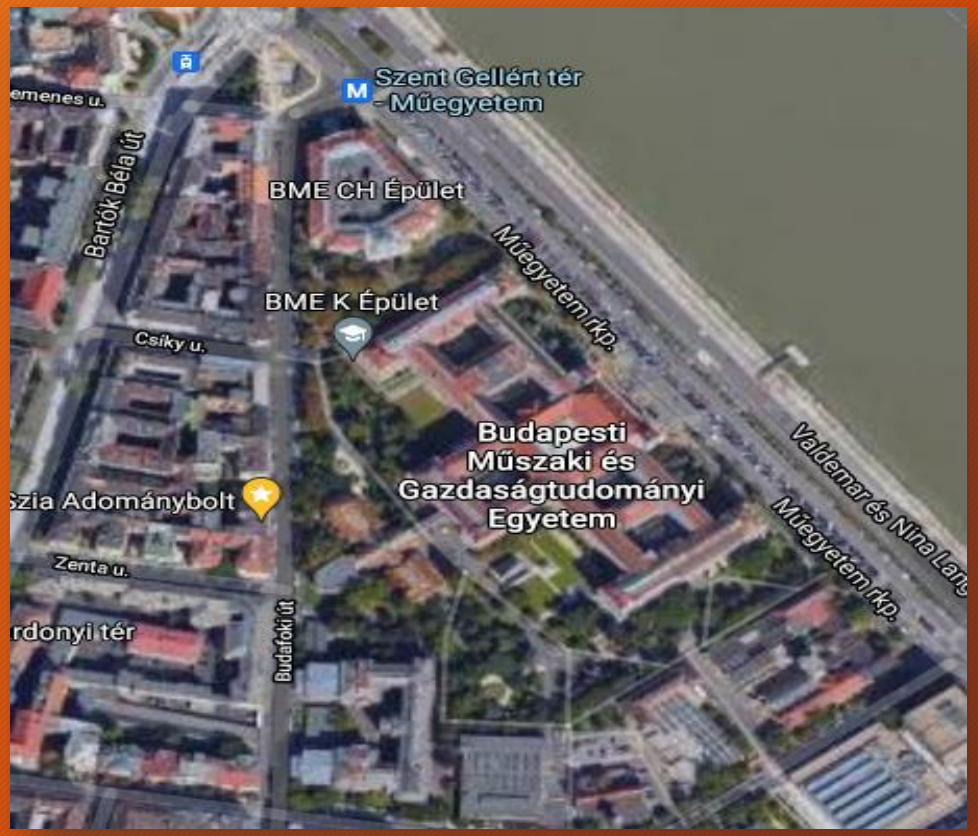
Performing the statistical analysis on the busiest street in 11th District Budapest in 2019.

Application of Algorithm

- To identify the busiest street the dataFormatted variable was processed through Example_Code_1.m
- busiest street which came out to be ‘Budafoki út’
- Although it is not one of the main roads of 11th District Budapest, but it runs along the BME
- JSON_Code_2 changes the character timestamps to MATLAB’s datetime
- JSON_OSM_Code_3B plots the timely data of a particular street and within a particular time period.



