# The effect of road network structure on speeding using GPS data

**A THESIS**
**SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL**
**OF THE UNIVERSITY OF MINNESOTA**
**BY**

**Toshihiro Yokoo**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**
**FOR THE DEGREE OF**
**MASTER OF SCIENCE**

**David Levinson**

**May, 2016**

# Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor David Levinson, for the continuous support of my Master study and related research and for his patience, motivation, and immense knowledge. His guidance helped me throughout the research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

Besides my advisor, I would like to thank the rest of my thesis committee: Professor Davis and Professor Mc Cullough, for their insightful comments and encouragement. I thank my colleagues for the help and support of research. I am also thankful for Jim Tilus and staff of the Center for Writing at the University of Minnesota. Without their editing assistance, I would not have finished this thesis.

I appreciate the financial support from West Nippon Expressway Company Ltd. (NEXCO-West) that funded my tuition to attend the University of Minnesota.

I am also grateful for my friends, especially my Brazilian friends, for happy times we spent together during my stay at the University of Minnesota. I deeply appreciate their friendship. Finally, I would like to thank my family: my parents and my sister for supporting me spiritually throughout writing this thesis and my life in general.

## Abstract

This paper analyzes the relationship between road network structure and speeding using GPS data collected from 152 individuals over a 7 day period. To investigate the relationship, we develop an algorithm and process to match the GPS data and GIS data accurately. Comparing actual travel speed from GPS data with posted speed limits, we measure where and when speeding occurs, and by whom. We posit that road network structure shapes the decision to speed. Speeding is large in both high speed limit zones (e.g. 60 mph (97 km/h)) and low speed limit zones (less than 25 mph (40 km/h)); in contrast, speeding is much lower in the 30 - 35 mph (48-56 km/h) zones. The results suggest driving patterns depend on the road type. We also find that if there are many intersections on the road, the average link speed (and speeding) drops. Long links are conducive to speeding.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Speeding behavior, driving faster than the speed limit, is affected by the road environment, the vehicle, and the driver. For example on interurban roads, young, educated, male drivers with cars are likely to exceed the speed limit [1, 2]. Considering this relationship, it is important to further understand specific influences on speeding behavior. This is especially true as studies show traffic crashes are often due to driver "manipulation error" (e.g., speeding and illegal overtaking) and "perception error" (e.g., wrong assessment of speed and distance) [1], and as speed increases, the percentage of both errors increases.

We posit that road network structures shape the decision to speed. Others have found that the percentage of speeding is large in both high speed limit zones (70 mph) and low speed limit zones (20, 30 mph); on the other hand, speeding is not remarkable in 60 mph zones [3]. This result suggests that drivers change their driving patterns depending on the road type. If there are many intersections on the road network, the average speed on the road linking those intersections would reduce due to external factors (e.g. traffic light, traffic condition, and approaching vehicle from other direction). In this thesis, we analyze the relationship between the road network structure variables and the percentage of speeding from the GPS data. To investigate the relationship, we develop an algorithm and process to match the GPS data and GIS data accurately.

- Chapter 2 presents a literature review
- Chapter 3 explains the material and methodology

- Chapter 4 describes results of analysis
- Chapter 5 discusses the analyses

# Chapter 2

# Literature review

## 2.1 Speed limit

Posted speed limits in Minnesota are regulated by statute [1] and the statute mentions that speed limit differs depending on the road type. (e.g., 30 mph on streets in urban districts, 65 mph on expressways, and 70 mph on rural interstate highways). Based on the statutes, the Minnesota Department of Transportation (MnDOT) establishes the posted speed limit on each road considering several factors (e.g., road type and condition, existing traffic control devices, and crash history) [4]. Moreover, MnDOT utilizes "the 85th percentile" as a guideline of speed limit, and therefore the posted speed limit would be close to the "the speed at or below which 85 percent of vehicles travel" [5]. This level is based on a study by David Solomon in 1964. He analyzed the relationship between accident and speed, and the results showed that the accident rate increases upward at the 85 to 90 percentile speed. This finding became common and went mainstream for traffic engineers [6]. However, there is doubt about the 85th percentile rule because this definition ignores the risk for both pedestrians and cyclists. Solomon's result was analyzed in highways, but now the 85th percentile is used not only in highways but also local residential roads. This is risky for both pedestrians and cyclists [7].

[1] 2014 Minnesota Statute: 169.14 SPEED LIMITS, ZONES; RADAR.

## 2.2 Reasons for speeding

Although speed limit is defined by engineering study and practice, some drivers don't follow the speed limit. It is widely understood that vehicle speed is influenced by three factors; drivers, vehicles and the road environment [8]. Greek data show that elderly, female and uneducated drivers who don't often drive on interurban roads are likely to follow the speed limit [1]. The traffic violator thinks the posted speed limit is not trustworthy, and they exceed the speed limit because "they are in a hurry", there is an "absence of police", and "speed limits are not reliable" [1]. A Survey of 207 drivers asked to answer the element among 14 factors that is related to the speed choice on the curve section found that "sight distance" has the most impact for the speed choice, followed by "pavement condition", "sharp curvature", and "additional warning signing", while "existence of free roadside space" and "speed limit signing" had little effect on speed choice to the driver [8]. Logically non-violators are more likely to comply with speed signage than violators.

Speed limit compliance in work zones is influenced by the vehicle gaps, traffic volume, time of day, leader vehicle, and vehicle size [2]. For example, light vehicles are likely to ignore the speed limit more than medium and heavy vehicles. When leading vehicle's speed is higher, the following vehicle also tends to exceed the speed limit. When the distance between vehicles is large, the driver is likely to exceed the posted speed limit. Also, high traffic volume tends to generate the non-compliance of speed limit. As the vehicle weight increases, the mean and variance of vehicle speed becomes small [9], clarifying that the level of speed limit compliance depends on surrounding traffic, road conditions, type of vehicle, and drivers.

## 2.3 Transportation network structure variables

Although some researchers analyze reasons for speeding regarding drivers, traffic condition, and vehicles, it is assumed that the road network structure also has an effect on speeding. Considering the relationship between transportation network structure variables (e.g., connectivity, hierarchy, circuity, treeness, entropy, and accessibility) and transportation performance (e.g., journey-to-work time and automobile mode share)

across 50 U. S. cities, it was found that large network connectivity decreases the commute time, large accessibility reduces auto mode share, and large city size increases the delay [10].

This analysis might be applied to speed limit compliance. For example, it is posited that a small number of intersections in the city (long link length) would be correlated to speeding. In order to analyze the relationship between network structure variables and speeding, we need to collect enormous amount of traffic data over a broad area. In general, a traffic detector is used to analyze the traffic data. However, the amount of data on local roads is small. Therefore, in our thesis we use GPS data because GPS data contains several information; time, position (latitude, longitude, altitude), distance, speed, the number of satellites [11]. If GPS data covers the whole network, it is more effective to analyze the relationship between network structures and speeding than loop detectors on selected facilities.

## 2.4 Use of GPS data in safety analysis

GPS data contains strengths and weaknesses. In terms of strengths, this data is likely to reflect a daily driving pattern. Compared to the driving experiment on a test route, participants are likely to drive on routes they prefer. Drivers live in Minnesota, so they would generally be familiar with their driving route. Moreover, compared with loop detector data, we can evaluate time series data of each driver and analyze extensive coverage not only for trunk roads but also residential roads [12]. GPS data is more practical and trustworthy for collecting traffic data than inductive loop detectors because inductive loop detectors are expensive to install and maintain and they are prone to failure [12]. However, GPS data is limited because route and date are differ by driver. Each driver drove under different conditions. It is assumed that some driving patterns are affected by weather, congestion and road type, and it is also difficult to define external factors of driving (e.g. a traffic light and a sudden deceleration due to the preceding car). Furthermore, GPS requires map matching [13].

## 2.5 Map matching

GPS data contains speed information while GIS maps possess the posted speed limit data, therefore when we match the GPS data with GIS maps, we can analyze the relationships between driving speed data and road networks. However, there are some problems about this process regarding the accuracy of GPS data and calculation time.

GPS devices provide highly accurate, but not perfect data because of the effect of atmospheric conditions, neighborhood structure, and selective availability [11]. That research shows that 86.5% of GPS data was accurate within 1.5m and 99.89% accurate within 2.0m. The accuracy of speed data depends on the device and road alignment (straight or curve). For example, non-differential GPS device [11] and the Wide-Angle Augmentation System (WAAS) [14] were used to analyze the accuracy of GPS data. In a straight line, 97.9% of non-differential GPS device speed data was within 0.2 m/s; on the other hand, 67% of WAAS GPS speed data was within 0.2 m/s. On a curve area, 86.7% of non-differential GPS device speed data was within 0.2 m/s. On the contrary, 65% of WAAS GPS speed data was within 0.2 m/s. This result showed that the accuracy differs by GPS device, and straight line data indicates higher accuracy than curve area. Accuracy of speed data depends on the method of calculating speed; "Doppler shift" and "differences in GPS position over time". The result in straight course showed that 90.8% of "Doppler shift" data was accurate within 0.1 m/s and 66.5% of "differences in GPS position over time" data was accurate within 0.1 m/s [11].

This inaccuracy of both GIS and GPS data lead to the difficulty of finding the actual route when we match GPS data and GIS map; however, if we match every GPS data manually, it would take a huge amount of time. Therefore, it is desirable to find a proper map-matching method and create an algorithm for analyzing the GPS data. In the ST-matching algorithm, each GPS point chooses the candidate of actual road network point and calculates the probability [15]. The probability is decided by a zero-mean normal distribution with a standard deviation of 20 meters. While the algorithm chooses the highest probability point for the map-matching, it also considers the following two key factors to increase the accuracy of map-matching:

- "True paths tend to be direct, rather than roundabout" [15] – Because of the measurement errors, some GPS points are likely to be located at different roads

that are close to the actual road. Therefore, if previous and future GPS data are located in the same road, one point that is located along another road will be modified to the same road.

- "True paths tend to follow the speed constraints of the road" [15] – In some areas, highways and local roads are close to each other, and therefore it is difficult to define the actual road network link from the nearest distance between the GPS data and GIS map. The author considers the average vehicle speed, and if the average speed is high enough, the driver is likely to use highways more than local roads.

Then, the author tests the ST-Matching in Beijing road network and this network contains 58,624 nodes and 130,714 links. The results mention that the accuracy and calculation time is related to the number of candidate points that algorithm chooses. As the number of candidates increases, the accuracy of matching data increases; on the other hand, the running time increases. For example, when the number of candidates is one point, the accuracy is about 80%. On the contrary, when the number of candidates is three points, the accuracy is approximately 95%.

One of the reasons for the map-matching failure is a missing link in the map [16]. Comparison of the accuracy of map-matching between high-resolution and low-resolution networks show that low-resolution networks cannot find actual trip links, and therefore, detailed GIS network maps are important for the map-matching and it is desirable to use the most accurate GIS map that covers majority of routes in the area [17] .

# Chapter 3

# Data and Processing

## 3.1 Data

This paper uses GPS data from the 2010 Twin Cities (Minneapolis - St. Paul area) Travel Behavior Inventory (TBI2010) administered by the Metropolitan Council between 2010 and 2012 (most data was collected in 2011) [18]. Each subject in the GPS component of the TBI carried a GPS pendant for 7 days. The raw GPS data contains the following trip information: Speed (km/h), Longitude, Latitude, Altitude (meters), Date (year/month/day), Time (hour/minute/second), Distance (meters), Course (degree), Number of satellites, HDOP. An example of GPS data is Table 3.1. Seven days of movement for each person is recorded in the data, and the trajectories were recorded every second as Table 3.1 shows.

In order to check the speed limit compliance, we will need to identify the road that participants drove, and then compare the driving speed with the posted speed limit. Therefore, we mainly use location data (Longitude and Latitude) and speed data (Speed (km/h)) in our analysis.

However, this GPS data by itself does not contain personal information (e.g. gender, age). Therefore, we use the associated records from the TBI2010 Household Interview Survey [19] as complementary data (e.g., Household data, Personal information, Trip data). These data can be matched on TripID. Among 274 GPS subjects (drawn from 250 households), 152 travelers have trip IDs matching the survey and no other data problems, allowing us to analyze 152 participants. (Although the total with trip ID

Table 3.1: Example of GPS data

| Longitude | Latitude | Speed (kilometer) | Course (degrees) | Number Of Satellites |
|---|---|---|---|---|
| -86.283816 | 39.712155 | 20 | 46 | 4 |
| -86.28378 | 39.712171 | 19 | 48 | 4 |
| -86.283718 | 39.712211 | 19 | 48 | 4 |
| -86.28367 | 39.712255 | 19 | 48 | 4 |
| -86.283631 | 39.712276 | 17 | 46 | 4 |

| HDOP | Altitude (meters) | DD/MM/YY | HH:MM:SS | Distance (meters) |
|---|---|---|---|---|
| 6.29 | 223 | 23/6/2011 | 15:45:34 | 0 |
| 6.3 | 223 | 23/6/2011 | 15:45:35 | 3 |
| 5.4 | 223 | 23/6/2011 | 15:45:36 | 6 |
| 6.29 | 223 | 23/6/2011 | 15:45:37 | 6 |
| 6.29 | 223 | 23/6/2011 | 15:45:38 | 4 |

is 160, the age of 8 travelers is under 16.) We append the GPS data with personal information describing each subject's gender, age, and education.

With the purpose of understanding actual speed limit data, we use a GIS map maintained by the Metropolitan Council and The Lawrence Group (TLG) that covers the majority of routes in the Twin Cities seven county metropolitan area. This GIS street map has the most accuracy at the present time. The map contains 290,231 links, and each link has several attributes (e.g. speed limit, length of link, street name, one-way).

To conduct the analysis, we use QGIS [20], an open source geographic information system.

## 3.2 Data processing and map-matching

In order to match GPS data and GIS data, we use 278 GPS subjects that are recorded in a CSV file; however, each file contains an enormous amount of data (total size is 1.89GB). Therefore we need to create an algorithm to improve the efficiency and speed of calculation and data processing while keeping the desirable accuracy.

Before the map matching, we need to extract the GPS data where the travel mode is car and it has personal information. The process of extracting appropriate driving

data is shown in Figure 3.1 that has 6 steps from "1: Trip generation" to "6: Matching of trip ID".

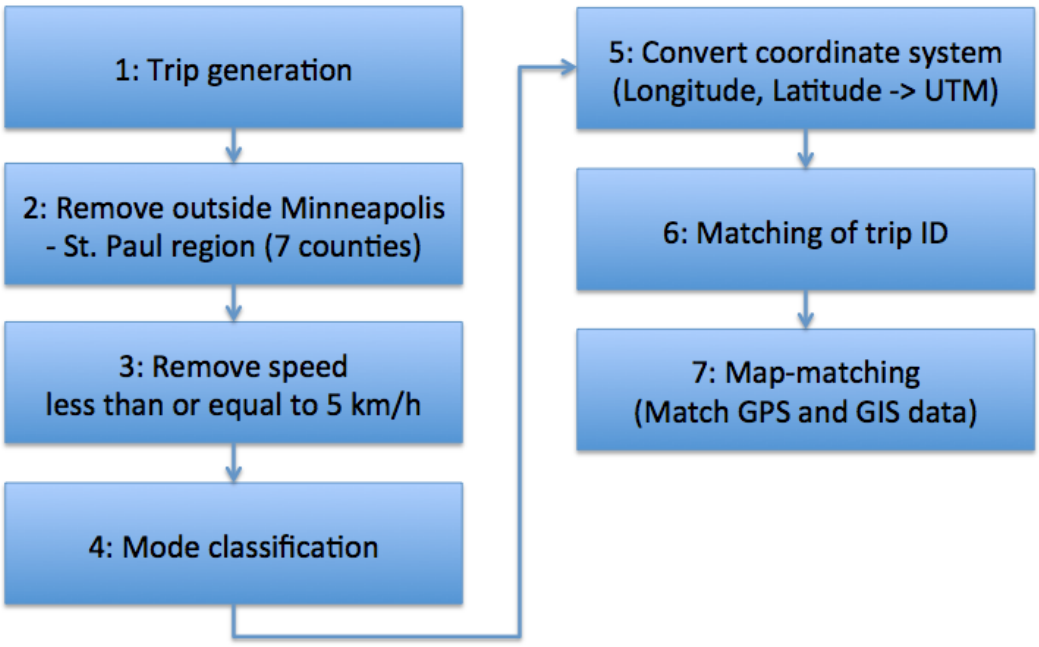



Figure 3.1: Flow chart of data processing and map-matching

Figure 3.2 is an example of initial GPS data. First, we divide the initial GPS data into each trip data (From Trip 1 to Trip 6) because time of initial GPS data is disconnected. Then, based on speed and location data we assign a travel mode from five types: Walk, Train, Bus, Bicycle, and Car. After mode identification, we extract driving trips (Trips 1 and 6) and match GPS data and GIS data. The algorithm is implemented in the Python language because QGIS supports Python.

In terms of the algorithm, we use an algorithm and methodology developed previously [21] as a reference. Some algorithms ( "1:Trip generation" A.4.1, "3:Remove speed less than or equal to 5 km/h" A.4.3 & A.4.5, "4:Mode classification" A.4.4) are reused [21], but converted to execute on Mac rather than Windows. We explain the details of each step in the next section.

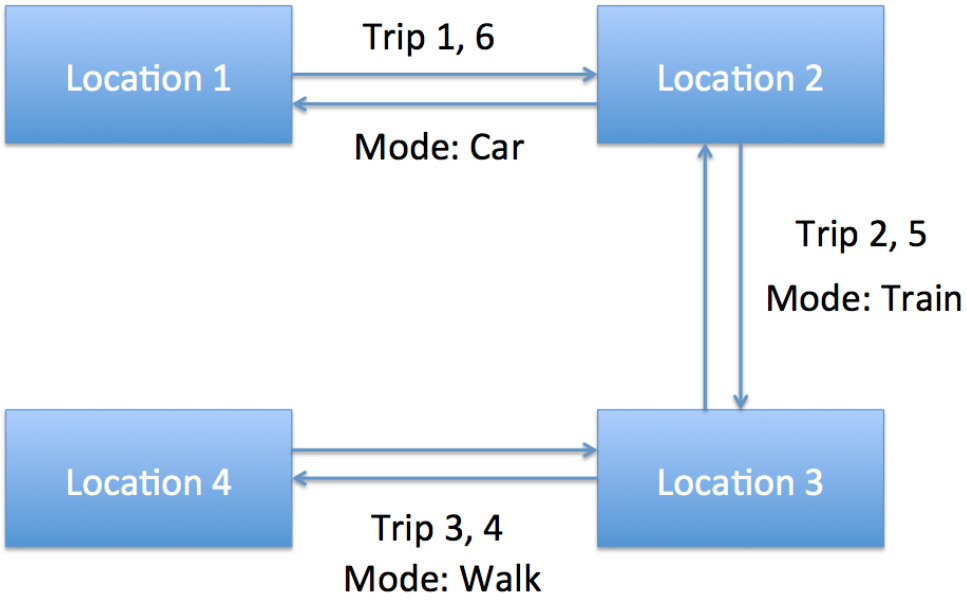



Figure 3.2: Image of initial GPS data

### 3.2.1 Trip generation

First, we extract the driving trips from GPS data of TBI2010. Each GPS data contains 7 days travel data and as Figure 3.3 shows, GPS travel records initially contains data from multiple trips. Since the initial GPS data doesn't record each trip and the time that the GPS data is disconnected, we define the trip based on the time difference. The algorithm divides the initial GPS data into the trip data when a time difference of the GPS data is more than 300 seconds. The code of this algorithm is mentioned in A.4.1.

As a result, the initial 278 GPS subjects made 16,902 trips. It means that each GPS subject made about 60 trip data on average. Figure 3.4 illustrates the GPS data after being decomposed into individual trips.

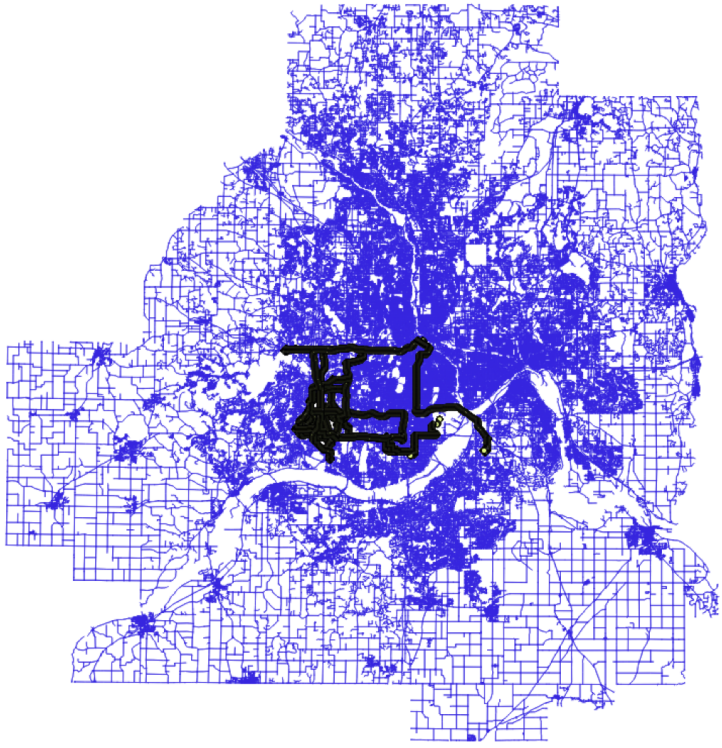


Figure 3.3: Example of initial GPS data (7 days)

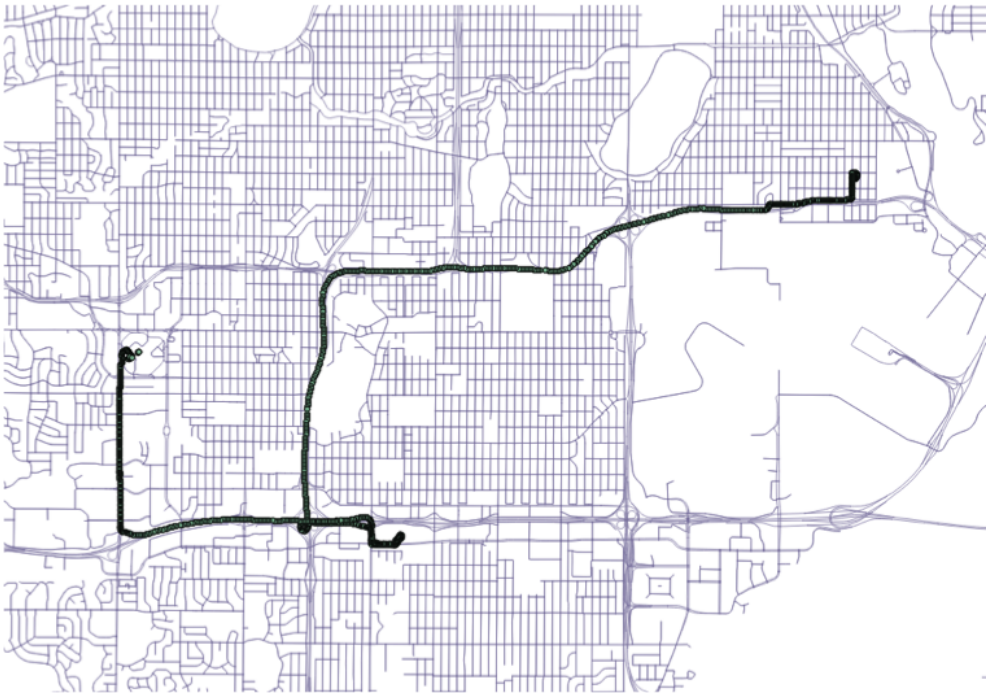


Figure 3.4: Example of one trip data

### 3.2.2 Remove outside Minneapolis - St. Paul region (7 counties)

Although participants live in the Twin Cities metro area, some people traveled outside of the Minneapolis area. The TLG map, known as the GIS map, covers the seven county metropolitan area in Minnesota; however, some GPS data are not located on this map. To remove the GPS data that are plotted outside Minneapolis - St.Paul region (7 counties), we use the following boundary to define the study area:

$$-94.0123 <= Longitude <= -92.7397$$

$$44.4714 <= Latitude <= 45.4139$$

Data outside these boundary coordinates are removed. Based on this algorithm, some trip data are modified and 791 trip data are removed. Then, the total number of trip decreases to 16,111 trip data. An example of the result is in Figure 3.5. Figure 3.5 shows GPS data that are plotted outside of the map are removed. The code of this algorithm is in A.4.2.