Application of Algorithm
Results



Example_Code_1.m

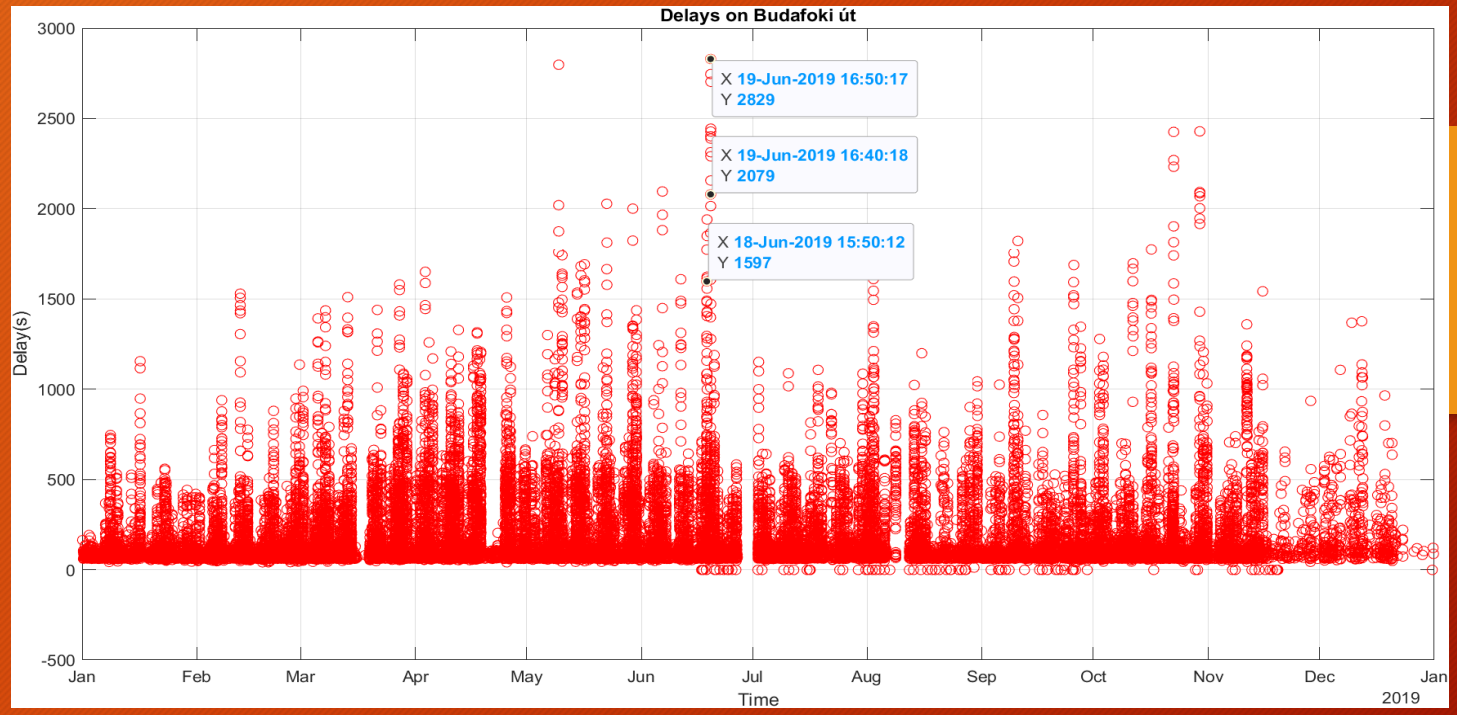
```
%% Load Data
load JSON_Code_1_Workspace dataFormatted

%% Identifying Busiest Street (with max numner of jam entries

entries = [];
for street = 1:length(dataFormatted)
      entries0 = length(dataFormatted{street, 1});
      entries = [entries ; entries0];
end

[max_jams, idx] = max(entries);
busiest_street = dataFormatted{idx, 1}(1).street;
%% Getting Data of Busiest street

Busiest_street_data = {dataFormatted{idx, 1}(:)};
```