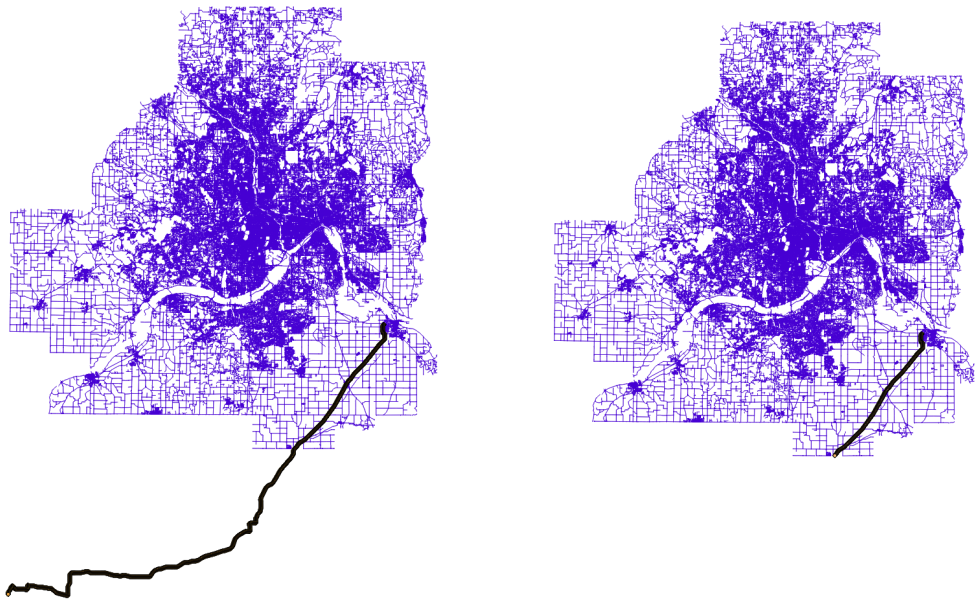


Figure 3.5: Remove outside Minneapolis - St. Paul region (7 counties) (Left: before, Right: after)

### 3.2.3 Remove speed that is less than or equal to 5 km/h

Before the mode classification, we remove the speed data that is less than 1 km/h.

As Figure 3.6 shows, enormous GPS data are gathered at one place and the speed of this area is almost 0 km/h. It is considered that participants stop moving because of shopping or preparation to go out. These data are included in the trip data especially at the beginning or ending. When these speed data (0 km/h) account for the large percentage of the trip data, they affect the result of the mode classification because the average speed decreases.

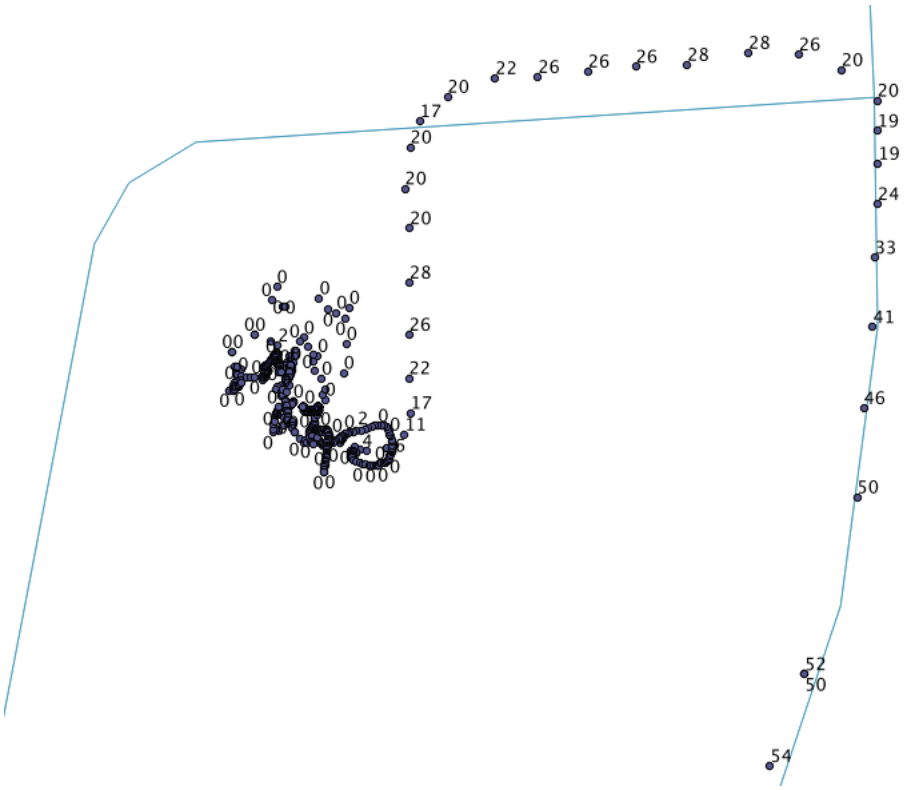



Figure 3.6: Image of 0 km/h speed data (Number means speed)

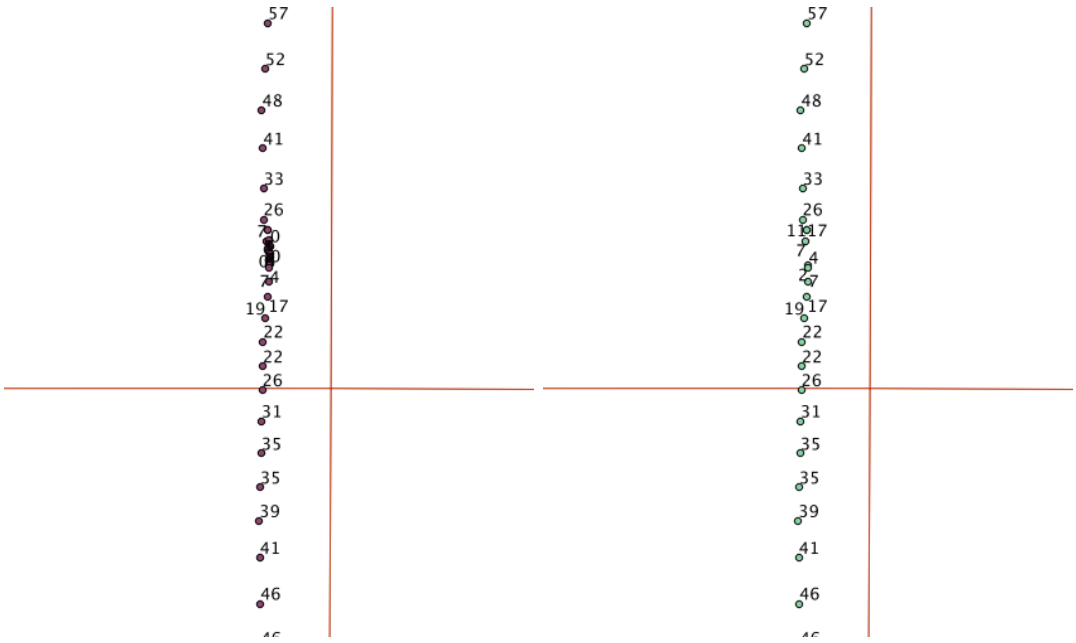



Figure 3.7: Remove speed that is less than 1 km/h (Left: before, Right: after)

An example of the result is in Figure 3.7. As Figure 3.7 shows, some GPS points are concentrated at the center area of the map and these speeds are 0 km/h. It seems that the vehicle was stopped at an intersection due to the traffic light. After the algorithm, these points are removed while other speed data remain. The code of this algorithm is mentioned in A.4.3. Due to this algorithm, 3,543 trips are removed and the total number of trips decreases to 12,568.

After the mode classification, we remove speed data that is less than or equal to 5 km/h from the driving trip data in order to remove the effect of traffic lights and traffic jams.

We want a driving condition that is not affected by external factors for analyzing speed limit compliance. If we include the effect of traffic lights and traffic jams, the result of speed limit compliance would be overestimated. The code of this algorithm is mentioned in A.4.5. In this algorithm, no trip data changes while some data are deleted.

### 3.2.4 Mode classification

GPS data contains not only driving data but also walking, biking, and transit data (typically lower speed data). The purpose of our research is to focus on driving speeding behavior, so the algorithm extracts the driving data. The definition of mode classification is from Wenyun [21]. The condition to classify the data is as follows; (When the data doesn't apply to any of travel mode conditions, it becomes an error mode)

1. Walk:
    - Maximum speed of all points $\leq$ 20km/h;
    - 85th percentile of speed of all points $\leq$ 10km/h;
    - Average speed of all points $\leq$ 6km/h;
    - Duration $\geq$ 60s.
2. Rail:
    - Distance from first point of speed accelerates to 10km/h to the nearest rail station $\leq$ 150m;
    - Distance from last point that speed is greater than 10km/h to the nearest rail station $\leq$ 150m;
    - Average speed of all points $\geq$ 10km/h;
    - Duration $\geq$ 60s.
3. Bus:
    - Distance from first point of speed accelerates to 10km/h to the nearest bus stop $\leq$ 50m;
    - Distance from last point that speed is greater than 10km/h to the nearest bus stop $\leq$ 50m;
    - Average speed of all points $\geq$ 10km/h;
    - Duration $\geq$ 60s.
4. Bicycle:

- 85th percentile of speed of all point >10km/h and <20km/h;
- Max speed of all points $\leq$ 30km/h;
- Duration $\geq$ 60s.

5. Car:

   - The remaining trip segments with average speed of all points $\geq$ 10km/h;
   - Duration $\geq$ 60s.

Both bus stop and train station data are obtained from `http://datafinder.org` [22] in the form of a shapefile, and then we save the attributes as a CSV file in QGIS. The bus stop data is from "Transit Stops" [23] and the train station data is from "Transitway Stations" [24].

After that we revise bus stop and train station data because the date of these data are different from GPS data. The bus stop data was updated in 2015 and the train station data was recorded in 2014 [22]; on the other hand, the GPS data was recorded in 2011. Therefore, we revise the bus stop (train station) data into the version in 2011 when possible.

1. Train station data

   We revise the train station data based on the attribute of transitway name (Blue line, Green line, Red line, and Northstar Line). Northstar Commuter Rail Line connects to downtown Minneapolis and Big Lake, and the service began in 2009 [25, 26]. The METRO Blue and Green Lines are light-rail services and the Blue Line connects downtown Minneapolis and Mall of America, and started service in 2004 [27]. The Green Line connects to downtown Minneapolis and downtown St.Paul, and opened in 2014 [28]. Therefore, we remove the Green Line data from the train station data.

   The original data also contains the METRO Red Line, but the METRO Red Line is a freeway bus rapid transit service [29]. After checking the trip data, we find that no GPS trips uses the METRO Red line.

2. Bus stop data

   The bus service is operated by Metro Transit and this database has 19,285 active and inactive bus stops in the Twin Cities seven county metropolitan area [23]. From the attribute 'ada_comm' of database, we remove the data of the bus stop before 2011. After this process, the total number of bus stops reduces to 16,545.

As the definition of mode classification indicates, we calculate the distance from the GPS data to the bus stop (train station). Before calculating, we change the coordinate system of the bus stop data (train station) because the GPS data is geographic latitude and longitude; while the bus stop data (train station) uses the Universal Transverse Mercator (UTM). The detail of converting the coordinate system is described in 3.2.5. The code of this algorithm is mentioned in A.4.4.

Table 3.2: Number of travel mode

| **Mode** | **Number of trip** |
|---|---:|
| Walk | 4,499 |
| Bus | 92 |
| Train | 3 |
| Bike | 115 |
| Drive | 4,560 |
| Error | 3,299 |
| Total | 12,568 |

The result of mode classification is in Table 3.2. After calculating the algorithm, 4,560 data are recorded as drive. Besides drive, many trips are classified as walk mode or error mode. It suggests that a lot of trips have small average speed (<10km/h) or small duration time (<60 sec).

We also check the accuracy of mode classification. In the TBI2010 Household Interview Survey, participants recorded their trip for a selected day. While each individual could report up to 3 modes, most of participants recorded only one. TBI data has 160 households' data. Among them, the date of 140 modes match the date of GPS data. Note that there are multiple GPS trips for each selected day. When one of them matches the mode of TBI data, it is recorded as accurate. As Table 3.3 illustrates, 4% of non-auto trips show up as auto. Therefore, our methods ensure few false positives.

Table 3.3: Accuracy of mode identification

| | | Identified as | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Walk | Rail | Bus | Bike | Car | Error | Total | Success rate (%) |
| TBI data | Walk | 14 | 0 | 0 | 0 | 0 | 2 | 16 | 88% |
| | Rail | 0 | 0 | 0 | 0 | 0 | 0 | 0 | #DIV/0! |
| | Bus | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0% |
| | Bike | 0 | 0 | 0 | 1 | 2 | 0 | 3 | 33% |
| | Car | 7 | 0 | 1 | 0 | 106 | 5 | 119 | 89% |
| | Total | 21 | 0 | 1 | 1 | 110 | 7 | 140 | |
| | Success rate (%) | 67% | #DIV/0! | 0% | 100% | 96% | | | |

### 3.2.5 Convert coordinate system

In order to match the GPS data and GIS data, the coordinate system of these two data should be the same; however, the GPS data is geographic latitude and longitude; while the TLG Map uses the Universal Transverse Mercator (UTM). We convert longitude and latitude data into UTM data following standard algorithms [30, 31]. The formula is mentioned in A.1 and the code of this algorithm is mentioned in A.4.6.

### 3.2.6 Matching person ID

And then, we match person ID of GPS data with the TBI2010 Household Interview Survey data in order to add personal information (gender, age, and education) to the GPS data for the statistical analysis. It is noted that file name of initial GPS data is person ID. If person ID doesn't match, the GPS trips removed. The code of this algorithm is mentioned in A.4.7.

After this process, 2,710 trip data (160 individuals) match the TBI data. However, about 40.6 % of data are removed in this process. Thanks to this process, we can extract driving trip data that has personal information.

### 3.2.7 Map-matching process

This algorithm is a main part of this study. Because of measurement error and map resolution, GPS location is not the exact location of the GIS link. Therefore, several

map-matching methods have been developed to identify the correct GIS road link [32, 33]. First we assume that the closest length between GPS and GIS data indicates the accurate map-matching. To begin, we match the GPS data and GIS data when the length between GPS point and GIS link is smallest. In order to estimate the length, the algorithm calculates the area of parallelogram (D) by vector outer product operations, then calculate the height of parallelogram (H) by dividing the area (D) by the length of the base (L), as shown in Figure 3.8.

When the height is smallest, the algorithm extracts the road information (e.g., link length, street name, roadtype, and speed limit) from the GIS data, and combines this data into the GPS data.

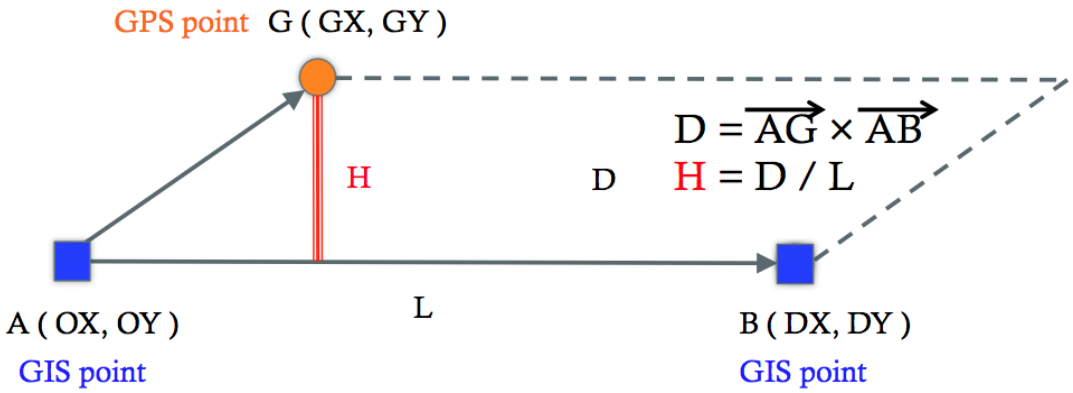



Figure 3.8: Method of calculating nearest GIS link

### Problem of accuracy

Based on the above matching process, we analyze the accuracy of matching data by checking the street name manually. When we check 6 trips of data (shown in Table 3.4), the result shows that 76.0 % of data are accurate on average; however, this percentage of accuracy is not enough to analyze the relationship between road network variables and the degree of speeding. In order to improve the accuracy we need to find the reason of inaccuracy on the map, and we find out that inaccuracy is significant in three types of area.

Table 3.4: Accuracy of matching data (6 trips)

| Trip | # of correct data | # of total data | Accuracy (%) |
|---|---|---|---|
| trip1 | 396 | 607 | 65.2% |
| trip2 | 307 | 374 | 82.1% |
| trip3 | 577 | 802 | 71.9% |
| trip4 | 524 | 642 | 81.6% |
| trip5 | 852 | 949 | 89.8% |
| trip6 | 768 | 1,134 | 67.7% |
| total | 3,424 | 4,508 | 76.0% |

### Curve section

The GIS link between two points is not necessarily a straight line like Figure 3.8; some GPS links are curved as seen Figure 3.9. When the algorithm calculates the shortest length between the GPS data and the curved GIS link, the algorithm is more likely to select the wrong GIS link. This failure significantly occurs in ramp section of interstate or freeway.

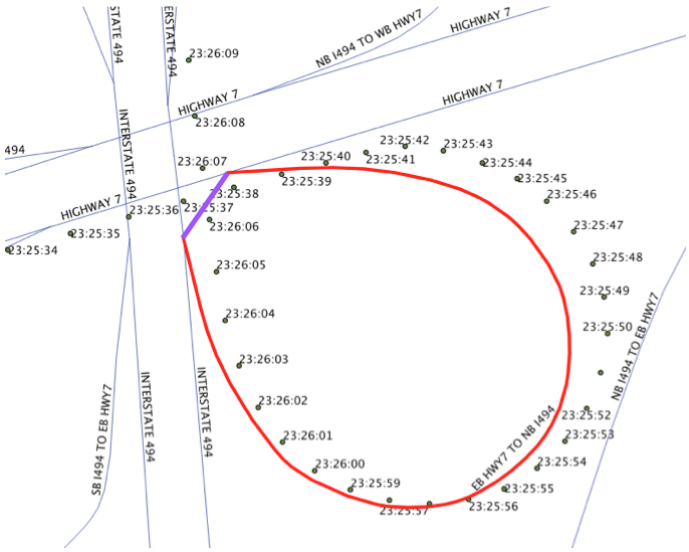



Figure 3.9: Curve Error
(Red line: used GIS link, Purple line: GIS link of algorithm, Green dot: GPS data)

### Intersection

When the GPS point is near an intersection area, a matching failure is likely to occur. As Figure 3.10 shows, the vehicle moves on the vertical road (NICOLLET AVE S); however, the algorithm selects coordinates on the horizontal road (86TH ST W).

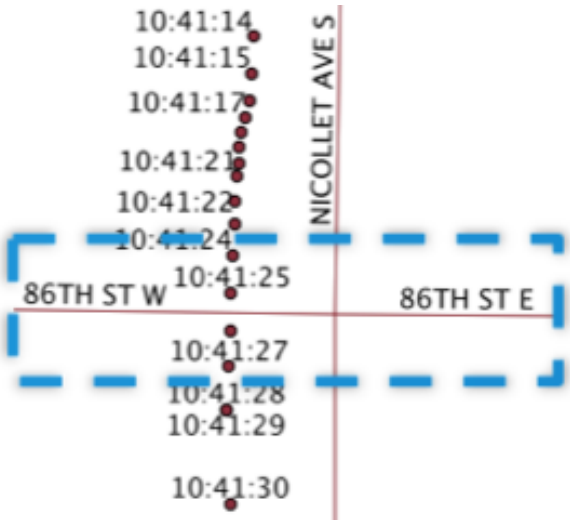



Figure 3.10: Intersection Error
(Line: GIS links, Dot: GPS data)

### Missing link

Although the TLG network data is the most accurate digital map in the Metropolitan area, some road information is missing. When vehicles drive on these missing roads, it is difficult to match the data correctly (Figure 3.11).

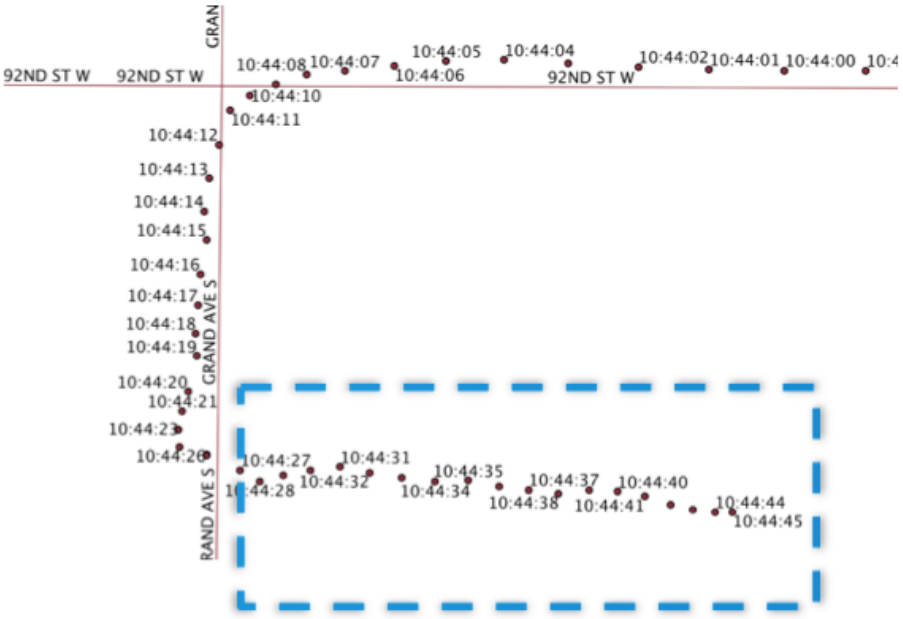



Figure 3.11: Missing Error
(Line: GIS links, Dot: GPS data)

### 3.2.8 Remove matching error and Map-matching

Based on the result of the map-matching algorithm, simply matching to the closest point between GPS and GIS data is far from perfect, and many errors are found at Intersections and due to missing links. To control for that, when GPS data is near an intersection area or far from a mapped link, this GPS point is removed. This process is useful for removing the error, but wastes data. The process is in Figure 3.12. The code of this algorithm is mentioned in A.4.8.

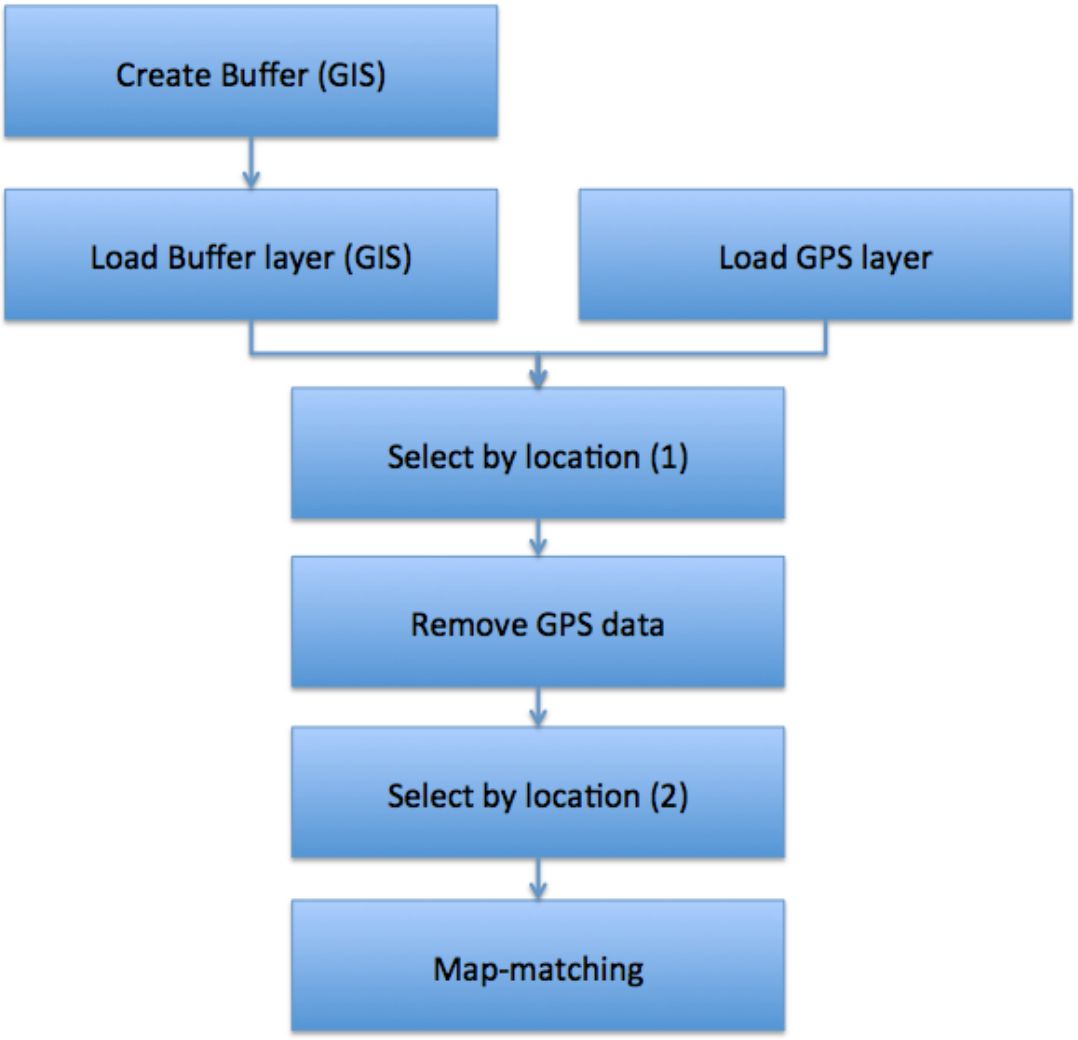



Figure 3.12: Flow chart of Remove matching error and Map-matching

**Narrowing down the number of GIS links**
**(From "Create Buffer" to "Select by location(1)")**

When one GPS point is used to calculate the length to all GIS links, it takes a lot of calculation time because the number of GIS points is enormous (=290,231). Therefore, we narrow down the area of the object to reduce the running time of the algorithm by the function of QGIS.

First we create a buffer for the GIS links (A.2). The initial GIS links are line like in Figure A.4 and it describes the centerline of the road. Then, we create the road area by using the "buffer" function in QGIS. It is true that road width is different depending on the road; however, we set "10m" as the buffer width for all roads. The image of the 10m buffer zone is shown in Figure A.5. It is noted that a 10m buffer shows accurate map-matching results more than other buffer sizes. Moreover, the 10m buffer retains a large amount of data.

And then, we extract specific GIS links that intersect with GPS points by the function "Select by location" (A.2.2). With the process, we can extract GIS links that are close to GPS points, and the extracted GIS link is similar to the driving route. (Figure 3.13)

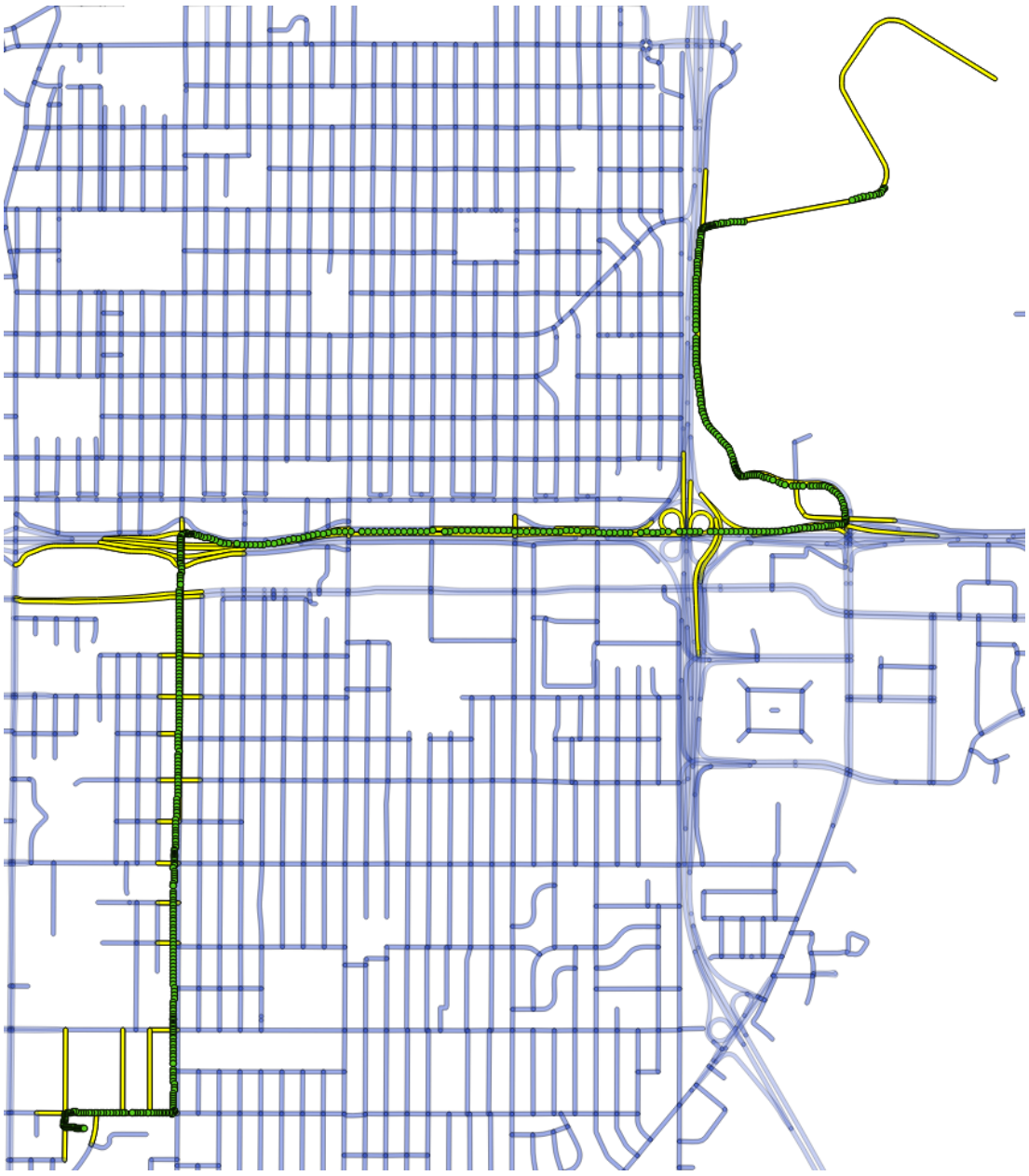


Figure 3.13: Result of narrowing down the number of GIS links
the yellow area (links) is selected by the function “Select by location”

- Problem of measurement error (GPS) and error at intersection area (GIS)
  Although this function can reduce the number of GIS links, it also has a problem. When GPS data is located far from the GIS link because of measurement error, some actual driving routes are removed. Figure 3.14 shows that participants drove on "NICOLLET AVE S"; however, this road is not chosen as a candidate driving route.

  Moreover, unused routes remain like "90TH ST W", "BLAISDELL AVE S", and "PILLSBURY AVE S". It is noted that these streets are unused driving routes; however, these streets are selected because they intersect with actual driving routes.

- Error at concentrated road (GIS)
  If roads are far from each other, the process of "Select by location" chooses only the actual driving route. But if some roads are close to each other, an unused driving route may be chosen. As Figure 3.15 shows, many ramp section of an interstate (such as "WB 1494 TO SB HWY77" and "NB HWY77 TO WB 1494") remain as candidates of driving routes although these links are not actual driving routes. Moreover, as Figure 3.16 shows when local roads are close to interstates, the local roads are also chosen as a candidate drive routes like "78TH ST E".

Especially, when we increase the buffer width, unused driving routes are likely to be chosen. On the other hand, when the buffer width is very narrow, some actual driving routes are removed because many GPS points cannot be plotted within the buffer area. Based on this outcome, we use "10m" as the buffer width.

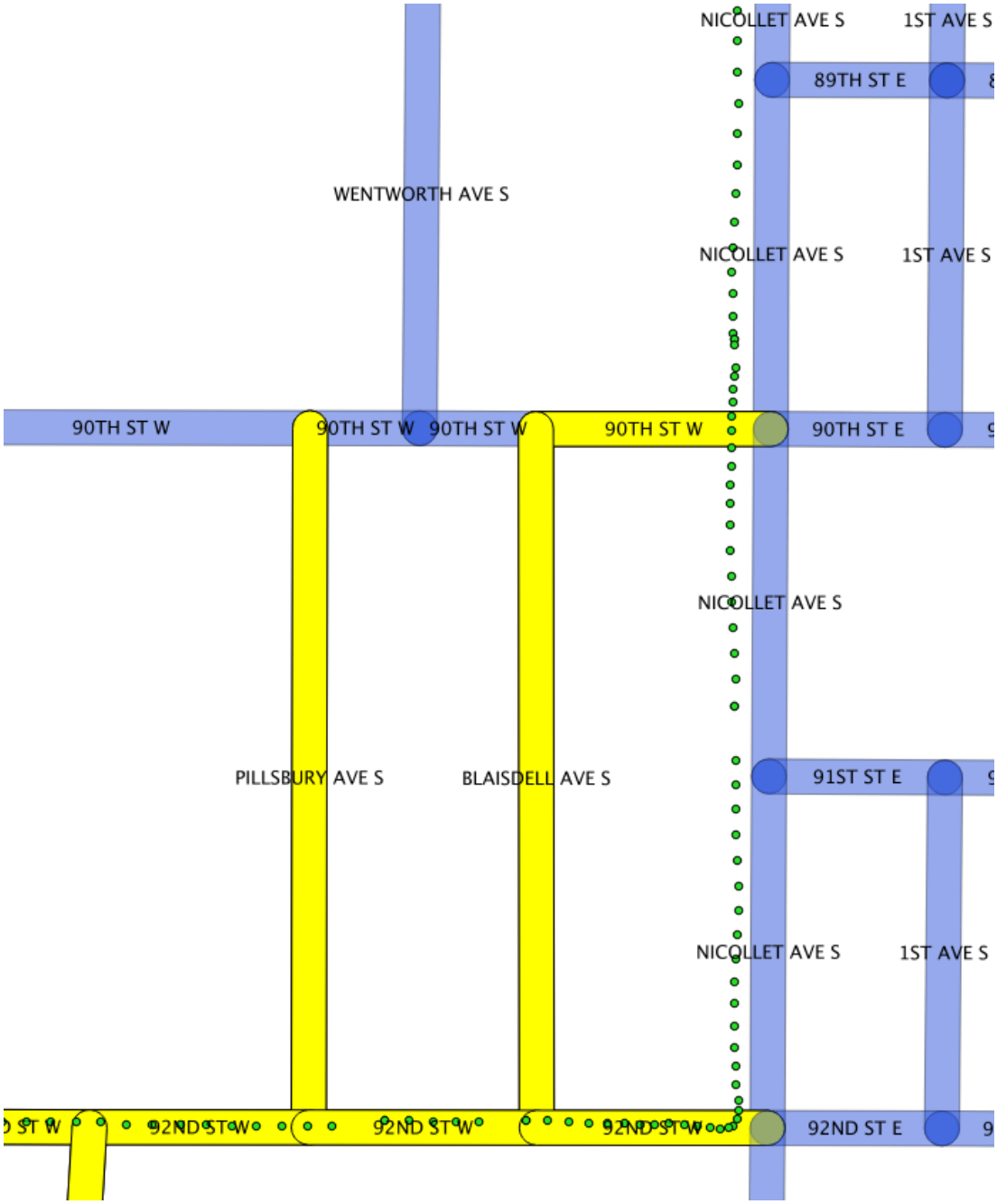



Figure 3.14: Problem of measurement error (GPS) and error at intersection area (GIS)

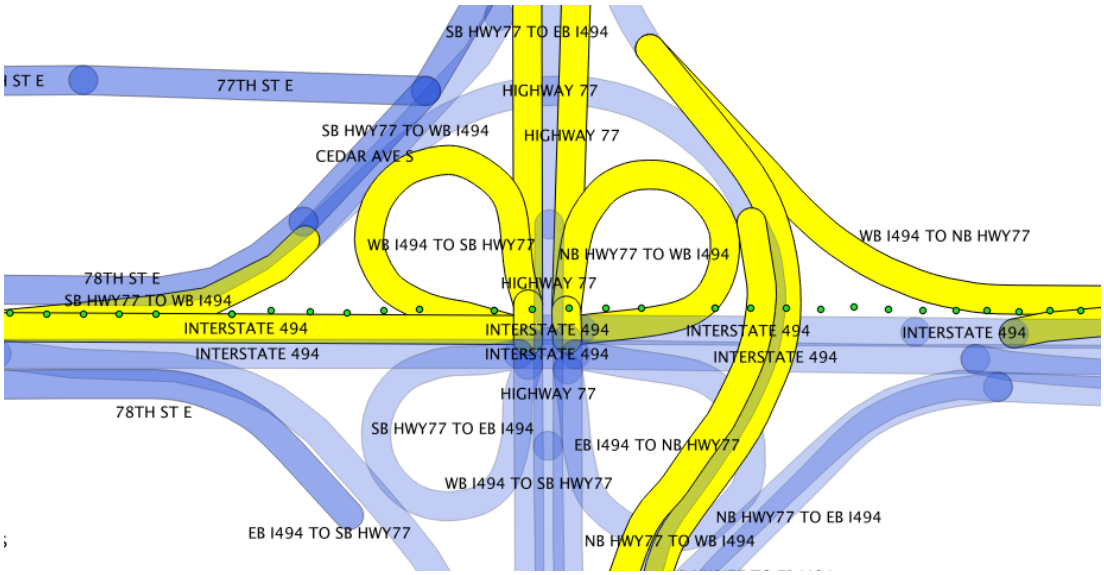



Figure 3.15: Error at concentrated road (GIS) Part.1

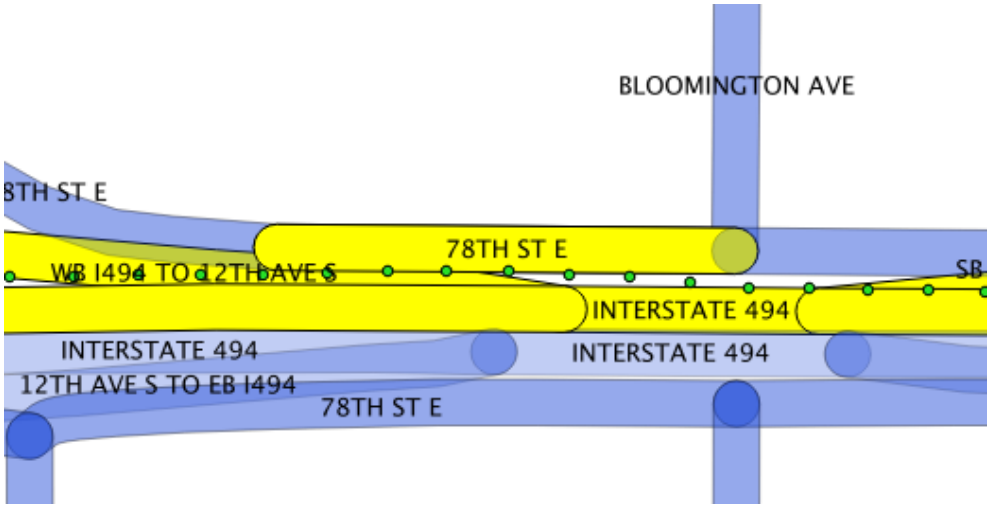



Figure 3.16: Error at concentrated road (GIS) Part.2

**Remove error at intersections and missing links**
**(From "Remove GPS data" to "Select by location(2)")**

Figure 3.17 is the image of the algorithm that removes error data points when the GPS data is in the intersection of two buffers and when they are not within a buffer. Each link is used to create a buffer containing all points within 10m of the GIS link. As Figure 3.17 shows, the intersected GIS link area has more than two layers. The algorithm counts the number when GPS data is inside the range area. When the GPS point is near an intersection, the number of layers becomes greater than two. Also, when GPS data is outside of the range of GIS data, the number of layers is zero. The algorithm only saves the GPS data with one layer.

This process can remove the effect of intersection, but there are some problems with this process. As Figure 3.17 shows, when the streets are laid out in a grid running north-south and east-west, the covering area of removed GPS data is small. However, when the streets are laid out diagonally or adjacent streets are close to each other (Figure 3.18), many areas have greater than two layers and a lot of GPS data are removed because of this process.

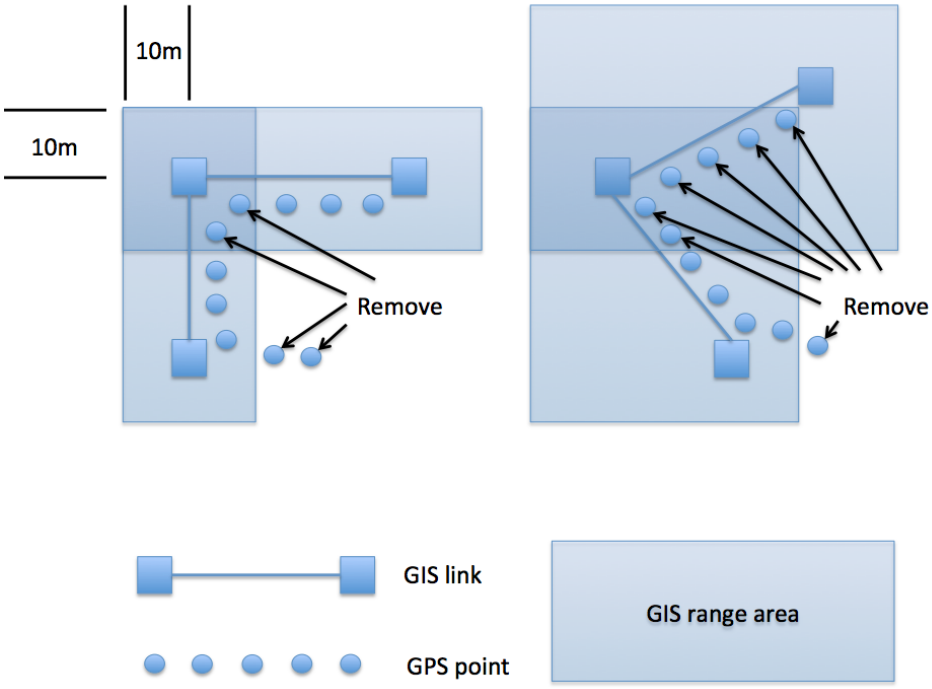



Figure 3.17: Image of algorithm Part.1 (Left:grid links, Right:diagonal links)

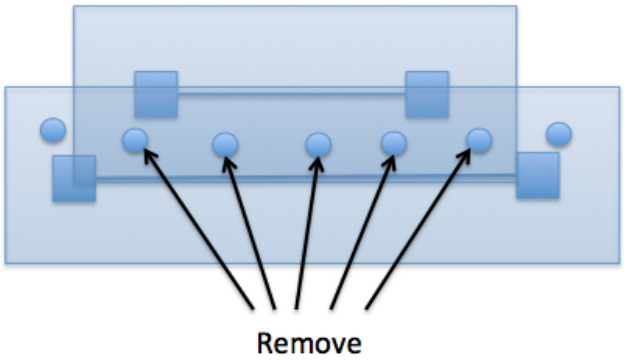



Figure 3.18: Image of algorithm Part.2

After removing GPS data at intersections and on missing links, we extract GIS links again by using the function "Select by location" that is the same process as A.2.2.

Figure 3.19 shows that this method can remove some unused links and save actual driving route links. The problem of this process is that most of the GPS data is removed when several road links are close to each other. For example, parallel roads and ramps are near the mainline of the interstate as shown in Figure 3.15. Even if drivers use the interstate, these GPS data are likely to be removed because of the process as we mentioned in Figure 3.18.