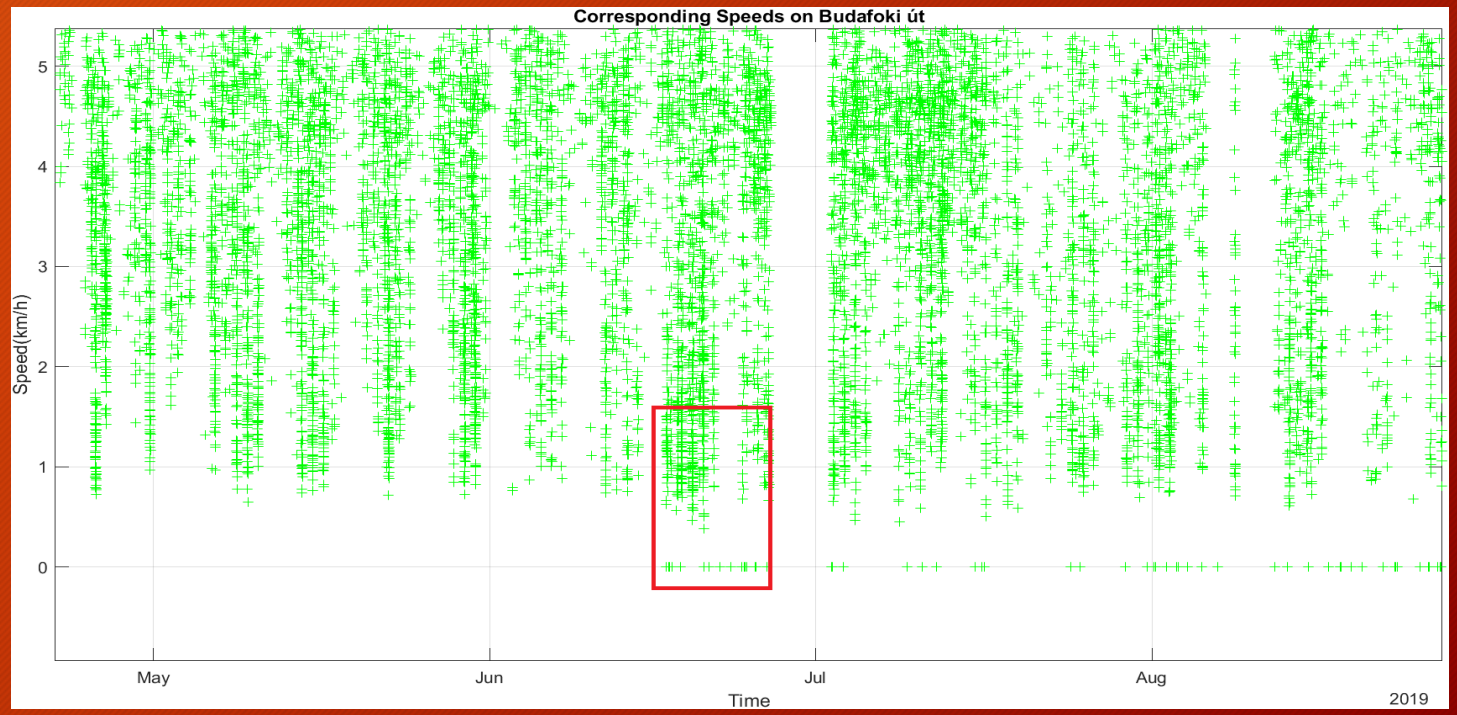
JSON_OSM_Code_3B.m

```
%% Further Analysis

%% Load Data
load JSON_Code_2_Workspace
dataFormatted_50 = Busiest_street_data;

%% Plot Street Data with Time

%Enter Street Name
street_name = busiest_street;
%Enter Start Date
start_time = datetime('2019-01-01T00:00:01+00:00','InputFormat','yyyy-MM-dd''
%Enter End Date
end_time = datetime('2019-12-31T23:59:59+00:00','InputFormat','yyyy-MM-dd''T'

```

Observations:

- Average delay length of the street lies between 500 to 700 seconds all over the year
- Almost every month has a spike in delay time around mid of the month
- Lengthiest delays can be seen on 19th of June between 15:00 hrs. to 17:00 hrs
- In contrast, a thick cluster of low speeds can also be observed around the same time in lower figure
- It can be perceived that there could be an event in the surrounding or in the BME campus which would cause a high pedestrian volume on the street and creating the long jams very frequently.