Moreover, we can see other errors that unused routes (links) are chosen at the bottom-right part of the map (Figure 3.19). As Figure 3.14 shows, GPS data is more than 10m away from the actual driving route ("NICOLLET AVE S"). In this case, actual links are removed and unused links remain.

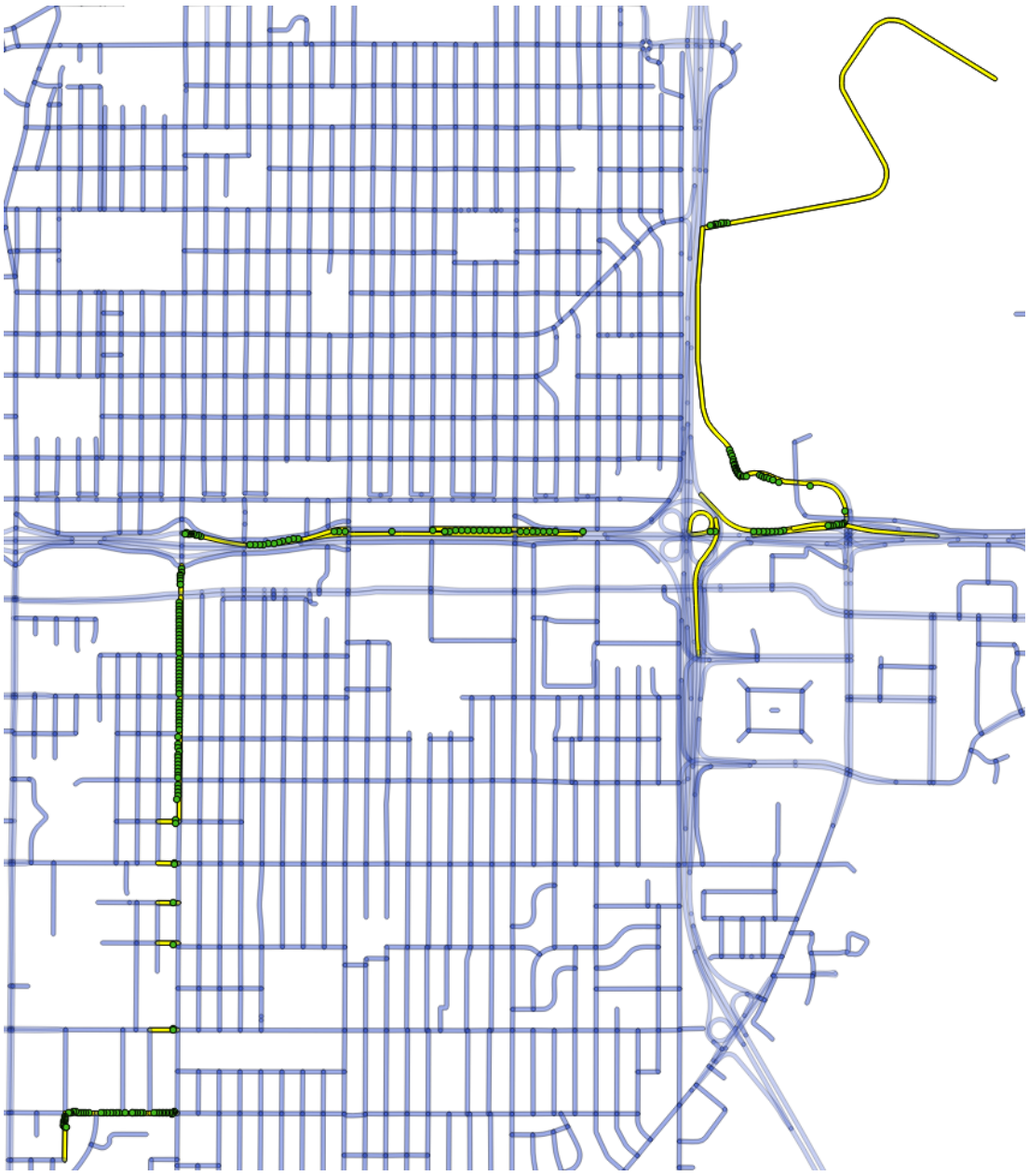


Figure 3.19: Result of removing error at intersection area and missing links
Yellow area (links) is selected by the function “Select by location”

### Map-matching

After this process, we match the extracted GPS data and GIS link by calculating the smallest length between the GPS data and GIS link (Figure 3.8). Also, we analyze the accuracy from the result of map-matching manually by visual inspection in QGIS. Visual inspection shows that 97.4% of matched points on 20 trips are accurate (Table 3.5), and therefore this result is satisfactory. The problem is that about 51.0% of the GPS data are removed and some driving information during the trip is missing due to the process. Therefore, we cannot analyze continuous trip data. In order to ensure few false positives, there are many false negatives. After this algorithm, the total number of trips is 2,710 (160 drivers).

Table 3.5: Accuracy of matching data (20 trips)

| **Trip** | **# of accuracy** | **# of total data** | **Accuracy (%)** | | **# of initial trip data** | **% of removal** |
|---|---|---|---|---|---|---|
| Trip1 | 7,103 | 7,283 | 97.5% | | 12,144 | 40.0% |
| Trip2 | 3,502 | 3,544 | 98.8% | | 5,786 | 38.7% |
| Trip3 | 2,945 | 3,048 | 96.6% | | 16,152 | 81.1% |
| Trip4 | 2,980 | 3,063 | 97.3% | | 6,056 | 49.4% |
| Trip5 | 2,645 | 2,655 | 99.6% | | 3,857 | 31.2% |
| Trip6 | 2,502 | 2,540 | 98.5% | | 5,814 | 56.3% |
| Trip7 | 2,496 | 2,536 | 98.4% | | 4,688 | 45.9% |
| Trip8 | 2,442 | 2,473 | 98.7% | | 4,666 | 47.0% |
| Trip9 | 2,337 | 2,439 | 95.8% | | 5,152 | 52.7% |
| Trip10 | 2,300 | 2,351 | 97.8% | | 4,783 | 50.8% |
| Trip11 | 2,280 | 2,390 | 95.4% | | 3,739 | 36.1% |
| Trip12 | 2,325 | 2,344 | 99.2% | | 5,637 | 58.4% |
| Trip13 | 2,176 | 2,311 | 94.2% | | 4,189 | 44.8% |
| Trip14 | 2,003 | 2,236 | 89.6% | | 3,722 | 39.9% |
| Trip15 | 2,138 | 2,171 | 98.5% | | 3,885 | 44.1% |
| Trip16 | 2,103 | 2,160 | 97.4% | | 3,586 | 39.8% |
| Trip17 | 2,063 | 2,117 | 97.4% | | 4,032 | 47.5% |
| Trip18 | 2,096 | 2,106 | 99.5% | | 5,468 | 61.5% |
| Trip19 | 2,044 | 2,100 | 97.3% | | 3,051 | 31.2% |
| Trip20 | 2,115 | 2,136 | 99.0% | | 3,730 | 42.7% |
| Total | 52,595 | 54,003 | 97.4% | | 110,137 | 51.0% |

As Table 3.6 shows, the results of map-matching are recorded as CSV file. CSV file has the following attributes;

1. Location data [Longitude, Latitude]

2. Driving speed [Speed(kilometer)]

3. Time of date [DD/MM/YY, HH:MM:SS]

4. Location data (UTM) [GX, GY]

5. Length between GPS data and GIS link [Min_height]

6. Street name [Streetname]

7. Street length (Link length, Unit:meter) [Streetlength]

8. Road type [Roadtype]

9. Speed limit of road link [Speedlimit(mph)]

Attributes 1,2,3 are from original GPS data, 4 is the result of coordinate conversion, 5 is the calculation result (Figure 3.8), 6,7,8,9 is from original GIS map data.

## 3.3 Data processing for Statistical Analysis

With the purpose of analyzing the hypotheses described in the next chapter, we need to calculate the speeding from the driving speed of GPS data and the speed limit of GIS

Table 3.6: Example of matching data (csv-file)

| Longitude | Latitude | Speed(kilometer) | DD/MM/YY | HH:MM:SS |
|---|---|---|---|---|
| -93.245091 | 44.875816 | 59 | 24/6/2011 | 10:34:18 |
| -93.245296 | 44.875791 | 56 | 24/6/2011 | 10:34:19 |
| -93.245486 | 44.875768 | 50 | 24/6/2011 | 10:34:20 |
| -93.245661 | 44.875733 | 48 | 24/6/2011 | 10:34:21 |
| -93.245816 | 44.875711 | 41 | 24/6/2011 | 10:34:22 |

| GX | GY | Min_hight | Streetname | Streetlength |
|---|---|---|---|---|
| 480641.3983 | 4969184.497 | 29.69856906 | CARGO RD | 2444.999086 |
| 480625.1979 | 4969181.768 | 24.76099366 | CARGO RD | 2444.999086 |
| 480610.183 | 4969179.259 | 20.16780883 | CARGO RD | 2444.999086 |
| 480596.3487 | 4969175.413 | 17.30193749 | CARGO RD | 2444.999086 |
| 480584.0986 | 4969173.006 | 13.87454209 | CARGO RD | 2444.999086 |

| Roadtype | Speedlimit(mph) |
|---|---|
| RD | 35 |
| RD | 35 |
| RD | 35 |
| RD | 35 |
| RD | 35 |

map. Moreover, we also need to add the personal information to GPS data in order to analyze the effect of the speeding statistically.

Figure 3.20 is a flow of data processing. First we remove unnecessary data (young driver and speed limit = 0 mph), and then we change the format that can be analyzed in statistical package *R*.

### 3.3.1 Remove data (age is less than 16)

When we check the driver's data which is TBI 2010 Household Interview Survey Data, some drivers are under 16. Therefore, we remove the trip data when the driver is under 16 years of age. The code of this algorithm is mentioned in A.4.9.

The algorithm removed 128 trip data (8 individuals) for whom the driver's age is under 16. Subsequently, the number of trip data decreases to 2,582 (152 individuals).

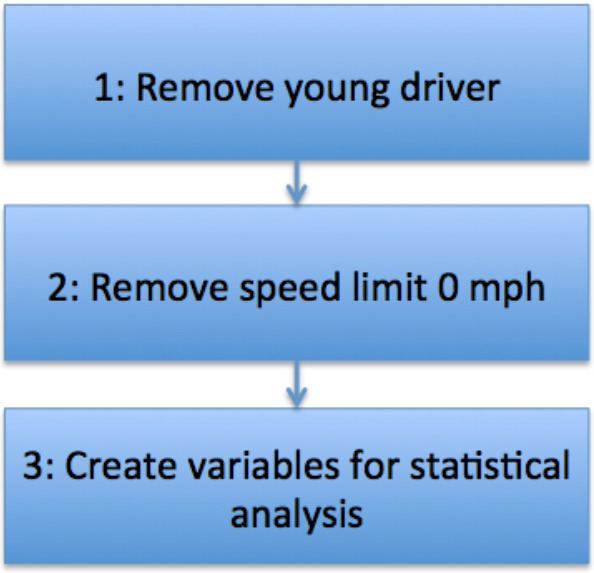



Figure 3.20: Flow chart of data processing

### 3.3.2 Remove speed limit 0 mph

After the map-matching process, we find out that some speed limits are 0 mph. In the analysis part, we will calculate how much the driver exceeds the speed limit (= driving speed / speed limit). However, the calculation doesn't work when the speed limit is 0 (mph). Therefore, we remove the GPS data where the speed limit is 0 mph. The code of this algorithm is mentioned in A.4.10.

No trip data are removed in this process. However, we remove 22 trip data apart from this process because no GPS data are recorded due to the map-matching process.

### 3.3.3 Create variables for statistical analysis

We create the format of dependent variable and independent variables for the statistical analysis. The code of this algorithm is mentioned in A.4.11.

- Convert the unit of speed limit
  When we compare the driving speed with the speed limit, the unit should be matched. However, driving speed is [km/h] while speed limit is [mph], so we convert the unit of speed limit from [mph] to [km/h].

- Change the hour from UTC to MN time

  Time zone in GPS data is Coordinated Universal Time (UTC). That is why we change the timezone from UTC to MN time. It is noted that the time difference between UTC and Minnesota depends on the date because of daylight saving time [34].

  "-6h" during standard time (01/01/2011 - 03/12/2011), (11/07/2011 - 12/31/2011)

  "-5h" during daylight Saving time (03/13/2011 - 11/06/2011)

- Calculate the speeding

  We calculate the speeding behavior from two perspectives:

  *1: Whether speeding or not (Percentage of speeding)*

  When the driving speed is over the speed limit, the value becomes 1. If not, the value becomes 0.

  *2: How much the driver exceeds the speed limit (Degree of speeding)*

  The result of percentage of speeding is critical because "driving at 60 mph in 40 mph zone" and "driving at 41 mph in 40 mph zone" are identical outcomes. Therefore, we also analyze another speeding behavior by calculating how much each GPS data exceeds the speed limit. $DegreeOfSpeeding$ is defined in Equation 3.1.

$$DegreeOfSpeeding = \frac{DrivingSpeed}{SpeedLimit} \tag{3.1}$$

  When the value is 1, driving speed equals the speed limit, and when the value is more than 1, the data shows speeding.

- Create dummy variables and add personal information

  In order to compute the multi-variate regression analysis in $R$, we create dummy variables for speed limit zone and time. Moreover, we add personal information (age, gender, education) that is recorded in Household Survey data to GPS data by matching trip ID. These personal data are also changed to dummy variables. The detail of these variables is mentioned in Table 4.3.

Finally, the remaining 2,560 trips (from 152 individuals) are compiled to small sample data that can be analyzed in $R$. The code of this algorithm is mentioned in A.4.12.

# Chapter 4

# Analysis

## 4.1 Hypotheses

We hypothesize that speeding is affected by road type (hierarchy of the network) and road characteristics.

1. Hierarchy (Road type) – *Hypothesis: Position near the top of the hierarchy of roads is correlated with speeding*

   We analyze the effect of hierarchy in the network. Compared with local roads, the driver is less affected by external factors on freeways. GIS data contains speed limit, street name and road type, and therefore we can calculate speeding depending on each road type.

2. Link length – *Hypothesis: Long link length is correlated with speeding*

   We investigate the effect of link length on speeding. When the length is small, there are many intersections in the network. The GIS network has link length data, so it is possible to calculate the relationship between link length and speeding behavior.

Table 4.1: Total data in each speed limit zone

| Speed limit(mph) | GPS data | | GIS link | |
|---|---|---|---|---|
| | # of data | percentage | # of link | percentage |
| 5 | 759 | 0.1% | 181 | 0.1% |
| 10 | 2,364 | 0.2% | 354 | 0.1% |
| 20 | 80 | 0.0% | 227 | 0.1% |
| 25 | 20,867 | 1.8% | 4,245 | 1.5% |
| 30 | 133,339 | 11.4% | 24,204 | 8.3% |
| 35 | 377,960 | 32.2% | 223,198 | 76.9% |
| 40 | 346,876 | 29.6% | 29,209 | 10.1% |
| 45 | 22,624 | 1.9% | 1,921 | 0.7% |
| 50 | 20,001 | 1.7% | 939 | 0.3% |
| 55 | 118,160 | 10.1% | 3,523 | 1.2% |
| 60 | 91,164 | 7.8% | 836 | 0.3% |
| 65 | 28,416 | 2.4% | 768 | 0.3% |
| 70 | 9,493 | 0.8% | 214 | 0.1% |
| Total | 1,172,103 | 100.0% | 290,231 | 100.0% |

## 4.2 Analysis Result

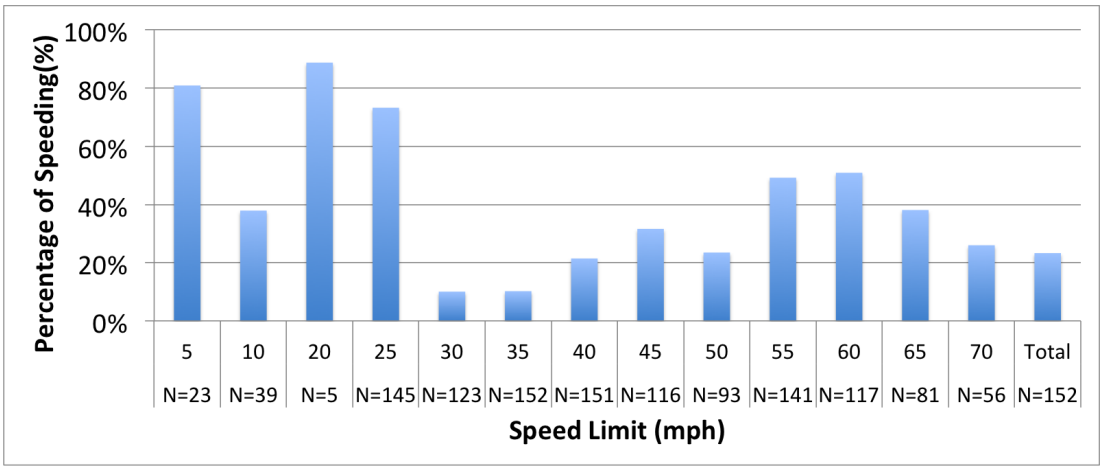
For 2,560 trips, we analyze the relationship between road network variables and speeding. The total amount of data are 1,172,103 GPS points. As Table 4.1 illustrates, most of the data are concentrated on 30, 35, 40 and 55 mph zone. Moreover, relatively few points are in low speed limit zones (less than 20 mph) and high speed limit zones (70 mph). (Note some travel that is retained in the analysis is on private roads or in parking lots, this explains travel on roads with 5 mph speed limits).

### 4.2.1 Road network structure

#### Hierarchy (Road type)

The bar chart shows the percentage of speeding on each speed limit zone (Figure 4.1). Overall, 23.3 percent of GPS observations exceed the speed limit. It can also be seen that speeding behavior is significant in low speed limit zones (e.g. from 5 to 25 mph) and high speed limit zones (e.g. from 55 to 65 mph). The number of individuals (N = *) suggests that most participants encountered speed limits between 25 and 65 mph while driving.

The 20 mph zone stand at 88.8%, and it is followed by 5 and 25 mph zone at 80.9%, 73.3% respectively. However, the amount of data in low speed limit zones is relatively small, therefore more data are required in order to improve the reliability of this result. In addition to low speed limit zones, speeding percentage in high speed limit zones (from 45 to 70 mph) is also more than average. On the other hand, speed limits from 30 to 40 mph, have a lower speeding percentage than lower and higher speed limit zones. Especially, 30 mph is the lowest number at 10.1%. The code of computing and counting number in $R$ is mentioned in A.4.19.




(N = *) indicates number of individuals persons encountering that particular speed limit

Figure 4.1: Percentage of speeding across speed limit zones

The provided box plot graph illustrates how much the driver exceeds the speed limit across each speed limit zone (Figure 4.2). When the speed limit is low (5, 20, and 25 mph zone), the driver exceeds the speed limit greatly more than other speed limit zones. The value of first quartile is about more than 1.0, therefore about 75% of data indicate the speeding in this speed limit zone. On the other hand, in intermediate and high speed limit zones (From 30 mph to 70 mph), speeding is not significant because many data are close to or less than 1.0. The code of creating boxplot in $R$ is mentioned in A.4.20.

When we check the median in box plot, it looks like a "V-shape" across speed limit zones. From 5 mph to 30 mph, the value of median are likely to increase as the

speed limit becomes small. In contrast, from 30 mph to 70 mph the median slightly rises as the speed limit increases. This result suggests that many driver exceed the speed limit significantly in low speed limit zones; on the other hand, they tend to drive proportionally nearer the speed limit in high speed limit zones. For intermediate speed limit zones, they are likely to drive below the speed limit.

Compared the result between the percentage of speeding (Figure 4.1) and the degree of speeding (Figure 4.2), we can see a similar trend although there are small differences.

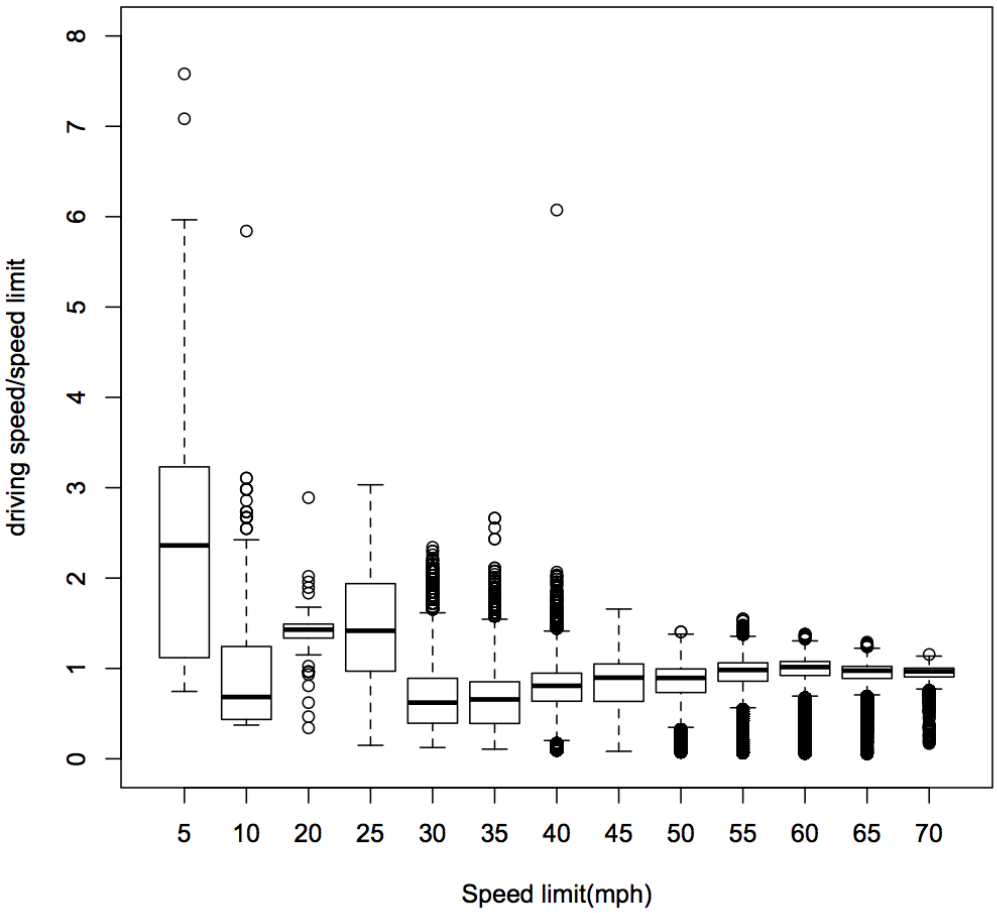
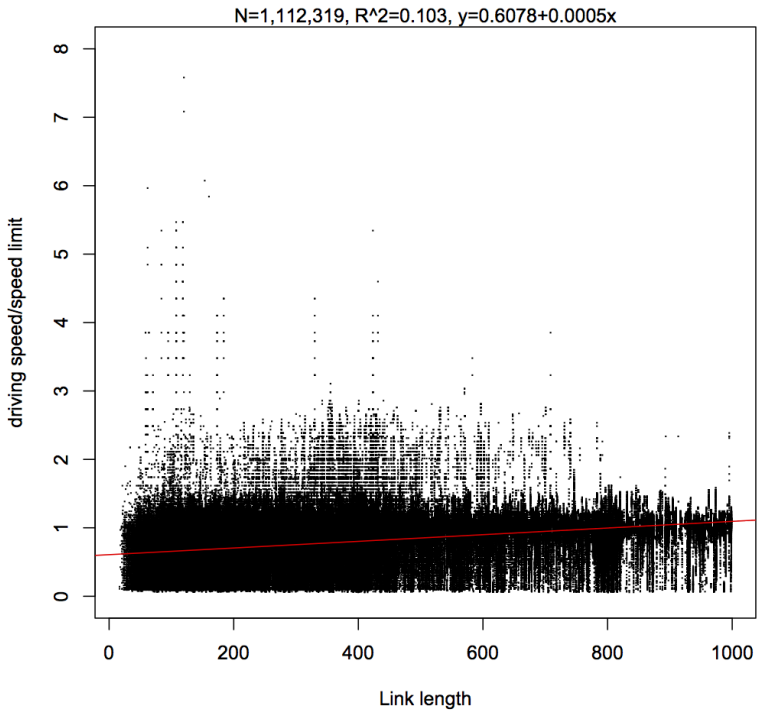



Figure 4.2: Degree of speeding across speed limit zones

### Link length

Figure 4.3 illustrates the relationship between link length and speeding behavior. Y axis is $Degree of Speeding$ defined in Equation 3.1.




(link length is less than 1,000m)

Figure 4.3: Relationship between link length and speeding

We get the information of link length from TLG network, and generally link length indicates the length between intersections. However, some links are divided into several parts between intersections especially in interstate and freeway zones. Exact link length on highways, which means the length between interchange, is much longer; however, highway link data is divided into several sections between interchanges. Therefore, we should be careful about the result of link length in high speed limit zones.

Most link lengths are less than 1,000m and some data in long link length might be too far away from the fitted regression line because of an outlier problem (Figure A.12).

To reduce data issues, we analyze the data for link length $\leq$ 1,000m.

Figure 4.3 and Table 4.2 shows that when link length is longer, the driver is more likely to exceed the speed limit. According to the regression results (Equation_1: Table 4.2), t-value is large and the p-value of overall F-test is very small. Therefore it appears that this model fits the data well; however, the value of $R^2$ is relatively small (0.103). Next, we compute the polynomial model (Equation 4.2) which fits the data better.

In the polynomial model, the value of $E(speeding|streetlength)$ reaches a maximum when street length is 761.2m. It implies that speeding is likely to decrease as the link length becomes longer than 761.2m. The code of creating graph and regression result in $R$ is mentioned in A.4.21.

**Equation_1**

$$E(speeding|streetlength) = \beta_0 + \beta_1 streetlength \tag{4.1}$$

**Equation_2**

$$E(speeding|streetlength) = \beta_0 + \beta_1 streetlength + \beta_2 streetlength^2 \tag{4.2}$$

Table 4.2: Regression Results

**Dependent variable : Speeding** (link length is less than 1,000m)

**Coefficients:**

| | **Equation_1** | | | **Equation_2** | | |
|---|---|---|---|---|---|---|
| | **Estimate** | **t-value** | | **Estimate** | **t-value** | |
| Intercept | 6.078E-01 | 1197.5 | *** | 5.142E-01 | 596.0 | *** |
| street_length | 4.881E-04 | 358.2 | *** | 1.142E-03 | 225.0 | *** |
| street_lengthˆ2 | - | - | | -7.501E-07 | -133.6 | *** |

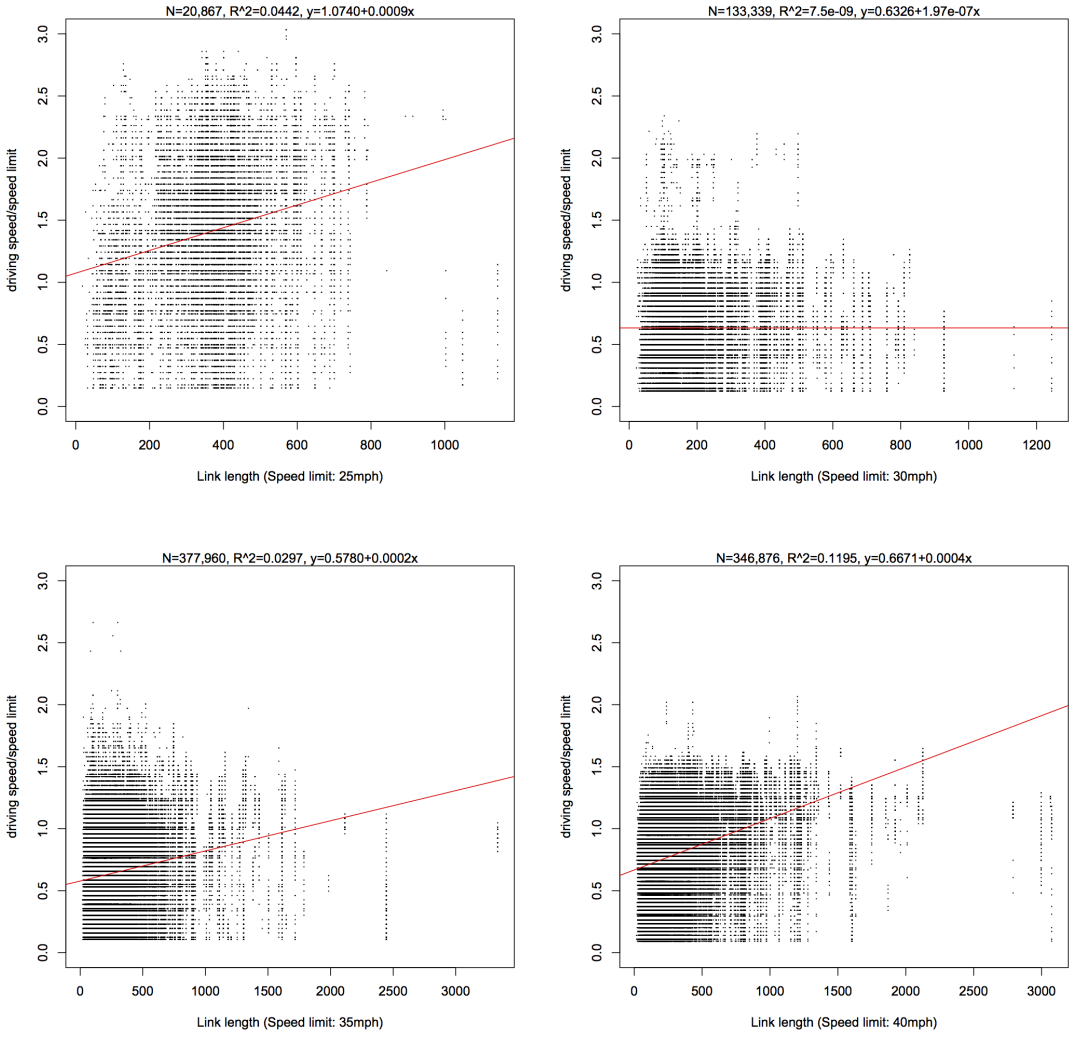
—

signif. Codes: 0 '***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | **Equation_1** | **Equation_2** |
|---|---|---|
| Multiple R-squared | 0.103 | 0.118 |
| F-statistic | 1.28e+05 | 7.41e+04 |
| p-value | <2.2e-16 | <2.2e-16 |

The result of Figure 4.3 is from all roads, but driving condition depends on road type, therefore, we analyze this relationship according to each speed limit zone. Figure 4.4

illustrates that the relationship between link length and speeding on the speed limit zone. An interesting finding is that the graph shows there is little correlation between link length and speeding at 30 mph zones. On the other hand, there is positive correlation at other speed limit zones.




(Top left: 25mph, Top right: 30mph, Bottom left: 35mph, Bottom right: 40mph)

Figure 4.4: Speeding by link length depending on speed limit zones

### 4.2.2 Time of day

The bar chart illustrates the proportion exceeding the speed limit across the time of day (Figure 4.5). Overall, speeding behavior varies by time of day and the percentage of speeding during daytime driving is lower than nighttime driving, with a peak at 4am of 64.7 %, and it is followed by 5am and 11pm (hour 23) at 40.1 % and 39.8% respectively. On the other hand, 8am and 9am are smaller than other morning hours, it appears that this result is related to the effect of rush hour. It is assumed that drivers cannot drive faster because of traffic conditions. Although the percentage at 3am is considerably lower than other time zones (3.7%), the number of participants are very small (N=3) and it appears that this result might be significantly influenced by personal factors.

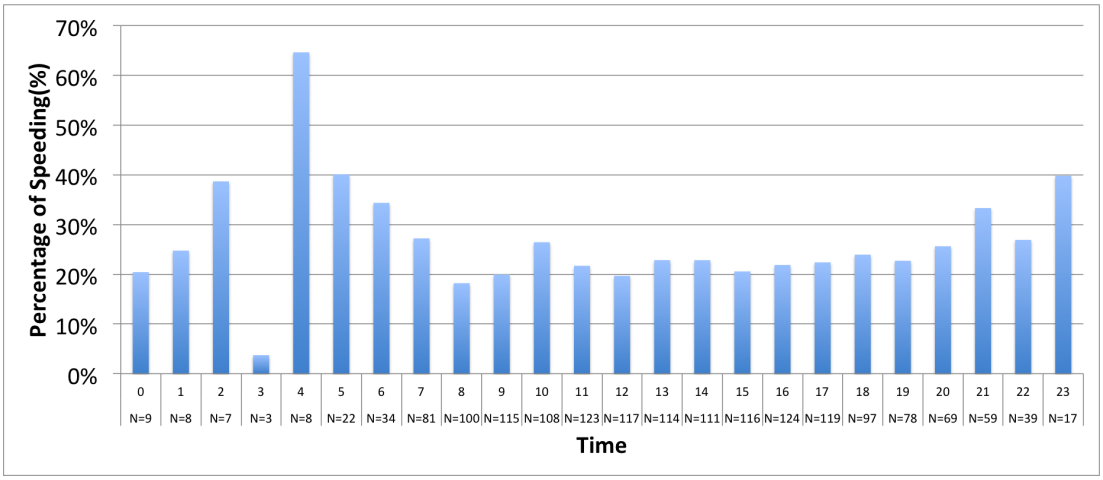



Figure 4.5: Percentage of speeding by time of day

The provided box plot graph illustrates how much the driver exceeds the speed limit across each time (Figure 4.6). Midnight shows a high degree of speeding compared with other times; however, it seems that there is little difference. In order to check this difference statistically, we calculate the t-test for the means of two time.

The Null hypothesis is as follows:

$$\mu_i = \mu_j (i \neq j) \tag{4.3}$$

Firstly we measure whether the mean value of 4am differs from 7am ($\mu_4 = \mu_7$). The

average value at 4am is 0.947 while 7am is 0.784 (Table A.7). According to the t-test, t-value: 25.300 and p-value: 0. Therefore, we reject the null hypothesis and it indicates that there is a significant difference. Secondly, we compute t-values at 12pm and 2pm ($\mu_{12} = \mu_{14}$). Average value at 12pm is 0.754 while 14am is 0.756. According to the t-test, t-value: -0.808, p-value: 0.419. Therefore, we cannot reject the null hypothesis. The result of significant code between all time is shown in Figure A.13. Although the difference of mean value is not significant between some time, mostly there is a significant difference between each time. The code of this algorithm is in A.4.13.

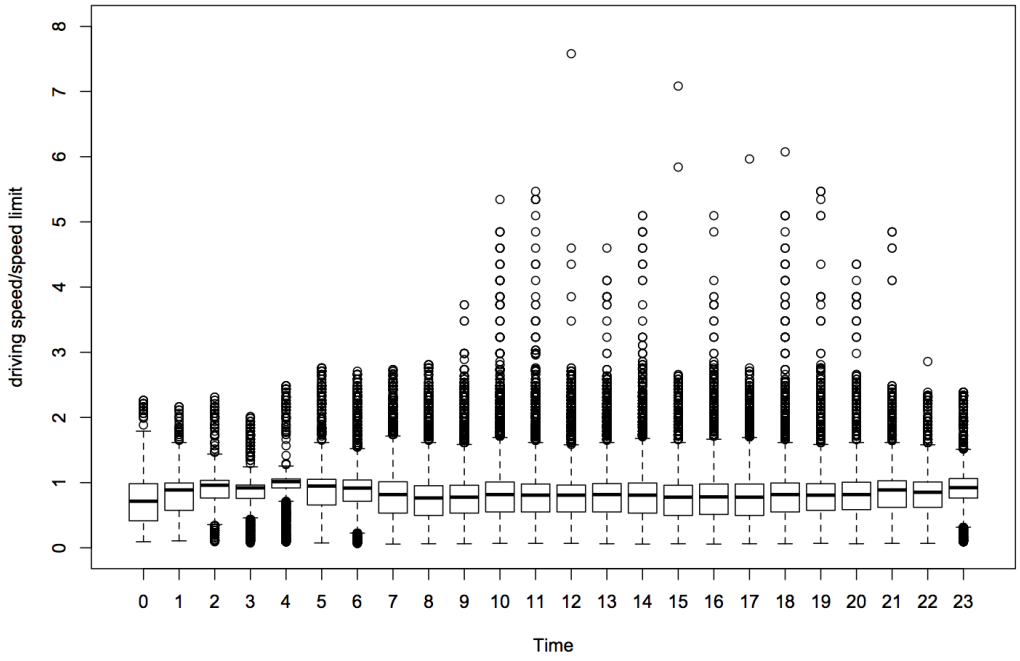



Figure 4.6: Degree of speeding across time

### 4.2.3 Personal information

Several researchers mention that speeding behavior depends on the driver [1, 3]. Unlike loop detector data, GPS data is linked to personal information. We choose gender, age, and education as personal data to analyze the data of 152 participants.

### Gender

Figure 4.7 shows speeding is undifferentiated by gender. While the number of participants is large (male drivers: 65, female drivers: 87), the percentage of speeding for male drivers is same as female drivers (Male, Female: 23.3%).

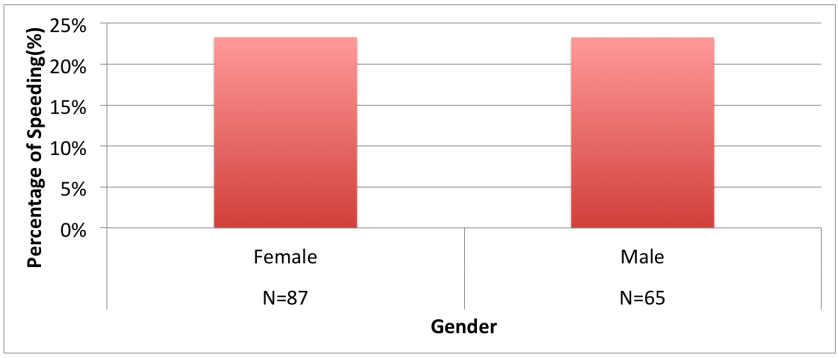



Figure 4.7: Percentage of speeding by gender

The given box plot compares the degree of speeding by gender (Figure 4.8). It appears that there is little difference between gender because average value is almost similar; Male: 0.754, Female: 0.775 (Table A.8). For checking this hypothesis, we measure whether the average value differs significantly across samples.
The null hypothesis is as follows:

$$\mu_{male} = \mu_{female} \tag{4.4}$$

According to the t-test, t-value: 33.660, p-value: 0. Therefore, we reject the null hypothesis and there is a difference between genders from statistical results. The code of this algorithm is mentioned in A.4.14.

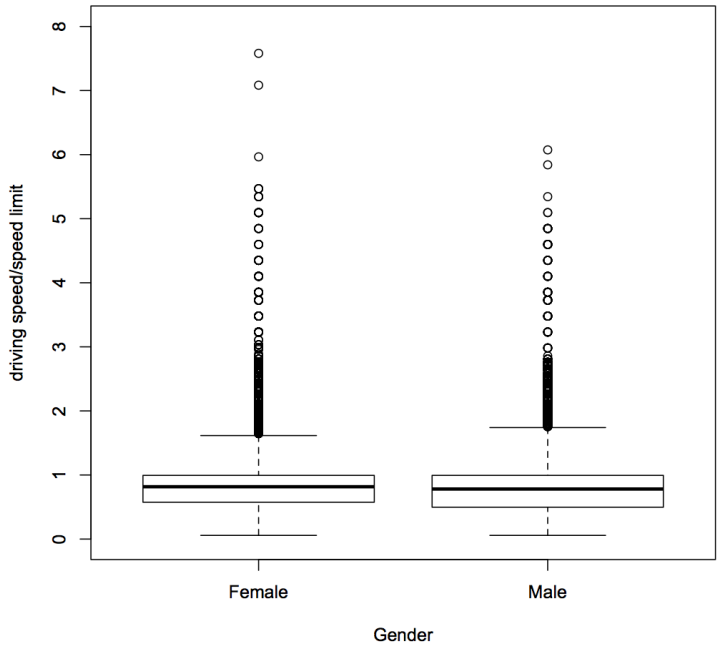



Figure 4.8: Degree of speeding by gender

### Age

Figure 4.9 presents the result of the percentage of speeding across age. The age group from 25 to 34 illustrates the highest speeding behavior at 34.9%. Although the number of participants are small, elderly drivers (85 and over) are also likely to exceed the speed limit (33.2%). It is followed by 55 to 64 and 35 to 44 at 25.0% and 24.0%, respectively. On the other hand, the age group who are 75 to 84 shows the lowest speeding at 13.5%.

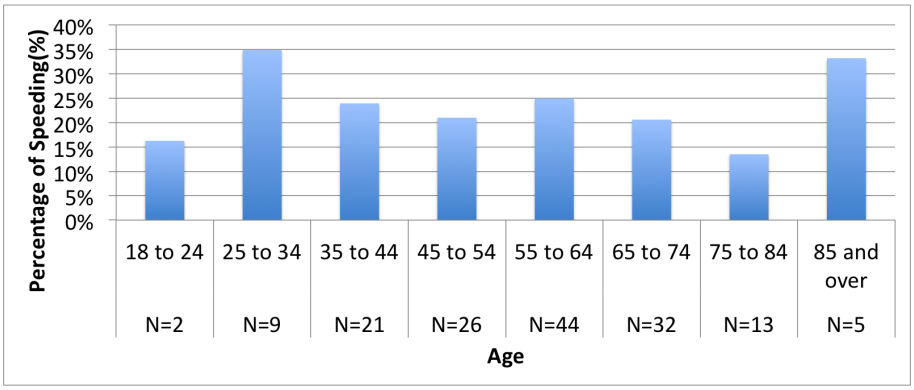



Figure 4.9: Percentage of speeding by age

Figure 4.10 illustrates the difference of degree of speeding across age groups. It appears that the degree of speeding is not different depending on the age group; however, all t-value is large according to the t-test (Equation 4.3). Therefore, we reject the null hypothesis and there is a significant difference between each age group (Figure A.14). The code of this algorithm is in A.4.15.

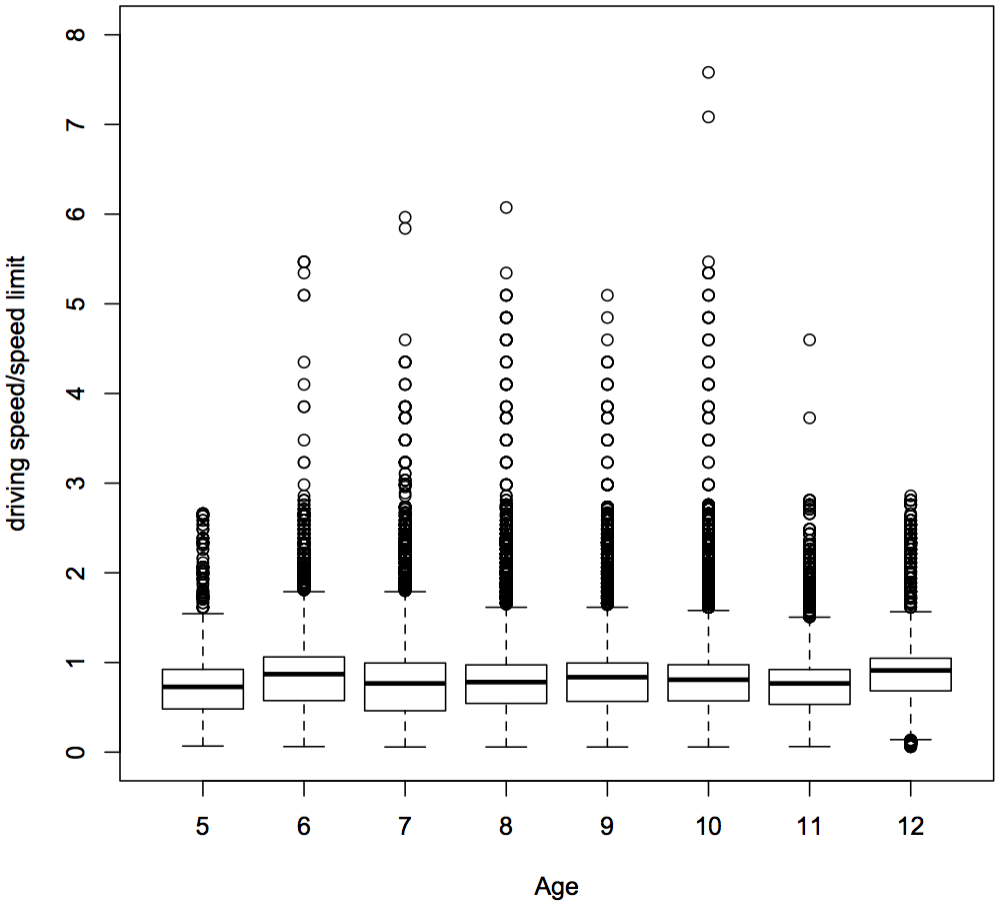



Figure 4.10: Degree of speeding across age
(5: 18 to 24, 6: 25 to 34, 7: 35 to 44, 8: 45 to 54, 9: 55 to 64, 10: 65 to 74, 11: 75 to 84, 12: 85 and over)

### Education

Figure 4.11 illustrates the result of education, and education level is divided into 3 groups; education_0 (High school graduate and Vocational/Technical training), education_1 (Some college and Associates degree) and education_2 (Bachelors degree and Graduate/Post-graduate degree). 'education_0' means lower education level while 'education_2' means higher education level. According to Figure 4.11, we cannot see a significant difference between education levels because education_0 (23.1%), education_1 (24.0%), and education_2 (23.1%).