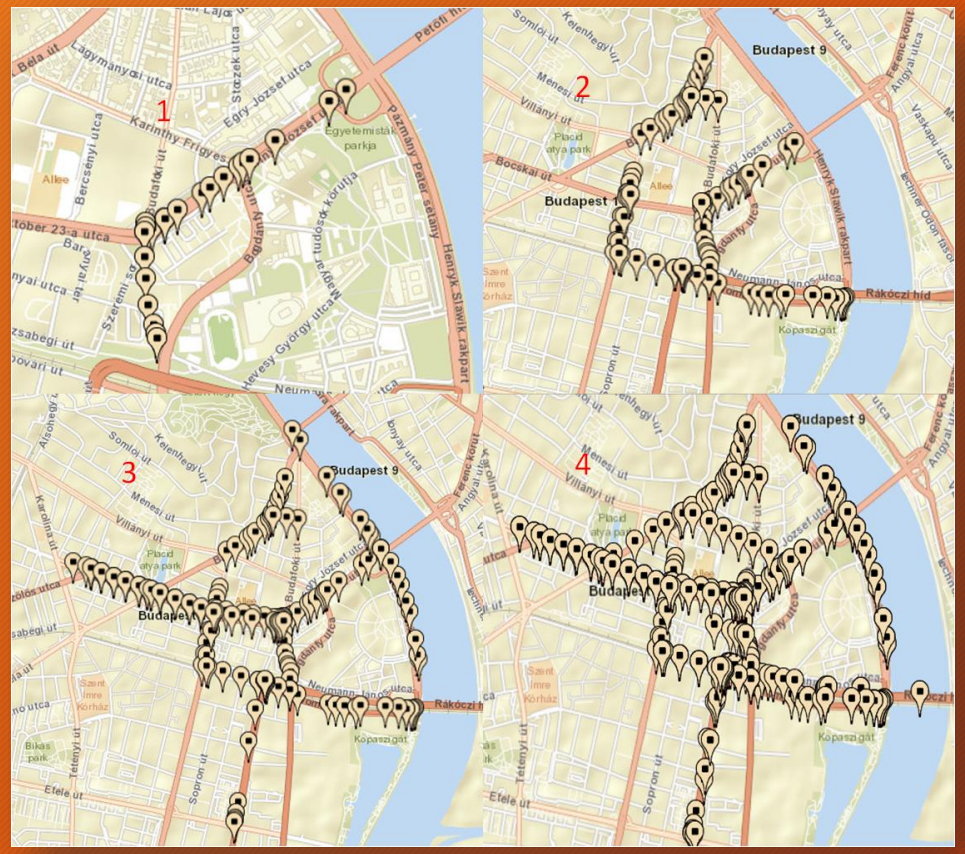




Application of Algorithm
Further Analysis

- With the help of JSON_Code_4C.m, this observation can be taken further to the visualization of jams over that critical time period.
- This code takes two time bounds and shows the projections of jams on the map and creates statistics table
- The start time and end time are set to 19th June 15:00 hrs. and 19th June 17:00 hrs.
- The results of the algorithm are shown in the figures below.

Application of Algorithm
Further Analysis



Statistics

55x8 table

	1 Streert Name	2 Direction	3 Entries	4 Max Jam Length(m)	5 Avg Jam Length(m)	6 Min Speed(km/h)	7 Max Delay(s)	8 Avg Delay(s)
1	""	"1"	"2"	"170"	"170"	"1.0861"	"129"	"121.5"
2	"Baranyai tér"	"1"	"15"	"215"	"184.2"	"0.28056"	"599"	"285.6"
3	"Baranyai tér"	"2"	"1"	"33"	"33"	"0.20833"	"154"	"154"
4	"Baranyai utca"	"2"	"24"	"518"	"249.1667"	"0.18889"	"1329"	"514.9583"
5	"Bartók Béla út"	"1"	"13"	"1129"	"533.4615"	"1.1444"	"464"	"220.4615"
6	"Bartók Béla út"	"2"	"30"	"1280"	"969.8667"	"1.5667"	"677"	"315.5667"
7	"Bercsényi utca"	"1"	"5"	"269"	"209.8"	"0.75278"	"152"	"98.8"
8	"Bertalan Lajos ...	"1"	"14"	"302"	"302"	"0.46667"	"598"	"311.3571"
9	"Bertalan Lajos ...	"2"	"7"	"365"	"242.1429"	"0.49167"	"605"	"303"
10	"Bocskai út"	"1"	"12"	"874"	"732.5"	"2.5083"	"137"	"98"
11	"Bocskai út"	"2"	"37"	"1109"	"797.5676"	"0.79444"	"1174"	"414.7838"
12	"Bogdánfy utca"	"1"	"14"	"641"	"641"	"1.0583"	"530"	"246.7143"
13	"Bogdánfy utca"	"2"	"6"	"573"	"468"	"1.6778"	"198"	"159.6667"
14	"Budafoki út"	"1"	"14"	"1191"	"572.5714"	"0.58333"	"1865"	"341"
15	"Budafoki út"	"2"	"51"	"2310"	"1033.1373"	"0.46667"	"2829"	"622.7843"
16	"Dombóvári út"	"1"	"9"	"409"	"373.4444"	"1.2944"	"288"	"162.4444"
17	"Dombóvári út"	"2"	"57"	"408"	"305.8596"	"0.46111"	"564"	"215.6842"
18	"Egry József utca"	"2"	"13"	"374"	"357.4615"	"0.58056"	"574"	"187"

Future Potential

- This algorithm is designed in such a way that it can be applied on Waze data from anywhere in the world.
- The integration of two datasets can open up a lot of opportunities as open-source maps can be integrated with the city's traffic conditions open for general public.
- Through the statistics obtained, the traffic demand can be forecasted in the case of special events. Which can be utilized to take preparatory steps such as opening alternate routes.
- The dynamic traffic network conditions can be integrated in the Open Street Maps, which are used by many web-based applications such as Amazon, Apple, Facebook, Baidu Maps etc.

Thank You for the Attention