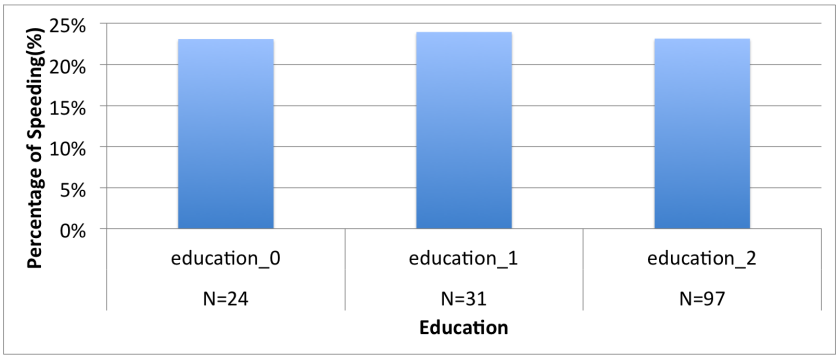



Figure 4.11: Speeding by education level

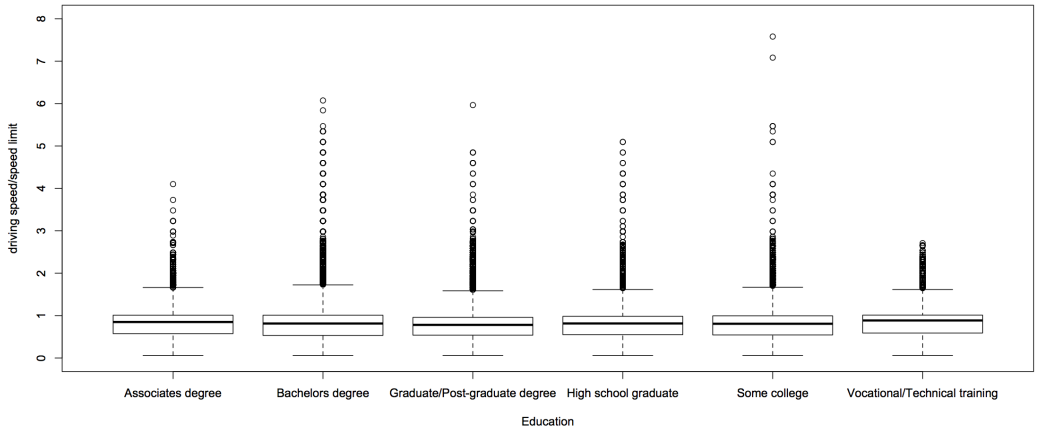



Figure 4.12: Degree of speeding across education

Figure 4.12 illustrates the difference of degree of speeding across education level. It seems that the mean value of each education status is similar; however, all t-value are large according to the t-test (Equation 4.3). Therefore, we reject the null hypothesis and there is a significant difference between each education level (Figure A.15). The code of this algorithm is in A.4.16.

### Speeding difference depends on driver

Figure 4.13 illustrates the difference of speeding across drivers, and the result shows that there are significant differences depending on the driver. Average is 22.5% and maximum number is 66.5%; on the other hand, minimum number is 0.0%. This graph looks like an inverse square law. The code of this algorithm is in A.4.17.

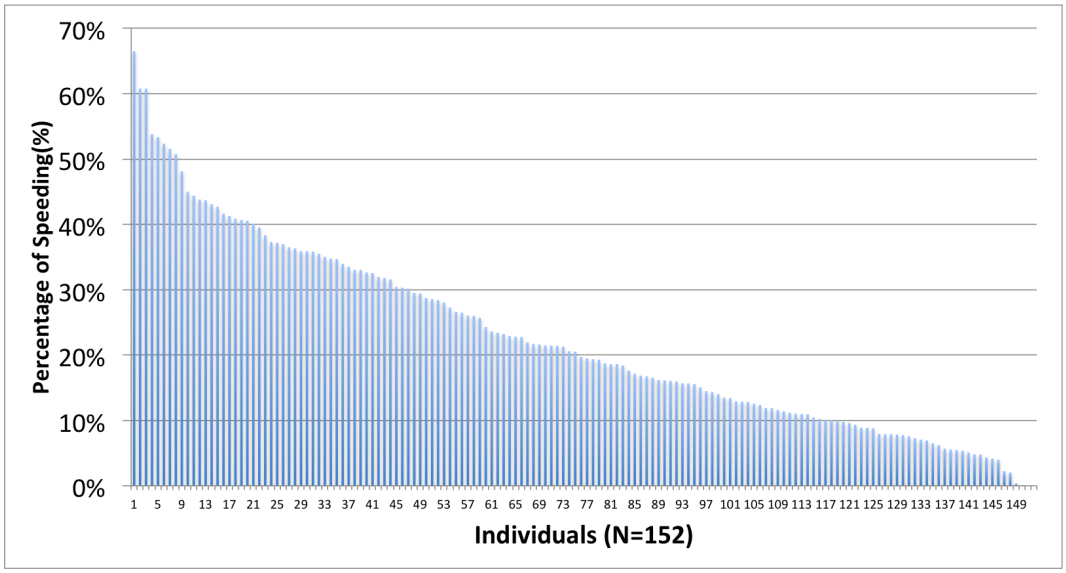



Figure 4.13: Percentage of speeding per individuals

### 4.2.4 Statistical result

Next, we analyze the relationship between several variables (road network structure and personal information) and the speeding statistically. Table 4.3 shows the list of dependent variable and independent variables.

**Dependent variable**

Degree of speeding (Equation 3.1)

**Independent variables**

We change the data into binary (0/1) variable except for "Street length". Moreover, in order to simplify the result, we categorize time into 4 parts; 1. morning (06:00 to 12:00), 2. afternoon (12:00 to 18:00), 3. night (18:00 to 24:00), 4. midnight (24:00 to 06:00). Education is also categorized into 3 parts; 1. education_0 (High school graduate and Vocational/Technical training), 2. education_1 (Some college and Associates degree) and 3. education_2 (Bachelors degree and Graduate/Post-graduate degree).

In order to avoid the dummy variable trap, we drop one category from the model.

- Speed limit: If all indicator is 0, speed limit zone is 70
- Age: If all indicator is 0, age is 18 to 24
- Education: If all indicator is 0, education is High school graduate or Vocational/Technical training [education_0]
- Time: If all indicator is 0, time is 24:00 to 6:00 [midnight]

Table 4.4 showing regression results is calculated using the statistical package *R*. The code of regression analysis in *R* is in A.4.22. While t-value of [age_75_to_84] and [morning] are small, t-value of other variables are large, therefore they are statistically significant. Overall F-test shows that F value is 1.496e+04 and p-value is less than 2.2e-16. Therefore, this model shows high significance. The findings from earlier are corroborated in this multivariate analysis.

Coefficient value indicates that when the street length is long, speeding is more likely to occur. Also, 30 and 35 mph zones are negatively associated with speeding. Persons who are young (18 to 24), male, drive in the afternoon, and/or are educated speed less. Overall the $R^2$ is 0.23.

Table 4.3: List of dependent and independent variables

**Dependent variable**

| Variables | Definition |
|---|---|
| Speeding | Speed data / speed limit zone |

**Independent variables**

| Variables | Definition |
|---|---|
| Street length | Link length |
| speed_25 | Indicator, 1= speed limit zone is less than 25 mph, 0 = otherwise |
| speed_30 | Indicator, 1= speed limit zone is 30 mph, 0 = otherwise |
| speed_35 | Indicator, 1= speed limit zone is 35 mph, 0 = otherwise |
| speed_40 | Indicator, 1= speed limit zone is 40 mph, 0 = otherwise |
| speed_45 | Indicator, 1= speed limit zone is 45 mph, 0 = otherwise |
| speed_50 | Indicator, 1= speed limit zone is 50 mph, 0 = otherwise |
| speed_55 | Indicator, 1= speed limit zone is 55 mph, 0 = otherwise |
| speed_60 | Indicator, 1= speed limit zone is 60 mph, 0 = otherwise |
| speed_65 | Indicator, 1= speed limit zone is 65 mph, 0 = otherwise |
| age_25_to_34 | Indicator, 1= age is 25 to 34, 0 = otherwise |
| age_35_to_44 | Indicator, 1= age is 35 to 44, 0 = otherwise |
| age_45_to_54 | Indicator, 1= age is 45 to 54, 0 = otherwise |
| age_55_to_64 | Indicator, 1= age is 55 to 64, 0 = otherwise |
| age_65_to_74 | Indicator, 1= age is 65 to 74, 0 = otherwise |
| age_75_to_84 | Indicator, 1= age is 75 to 84, 0 = otherwise |
| age_85_and_over | Indicator, 1= age is more than 85, 0 = otherwise |
| male | Indicator, 1= gender is male, 0 = female |
| education_1 | Indicator, 1= education level is Some college or Associates degree, 0 = otherwise |
| education_2 | Indicator, 1= education level is Bachelors degree or Graduate/Post-graduate degree, 0 = otherwise |
| morning | Indicator, 1= time is 06:00 to 12:00, 0 = otherwise |
| afternoon | Indicator, 1= time is 12:00 to 18:00, 0 = otherwise |
| evening | Indicator, 1= time is 18:00 to 24:00, 0 = otherwise |

Table 4.4: Output of statistical analysis in *R*

**Coefficients:**

| | **Estimate** | **t-value** | |
|---|---|---|---|
| (Intercept) | 6.878e-01 | 155.171 | *** |
| streetlength | 1.650e-04 | 173.801 | *** |
| speed_25 | 6.365e-01 | 173.934 | *** |
| speed_30 | -9.581e-02 | -29.015 | *** |
| speed_35 | -1.035e-01 | -32.139 | *** |
| speed_40 | 3.614e-02 | 11.255 | *** |
| speed_45 | 6.506e-02 | 17.665 | *** |
| speed_50 | 4.953e-02 | 13.265 | *** |
| speed_55 | 1.226e-01 | 38.317 | *** |
| speed_60 | 1.166e-01 | 36.527 | *** |
| speed_65 | 7.844e-02 | 22.555 | *** |
| age_25_to_34 | 5.604e-02 | 23.292 | *** |
| age_35_to_44 | 1.221e-02 | 5.462 | *** |
| age_45_to_54 | 3.321e-02 | 14.976 | *** |
| age_55_to_64 | 4.196e-02 | 19.246 | *** |
| age_65_to_74 | 3.868e-02 | 17.403 | *** |
| age_75_to_84 | 3.389e-03 | 1.420 | |
| age_85_and_over | 8.744e-02 | 30.881 | *** |
| male | -1.341e-02 | -23.704 | *** |
| education_1 | -4.808e-03 | -4.906 | *** |
| education_2 | -1.705e-02 | -21.314 | *** |
| morning | 4.021e-03 | 1.796 | . |
| afternoon | -1.322e-02 | -5.934 | *** |
| night | 1.595e-02 | 6.972 | *** |

—

signif. Codes: 0 '***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | |
|---|---|
| Multiple R-squared | 0.2269 |
| F-statistic | 1.496e+04 |
| p-value | <2.2e-16 |

### 4.2.5 Fixed effects estimation with integrated GPS points

While analytical results of this paper are from all GPS points, we also analyze the results from integrated GPS points. GPS points were recorded every second; however, it is considered that speeding patterns of drivers in the same link are similar. Therefore, we integrate multiple GPS points from one link into one analysis record; however, we separate the link data depending on drivers and trips because driving environments and driving patterns are different. The total amount of GPS points is 115,536 (152 drivers and 24,262 link). The code of this algorithm is in A.4.18.

Moreover, in order to eliminate the fixed effect prior to estimation, we use the dummy variable regression. We put in a dummy variable for each individual along with the independent variables. While the t-value of [speed_40], [speed_45], [speed_50], [age_55_to_64], [age_85_and_over] and [male] are small, the t-value of other variables are large. Overall F-test shows that F value is 280.5 and p-value is less than 2.2e-16. Therefore, this model shows high significance. Overall the $R^2$ is 0.29.

A positive coefficient value indicates that when the street length is long, speeding is likely to occur. Also, 30 and 35 mph zones are negatively associated with speeding. Persons who are young, drive at midnight (24:00 to 6:00), and/or are educated speed more (Table 4.5). Compared with the result of Table 4.4, the result of road network structure is similar, but the result of personal information is different. The code of regression result in $R$ is mentioned in A.4.23.

Figure 4.14 illustrates the estimate of coefficients across drivers, and the result shows that there are significant differences depending on the driver. 62.4% of individuals are negative and some individuals' estimates are significantly large.

Table 4.5: Output of the dummy variable regression in $R$

**Coefficients:**

| | **Estimate** | **t-value** | |
|---|---|---|---|
| (Intercept) | 8.386e-01 | 35.082 | *** |
| streetlength | 1.280e-04 | 34.736 | *** |
| speed_25 | 6.587e-01 | 53.206 | *** |
| speed_30 | -3.703e-02 | -3.125 | ** |
| speed_35 | -1.177e-01 | -10.162 | *** |
| speed_40 | -8.413e-03 | -0.727 | |
| speed_45 | -3.285e-03 | -0.255 | |
| speed_50 | 8.432e-03 | 0.636 | |
| speed_55 | 1.056e-01 | 9.164 | *** |
| speed_60 | 9.919e-02 | 8.538 | *** |
| speed_65 | 3.506e-02 | 2.823 | ** |
| age_25_to_34 | 7.488e-02 | 4.963 | *** |
| age_35_to_44 | -1.055e-01 | -3.933 | *** |
| age_45_to_54 | -2.184e-01 | -8.490 | *** |
| age_55_to_64 | -3.575e-02 | -1.513 | |
| age_65_to_74 | -1.572e-01 | -7.345 | *** |
| age_75_to_84 | -5.696e-02 | -3.927 | *** |
| age_85_and_over | -3.847e-02 | -1.474 | |
| male | -1.026e-02 | -0.580 | |
| education_1 | 1.520e-01 | 9.020 | *** |
| education_2 | 1.347e-01 | 4.676 | *** |
| morning | -3.225e-02 | -4.726 | *** |
| afternoon | -3.566e-02 | -5.254 | *** |
| night | -2.295e-02 | -3.285 | ** |
| person1 | 8.224e-02 | 5.936 | *** |
| person2 | -3.585e-01 | -8.735 | *** |
| — person 151 | | | |

—

Signif. codes:,0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

| | |
|---|---|
| Multiple R-squared | 0.2851 |
| F-statistic | 280.5 |
| p-value | <2.2e-16 |

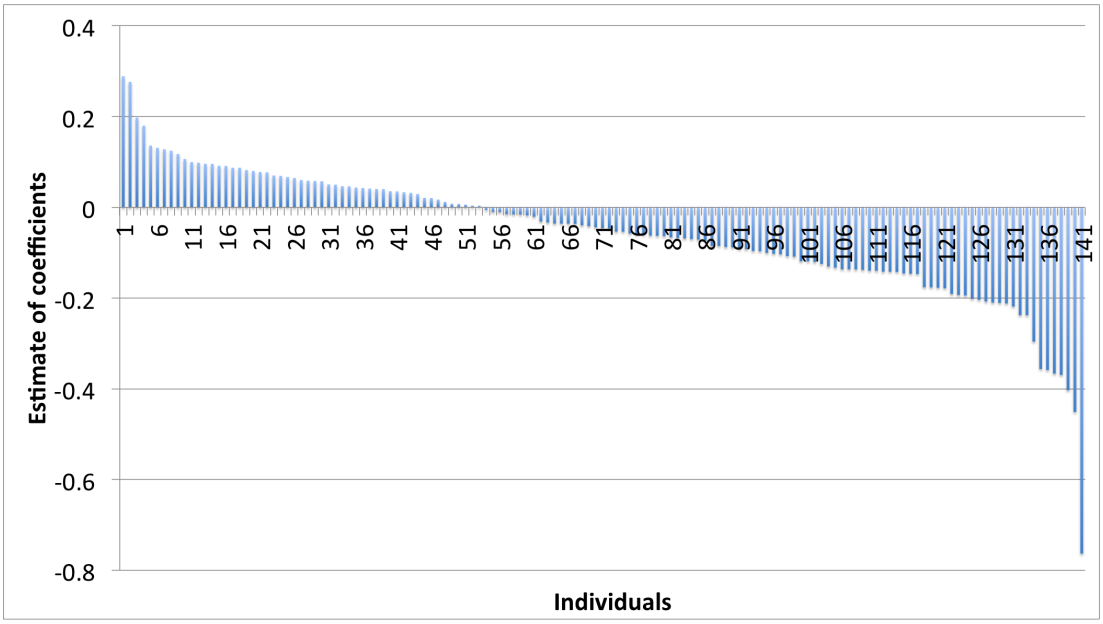



Figure 4.14: Estimate of coefficients per individuals

# Chapter 5

# Conclusion and Discussion

## 5.1 Conclusion

This thesis investigates GPS-based speed data from 152 participants in Minnesota to examine the relationship between road network structure and speeding. The most pertinent findings from the results are that speeding behavior is significant in both low and high speed limit zones, and long link length is correlated with speeding. In terms of degree of speeding, the t-test indicates that there are significant differences across gender, age, and education level. Moreover, our algorithm and process shows sufficient accuracy of map-matching with very few false positives, although half of GPS data are removed due to the process. We analyzed 1,172,103 GPS points.

Unlike the speed data of a loop detector, GPS-based analysis enables us to investigate the speed information over a broad area. Multi-variate regression analysis finds persons who are young (18 to 24), male, drive in the afternoon, and/or are educated speed less than other age groups, female drivers, other times of the day, and/or less educated drivers. Future research will aim to reduce the amount of excluded data (false negatives).

## 5.2 Discussion

- Comparison with other studies
  Speeding behavior across speed limit zones has been previously analyzed [3], and according to both that result and our result, speeding illustrates a 'V-shape' across

all speed limit zones. In the previous study, the 60 mph zone had the lowest speeding behavior [3] while our result shows the 30 mph zone is the lowest. It is assumed that the difference in road characteristics might affect the result.

Other studies mention that elderly, female and uneducated drivers who don't often drive on interurban roads are likely to follow the speed limit [1, 3]. In our study, young (18 to 24), male, drive in the afternoon, and/or are educated drivers showed less speeding behavior than comparison drivers. To confirm our finding, we need to see this difference in detail in further studies.

- Over-fitting problem

  We perform statistical analysis from several independent variables. But it is considered the model might be affected by over-fitting problems (Table 4.4). When the independent variables are only speed limit zones (Table A.1), the $R^2$ is 0.20 and it is close to the value of Table 4.4 (the $R^2$ is 0.23). On the other hand, $R^2$ is small when other factors are independent variables (Table A.2 to Table A.6). Therefore, the speed limit zone has a more significant impact on speeding compared with other factors.

- Data

  Because of the limited data, we cannot analyze the relationship between speeding behavior and some road characteristics like road shape (curvature) and lane width. Also we cannot analyze the effect of vehicle size.

  If we add this information to GPS data, we would analyze more interesting effects of speeding. For example, if we add crash data into GPS data, we can analyze whether speeding is related to crashes.

  We analyze 152 drivers; however, the amount of data differs depending on the driver. As Figure 4.13 shows, speed limit compliance depends on the driver.

- Algorithm and process

  We investigate GPS data of each participants for 7 days; however, 51.0% of GPS data are removed in our algorithm in order to secure the accuracy of the map-matching. Many GPS points near intersections and interstates are deleted due to our algorithm.

If drivers show significant speeding behavior in the excluded GPS data, analysis results are likely to be biased. In order to avoid this bias, we need to decrease the amount of excluded GPS data. For example, an algorithm that increases the accuracy of map-matching near intersections and interstates has been proposed. This algorithm considers neighboring GPS points for improving the accuracy of map-matching near the intersection [15]. The algorithm should consider the average speed of GPS data by distinguishing between nearby parallel routes.

Some actual driving links are also deleted due to the process and it makes it difficult to find the whole trip data. A solution is that if the street name and speed limit in front and behind the missing link is identical, then data of missing link is also identical to adjacent link (Figure 5.1). Further use of upstream and downstream points to identify the most likely link for a point in overlapping layers is also a promising direction.

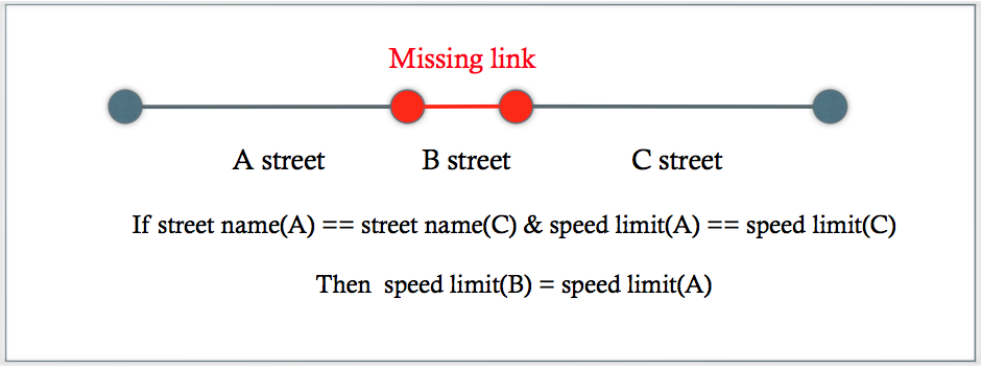



Figure 5.1: Image of complementing missing link

If we keep both whole driving GPS data and satisfactory accuracy, we would be able to analyze the effect of other road network structures:

1. Discontinuity of speed limit zones – *Hypothesis: Large discontinuity is correlated with speeding*

   This network variable means the effect of discontinuity (speed limit distribution) during the trip. When the speed limit is uniform during the trip, it is predicted that the amount of acceleration and deceleration is small. This situation would create stable driving during the trip.

2. Geographical feature (Curve or straight?) – *Hypothesis: Straight roads are correlated with speeding*

   The driver is significantly affected by geographical features (network shape). When the driver drives on a curved section and changes direction, it would affect speeding. GPS data has course (degrees) data, so we can calculate the amount of degree change.

- Differences of speeding across speed limit zones

  Figure 4.1 and Figure 4.2 illustrate that speeding behavior is significant in low speed limit areas more than high speed limit areas. However, traffic enforcement against speeding isn't usually conducted in low speed limit zones. Speed traps or speed cameras are generally installed in high speed limit zones (to be clear, speed cameras are not used in Minnesota). Speeding behavior on local roads is obviously dangerous for the pedestrian and it risks vehicle-to-pedestrian collisions. On the other hand, speeding behavior is relatively small near 30 and 35mph zones. These 35mph zones are most prevalent in Minneapolis and the road geometry and traffic condition in this zone fits the driver's preferred speed for the physical conditions by the road.

# References


[1] George Kanellaidis, John Golias, and Kimonas Zarifopoulos. A survey of drivers' attitudes toward speed limit violations. *Journal of Safety Research*, 26(1):31–40, 1995.

[2] Ashim Kumar Debnath, Ross Blackman, and Narelle Haworth. A Tobit model for analyzing speed limit compliance in work zones. *Safety Science*, 70:367–377, 2014.

[3] Frank Lai and Oliver Carsten. What benefit does Intelligent Speed Adaptation deliver: a close examination of its effect on vehicle speeds. *Accident; analysis and prevention*, 48:4–9, 2012.

[4] Minnesota Department of Transportation. Speed limits in minnesota. `http://www.dot.state.mn.us/speed/`, 2015. Accessed Nov. 23, 2015.

[5] House Research Department. Minnesota speed limits. `http://www.house.leg.state.mn.us/hrd/pubs/ss/ssspdlt.pdf`, 2015. Accessed Nov. 23, 2015.

[6] Mark Rask. *American Autobahn*. Vanguard Non-Fiction Books, 1999.

[7] COPENHAGENIZE.COM. The 85th percentile folly. `http://www.copenhagenize.com/2012/11/the-85th-percentile-folly.html`, 2012. Accessed Dec. 15, 2015.

[8] George Kanellaidis. Factors affecting drivers' choice of speed on roadway curves. *Journal of Safety Research*, 26(1):49–56, 1995.

[9] Ahmad Abdullah Saifizul, Hideo Yamanaka, and Mohamed Rehan Karim. Empirical analysis of gross vehicle weight and free flow speed and consideration on its relation with differential speed limit. *Accident; analysis and prevention*, 43(3):1068–73, 2011.

[10] David Levinson. Network Structure and City Size. *PLoS ONE*, 7(1):e29721, 2012.

[11] Andrew D. Townshend, Charles J. Worringham, and Ian B. Stewart. Assessment of speed and position during human locomotion using nondifferential GPS. *Medicine and Science in Sports and Exercise*, 40(1):124–132, 2008.

[12] Juan C Herrera, Daniel B Work, Ryan Herring, Xuegang Ban, Quinn Jacobson, and Alexandre M Bayen. Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment. *Transportation Research Part C*, 18:568–583, 2010.

[13] Yin Wang, Yanmin Zhu, Zhaocheng He, Yang Yue, and Qingquan Li. Challenges and opportunities in exploiting large-scale gps probe data. *HP Laboratories, Technical Report HPL-2011-109*, 21, 2011.

[14] T. H. Witte and A. M. Wilson. Accuracy of WAAS-enabled GPS for the determination of position and speed over ground. *Journal of Biomechanics*, 38:1717–1722, 2005.

[15] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate GPS trajectories. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09*, (c):352–361, 2009.

[16] Fernando Torre, David Pitchford, Phil Brown, and Loren Terveen. Matching GPS Traces to ( Possibly ) Incomplete Map Data : Bridging Map Building and Map Matching. *SIGSPATIAL '12 Proc. 20th Int. Conf. Adv. Geogr. Inf. Syst.*, pages 546–549, 2012.

[17] F. Marchal, J. Hackney, and K. Axhausen. Efficient Map Matching of Large Global Positioning System Data Sets: Tests on Speed-Monitoring Experiment in Zürich. *Transportation Research Record*, 1935(1):93–100, 2005.

[18] Metropolitan Council. Travel behavior inventory survey. http://metrocouncil.org/Transportation/Planning-2/Transit-Plans,-Studies-Reports/Other-Studies-Reports/Travel-Behavior-Inventory/2010-Travel-Behavior-Inventory-(TBI)-Survey-Data-R.aspx. Accessed Nov. 24, 2015.

[19] Jonathan Ehrlich. Travel behavior inventory (tbi) 2010 household interview survey. https://gisdata.mn.gov/dataset/us-mn-state-metc-society-tbi-home-interview2010, 2013. Accessed Nov. 23, 2015.

[20] Tim Sutton and Otto Dassau. Qgis. http://www.qgis.org/en/site/index.html, 2015. Accessed Nov. 24, 2015.

[21] Wenyun Tang, David Levinson, and Lin Cheng. An empirical study of the deviation between actual and shortest travel time paths. http://nexus.umn.edu/papers/PeopleAreNotRational.pdf, 2014. Accessed Nov. 24, 2015.

[22] MetroGIS. Datafinder catalog. http://www.datafinder.org/catalog/index.asp, 2015. Accessed Nov. 24, 2015.

[23] Jon Hoekenga. Transit stops. http://www.datafinder.org/metadata/TransitStops.html, 2012. Accessed Nov. 24, 2015.

[24] Rachel Wiken. Transitway stations. http://www.datafinder.org/metadata/TransitwayStations.html, 2014. Accessed Nov. 24, 2015.

[25] Metro Transit. Northstar commuter rail line. http://www.metrotransit.org/northstar. Accessed Nov. 24, 2015.

[26] THE NORTHSTAR CORRIDOR DEVELOPMENT AUTHORITY. Northstar corridor - about us. http://mn-getonboard.com/aboutus.html. Accessed Nov. 24, 2015.

[27] Metro Transit. Metro blue line. http://www.metrotransit.org/metro-blue-line. Accessed Nov. 24, 2015.

[28] Metro Transit. Metro green line. http://www.metrotransit.org/metro-green-line. Accessed Nov. 24, 2015.

[29] Metro Transit. Metro red line. http://www.metrotransit.org/metro-red-line. Accessed Nov. 24, 2015.

[30] Charles F F Karney. Transverse Mercator with an accuracy of a few nanometers. *Journal of Geodesy*, 85(8):475–485, 2011, arXiv:1002.1417.

[31] Kazushige KAWASE. Concise Derivation of Extensive Coordinate Conversion Formulae in the Gauss-Krüger Projection. *Bulletin of the Geospatial Information Authority of Japan*, 60(section 2):1–6, 2012.

[32] Washington Yotto Ochieng, Mohammed A. Quddus, and Robert B. Noland. Map-matching in complex urban road networks. *Brazilian Journal of Cartography*, 55(2):1–14, 2003.

[33] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[34] timeanddate.com. Time zone in minneapolis, minnesota, u.s.a. http://www.timeanddate.com/time/zone/usa/minneapolis, 2015. Accessed Nov. 24, 2015.

[35] United Nations Office for Outer Space Affairs. World geodetic system 1984. http://www.unoosa.org/pdf/icg/2012/template/WGS_84.pdf. Accessed Nov. 24, 2015.

[36] MN.IT Services. Coordinate specifications for spatial data exchange between minnesota state agencies. http://mn.gov/mnit/programs/policies/geospatial/gis-pages/spatial-data-exchange.jsp, 1998. Accessed Nov. 24, 2015.

# Appendix A

# Glossary and Acronyms

## A.1 Formula

### A.1.1 Convert coordinate system

Convert the unit of Longitude and Latitude from degree to radian

$$\lambda = Longitude[radian] = Longitude[degree] * \pi/180 \tag{A.1}$$

$$\varphi = Latitude[radian] = Latitude[degree] * \pi/180 \tag{A.2}$$

f means flattening factor of the earth in WGS84 [35]

$$f = 1/298.257223563 \tag{A.3}$$

$n$ means third flattening

$$n = \frac{f}{2-f} \tag{A.4}$$

$a$ means semi-major axis (Unit: m) [35]

$$a = 6,378,137 \tag{A.5}$$

Following formula is from [30]

$$A = \frac{a}{1+n}(1 + \frac{n^2}{4} + \frac{n^4}{64}) \tag{A.6}$$

$$\alpha_1 = \frac{1}{2}n - \frac{2}{3}n^2 + \frac{5}{16}n^3 \tag{A.7}$$

$$\alpha_2 = \frac{13}{48}n^2 - \frac{3}{5}n^3 \tag{A.8}$$

$$\alpha_3 = \frac{61}{240}n^3 \tag{A.9}$$

Following formula is from [31]

$$\tan\chi = \sinh(\tanh^{-1}\sin\varphi - \frac{2\sqrt{n}}{1+n}\tanh^{-1}(\frac{2\sqrt{n}}{1+n}\sin\varphi)) \tag{A.10}$$

UTM zone in Minnesota is 15 [36]

$$\lambda_0'[degree] = 15 * 6 - 183 \tag{A.11}$$

Convert the unit of Longitude and Latitude from degree to radian

$$\lambda_0[radian] = \lambda_0' * \pi/180 \tag{A.12}$$

Following formula is from [31]

$$\xi = \tanh^{-1}(\tan\chi/\cos(\lambda - \lambda_0)) \tag{A.13}$$

$$\eta = \tanh^{-1}(\frac{\sin(\lambda - \lambda_0)}{\sqrt{1+\tan\chi^2}}) \tag{A.14}$$

$E_0$ means false easting (Unit: m) [36]

$$E_0 = 500,000 \tag{A.15}$$

$m_0$ means scale factor [36]

$$m_0 = 0.9996 \tag{A.16}$$

Plane rectangular coordinates easting X and northing Y (Coordinate system: UTM) [31]

$$X = X0 + m_0 A(\eta + \sum_{k=1}^{3}\alpha_k\cos(2k\xi)\sinh(2k\eta)) \tag{A.17}$$

$$Y = m_0 A(\xi + \sum_{k=1}^{3}\alpha_k\sin(2k\xi)\cosh(2k\eta)) \tag{A.18}$$

## A.2 Process

### A.2.1 Create buffer of GIS links

It is noted that we don't have the algorithm about this process.

1-1. Put GIS map on QGIS

- Drag .shp file to QGIS screen (Figure A.1)

1-2. Create a buffer

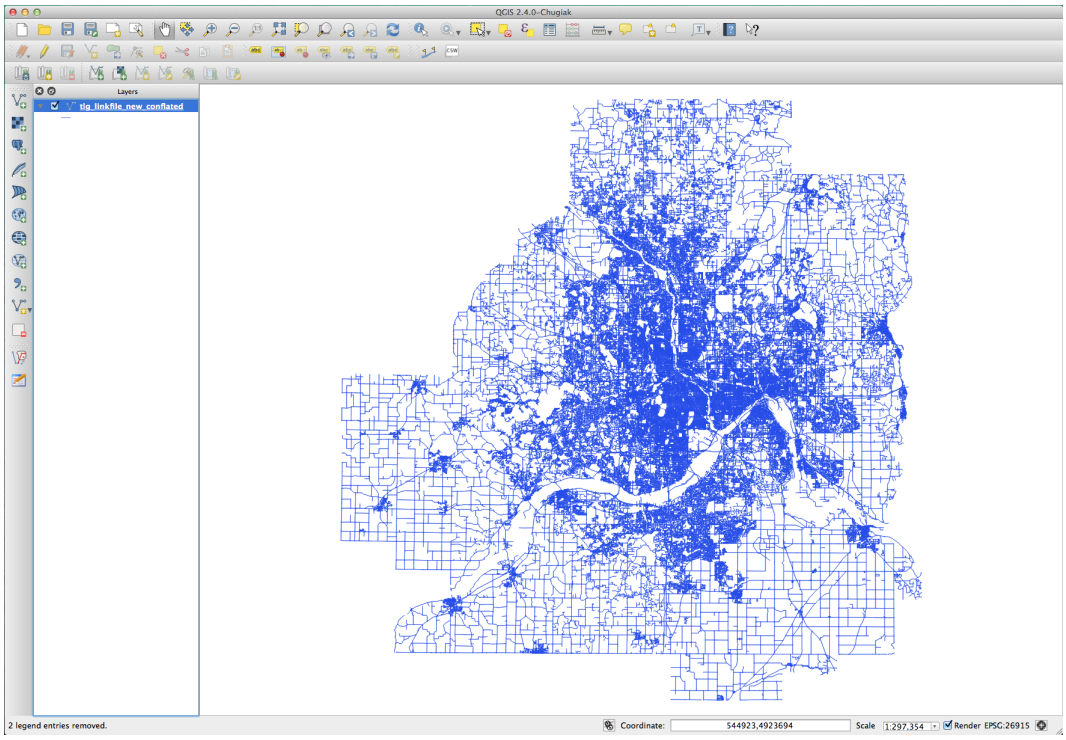



Figure A.1: GIS map

- Choose a "Buffer(s)" function

Vector ->Geoprocessing tools ->Buffer(s) (Figure A.2)

- Create 10m buffer(s)

Input "10" as 'Buffer distance'. It is noted that the unit of buffer distance is meters.

We choose "Input vector layer" as GIS map data. (Figure A.3)

- Result of output (10m buffer)

As Figure A.5 shows, GIS links change from line to polygon.

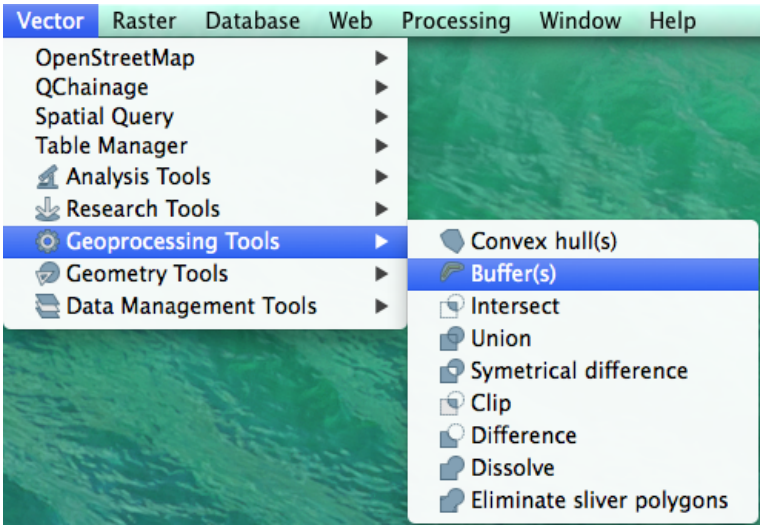



Figure A.2: Buffer process 1

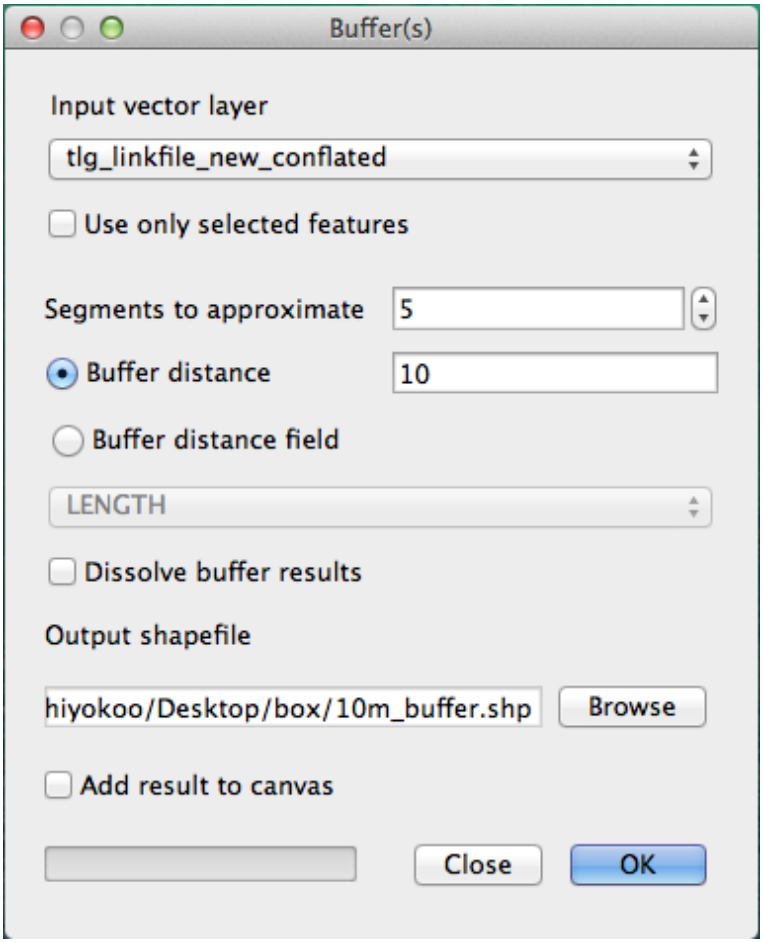



Figure A.3: Buffer process 2

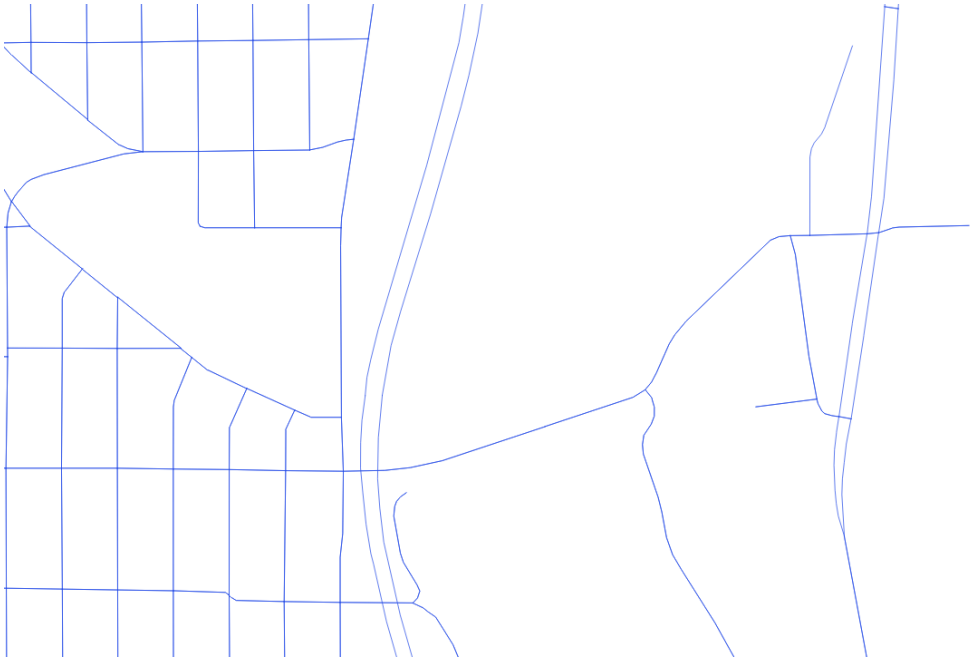


Figure A.4: GIS links

Figure A.5: GIS buffer(s) (10m)

### A.2.2 Extract GIS links that are close to GPS data

2.1. Load a GPS Layer from a Delimited Text File
Layer ->Add Delimited Text Layer (Figure A.6)

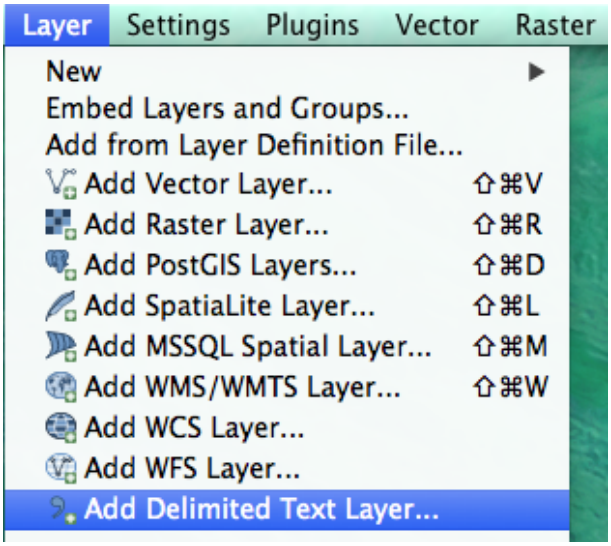
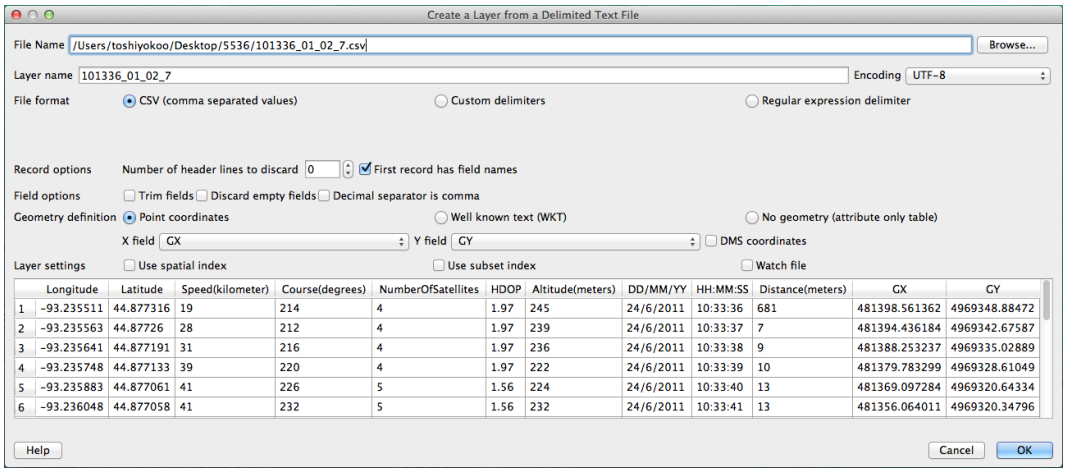



Figure A.6: Add Delimited Text Layer

Input the GPS file and set X field and Y field as the location data of UTM coordinates. (Figure A.7)
File format: CSV file
Geometry definition: Point coordinates
X field: X (Equation A.17), Y field: Y (Equation A.18) [X and Y are UTM coordinates]

2.2. Change coordinate system
Choose a GPS layer and click properties button
General ->Coordinate reference system
Press “Specify” button
Coordinate Reference System: “NAD83 / UTM zone 15N”, Authority ID: “ESPG: 26915” (Figure A.8)
- Result of output
Figure A.9 is the result of the process




| | Longitude | Latitude | Speed(kilometer) | Course(degrees) | NumberOfSatellites | HDOP | Altitude(meters) | DD/MM/YY | HH:MM:SS | Distance(meters) | GX | GY |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -93.235511 | 44.877316 | 19 | 214 | 4 | 1.97 | 245 | 24/6/2011 | 10:33:36 | 681 | 481398.561362 | 4969348.88472 |
| 2 | -93.235563 | 44.87726 | 28 | 212 | 4 | 1.97 | 239 | 24/6/2011 | 10:33:37 | 7 | 481394.436184 | 4969342.67587 |
| 3 | -93.235641 | 44.877191 | 31 | 216 | 4 | 1.97 | 236 | 24/6/2011 | 10:33:38 | 9 | 481388.253237 | 4969335.02889 |
| 4 | -93.235748 | 44.877133 | 39 | 220 | 4 | 1.97 | 222 | 24/6/2011 | 10:33:39 | 10 | 481379.783299 | 4969328.61049 |
| 5 | -93.235883 | 44.877061 | 41 | 226 | 5 | 1.56 | 224 | 24/6/2011 | 10:33:40 | 13 | 481369.097284 | 4969320.64334 |
| 6 | -93.236048 | 44.877058 | 41 | 232 | 5 | 1.56 | 232 | 24/6/2011 | 10:33:41 | 13 | 481356.064011 | 4969320.34796 |

Figure A.7: Create a Layer from a Delimited Text File

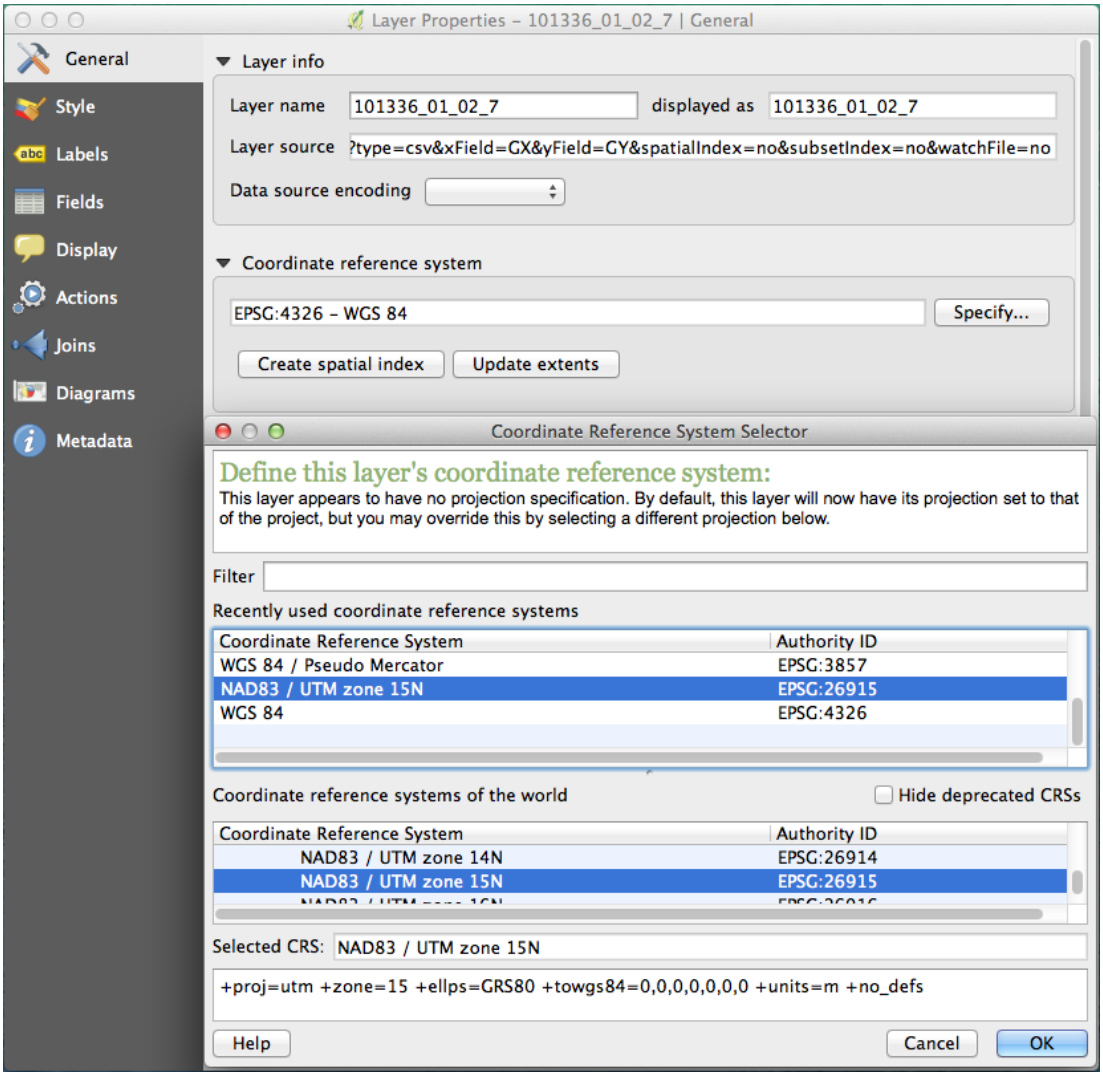



Figure A.8: Coordinate Reference System Selector

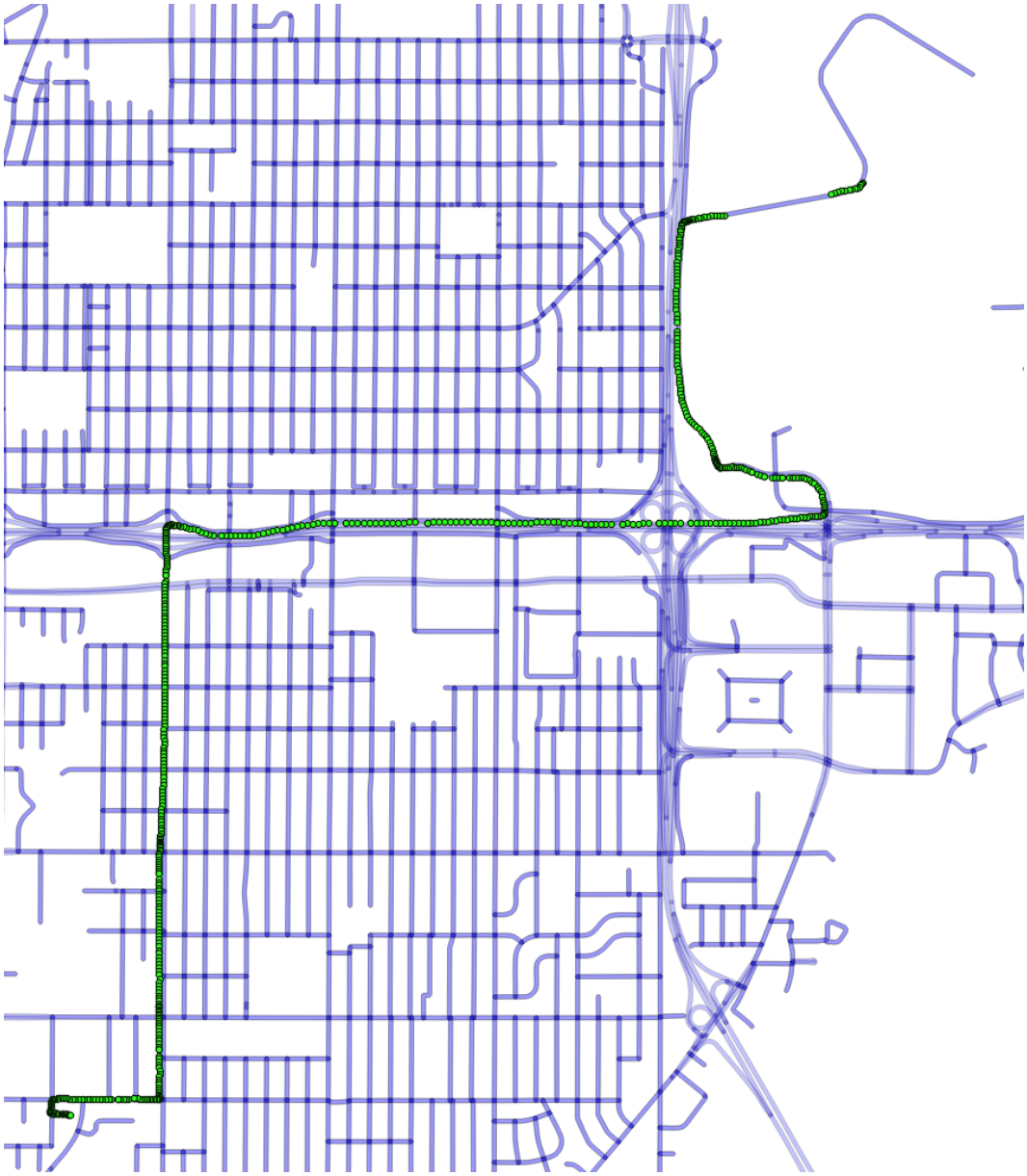


Figure A.9: Result of output
(Blue lines: 10m GIS buffer, Green dots: GPS data)

2.3. Extract GIS links

- Choose a “Select by location” function

Vector ->Research tools ->Select by location (Figure A.10)

Select “Select features in:” as GIS buffer data and “that intersect features in:” as GPS data. (Figure A.11)

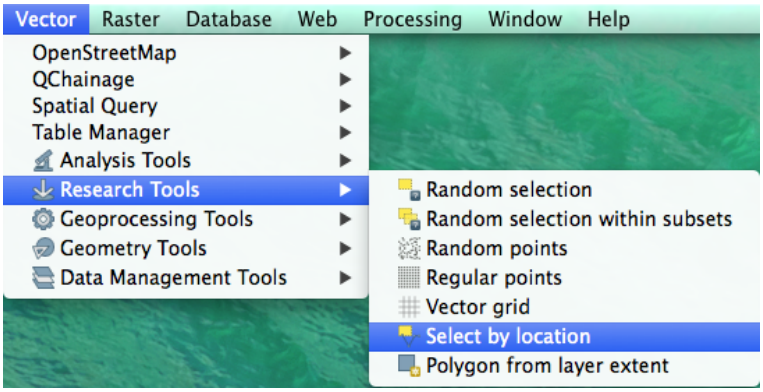



Figure A.10: Select by location 1

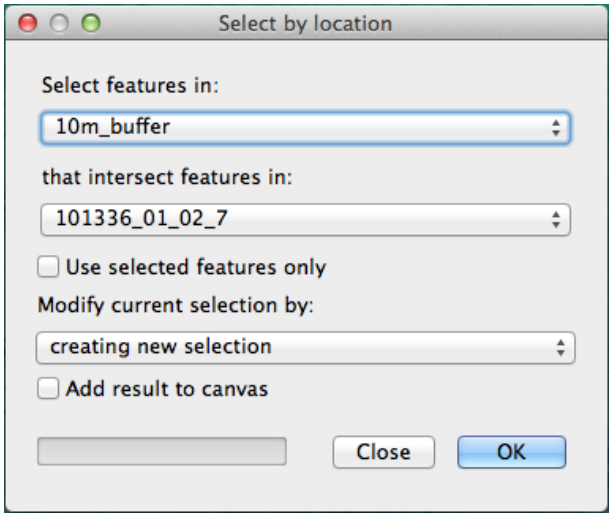



Figure A.11: Select by location 2

## A.3 Result

### A.3.1 Output of speeding analysis in *R*

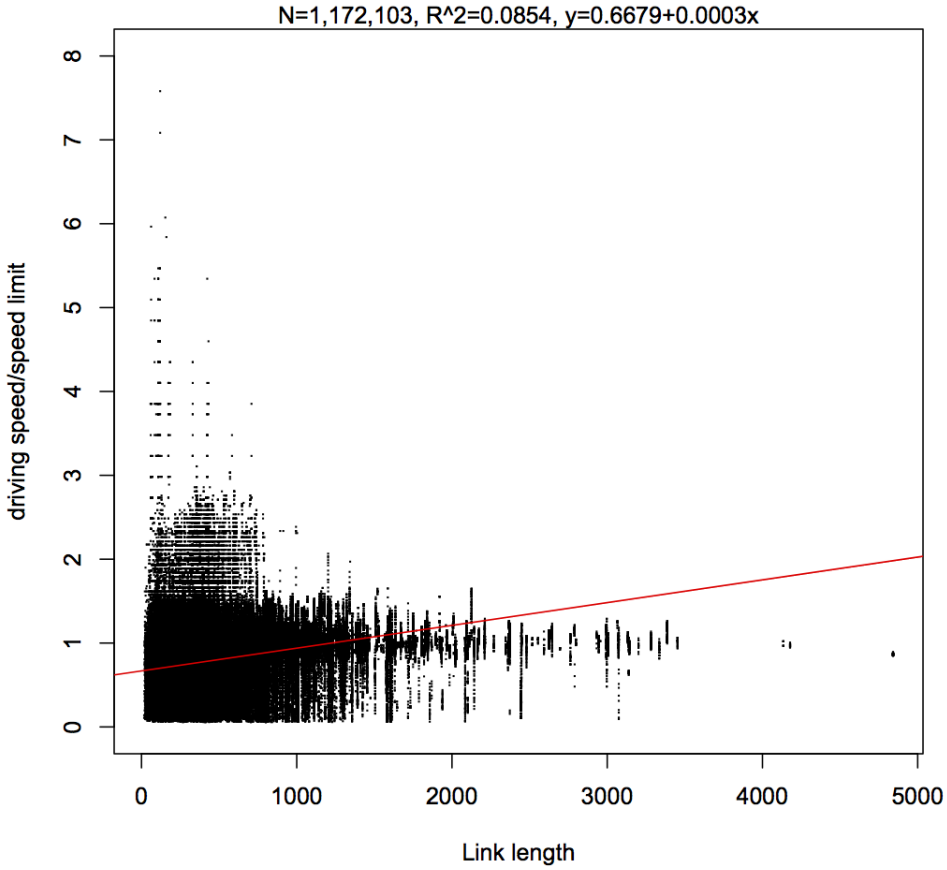



Figure A.12: Relationship between link length and speeding

### A.3.2 Output of statistical analysis in *R*

Table A.1: Independent variable: Speed limit zones

**Coefficients:**

| | **Estimate** | **t-value** | |
|---|---|---|---|
| (Intercept) | 0.939 | 310.909 | *** |
| speed_25 | 0.460 | 128.836 | *** |
| speed_30 | -0.307 | -98.108 | *** |
| speed_35 | -0.302 | -98.792 | *** |
| speed_40 | -0.154 | -50.316 | *** |
| speed_45 | -0.109 | -30.307 | *** |
| speed_50 | -0.104 | -28.253 | *** |
| speed_55 | -0.013 | -4.137 | *** |
| speed_60 | 0.008 | 2.552 | * |
| speed_65 | -0.007 | -1.912 | . |

—

signif. Codes: 0 '***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | |
|---|---|
| Multiple R-squared | 0.2021 |
| F-statistic | 3.299e+04 |
| p-value | <2.2e-16 |

Table A.2: Independent variable: Link length

**Coefficients:**

| | **Estimate** | **t-value** | |
|---|---|---|---|
| (Intercept) | 6.679e-01 | 1604.8 | *** |
| streetlength | 2.716e-04 | 330.7 | *** |

—

signif. Codes: 0 '***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | |
|---|---|
| Multiple R-squared | 0.0854 |
| F-statistic | 1.094e+05 |
| p-value | <2.2e-16 |

Table A.3: Independent variable: Age group

**Coefficients:**

| | Estimate | t-value | |
|---|---|---|---|
| (Intercept) | 0.703 | 308.161 | *** |
| age_25_to_34 | 0.117 | 45.662 | *** |
| age_35_to_44 | 0.039 | 16.219 | *** |
| age_45_to_54 | 0.050 | 20.862 | *** |
| age_55_to_64 | 0.083 | 35.254 | *** |
| age_65_to_74 | 0.061 | 25.509 | *** |
| age_75_to_84 | 0.013 | 5.093 | *** |
| age_85_and_over | 0.139 | 44.868 | *** |

—

signif. Codes: 0,'***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | |
|---|---|
| Multiple R-squared | 0.00717 |
| F-statistic | 1210 |
| p-value | <2.2e-16 |

Table A.4: Independent variable: Gender

**Coefficients:**

| | Estimate | t-value | |
|---|---|---|---|
| (Intercept) | 0.775 | 1921.23 | *** |
| male | -0.021 | -33.843 | *** |

—

signif. Codes: 0,'***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | |
|---|---|
| Multiple R-squared | 0.00098 |
| F-statistic | 1145 |
| p-value | <2.2e-16 |

Table A.5: Independent variable: Education

**Coefficients:**

| | **Estimate** | **t-value** | |
|---|---|---|---|
| (Intercept) | 0.777 | 1010.81 | *** |
| education_1 | -0.010 | -9.861 | *** |
| education_2 | -0.013 | -15.479 | *** |

—

signif. Codes: 0,'***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

| | |
|---|---|
| Multiple R-squared | 0.00020 |
| F-statistic | 119.9 |
| p-value | <2.2e-16 |

Table A.6: Independent variable: Time

**Coefficients:**

| | **Estimate** | **t-value** | |
|---|---|---|---|
| (Intercept) | 0.849 | 343.761 | *** |
| morning | -0.079 | -31.217 | *** |
| afternoon | -0.098 | -39.011 | *** |
| night | -0.057 | -22.125 | *** |

—

signif. Codes: 0 '***' 0.001,'**',0.01 '*' 0.05 '.' 0.1 ' ' 1

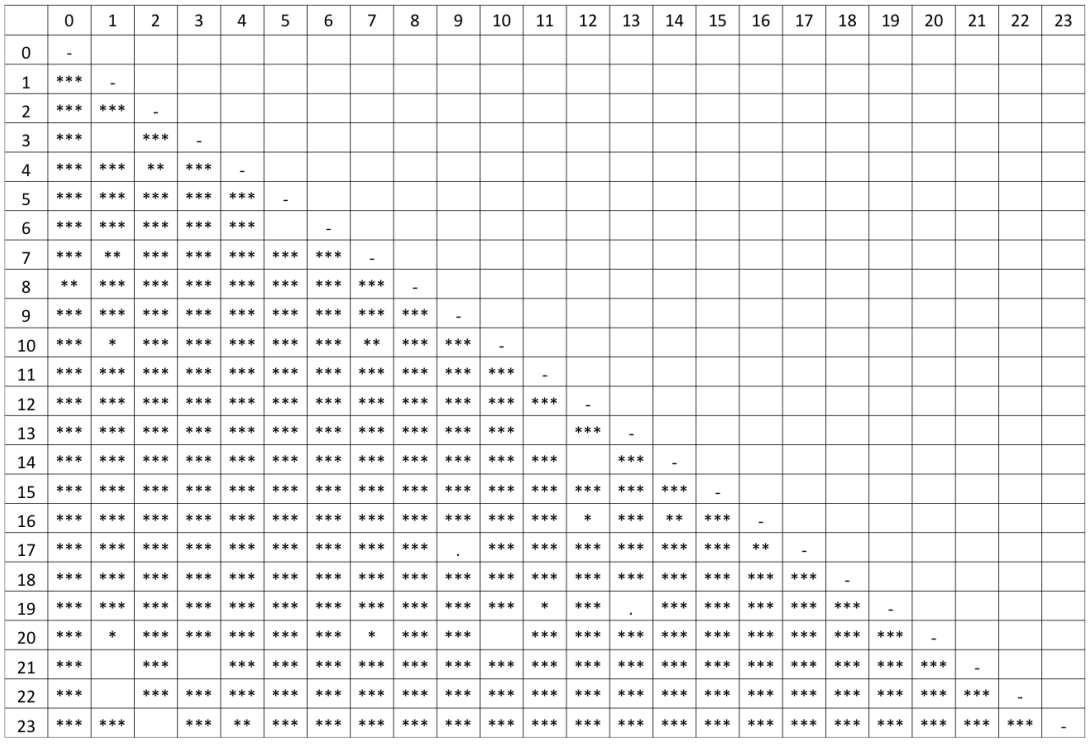
| | |
|---|---|
| Multiple R-squared | 0.0030 |
| F-statistic | 1186 |
| p-value | <2.2e-16 |

Table A.7: Mean value: Degree of speeding across time

| **Hour** | **Mean** |
|---|---|
| 0 | 0.709 |
| 1 | 0.816 |
| 2 | 0.913 |
| 3 | 0.831 |
| 4 | 0.947 |
| 5 | 0.870 |
| 6 | 0.870 |
| 7 | 0.784 |
| 8 | 0.727 |
| 9 | 0.744 |
| 10 | 0.789 |
| 11 | 0.766 |
| 12 | 0.754 |
| 13 | 0.766 |
| 14 | 0.756 |
| 15 | 0.736 |
| 16 | 0.751 |
| 17 | 0.747 |
| 18 | 0.777 |
| 19 | 0.770 |
| 20 | 0.789 |
| 21 | 0.824 |
| 22 | 0.810 |
| 23 | 0.925 |

Table A.8: Mean value: Degree of speeding across gender

| **Gender** | **Mean** |
|---|---|
| female | 0.775 |
| male | 0.754 |



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | - | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | *** | - | | | | | | | | | | | | | | | | | | | | | | |
| 2 | *** | *** | - | | | | | | | | | | | | | | | | | | | | | |
| 3 | *** | | *** | - | | | | | | | | | | | | | | | | | | | | |
| 4 | *** | *** | ** | *** | - | | | | | | | | | | | | | | | | | | | |
| 5 | *** | *** | *** | *** | *** | - | | | | | | | | | | | | | | | | | | |
| 6 | *** | *** | *** | *** | *** | | - | | | | | | | | | | | | | | | | | |
| 7 | *** | ** | *** | *** | *** | *** | *** | - | | | | | | | | | | | | | | | | |
| 8 | ** | *** | *** | *** | *** | *** | *** | *** | - | | | | | | | | | | | | | | | |
| 9 | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | | | | | | | | | | | | | |
| 10 | *** | * | *** | *** | *** | *** | *** | ** | *** | *** | - | | | | | | | | | | | | | |
| 11 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | | | | | | | | | | | |
| 12 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | | | | | | | | | | |
| 13 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | | *** | - | | | | | | | | | | |
| 14 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | | *** | - | | | | | | | | | |
| 15 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | | | | | | | |
| 16 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | * | *** | ** | *** | - | | | | | | | |
| 17 | *** | *** | *** | *** | *** | *** | *** | *** | *** | . | *** | *** | *** | *** | *** | *** | ** | - | | | | | | |
| 18 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | | | | |
| 19 | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | * | *** | . | *** | *** | *** | *** | *** | - | | | | |
| 20 | *** | * | *** | *** | *** | *** | *** | * | *** | *** | | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | | |
| 21 | *** | | *** | | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | | |
| 22 | *** | | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - | |
| 23 | *** | *** | | *** | ** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | *** | - |

Figure A.13: Significant code: Degree of speeding across time

Table A.9: Mean value: Degree of speeding across age group

| **Age** | **Mean** |
|---|---|
| 5: 18 to 24 | 0.703 |
| 6: 25 to 34 | 0.820 |
| 7: 35 to 44 | 0.742 |
| 8: 45 to 54 | 0.753 |
| 9: 55 to 64 | 0.786 |
| 10: 65 to 74 | 0.764 |
| 11: 75 to 84 | 0.716 |
| 12: 85 and over | 0.842 |

| | 18 to 24 | 25 to 34 | 35 to 44 | 45 to 54 | 55 to 64 | 65 to 74 | 75 to 84 | 85 and over |
|---|---|---|---|---|---|---|---|---|
| 18 to 24 | - | | | | | | | |
| 25 to 34 | *** | - | | | | | | |
| 35 to 44 | *** | *** | - | | | | | |
| 45 to 54 | *** | *** | *** | - | | | | |
| 55 to 64 | *** | *** | *** | *** | - | | | |
| 65 to 74 | *** | *** | *** | *** | *** | - | | |
| 75 to 84 | *** | *** | *** | *** | *** | *** | - | |
| 85 and over | *** | *** | *** | *** | *** | *** | *** | - |

Figure A.14: Significant code: Degree of speeding across age group

Table A.10: Mean value: Degree of speeding across education level

| **Education** | **Mean** |
|---|---|
| Associates degree | 0.789 |
| Bachelors degree | 0.774 |
| Graduate/Post-graduate degree | 0.750 |
| High school graduate | 0.768 |
| Some college | 0.760 |
| Vocational/Technical training | 0.806 |

| | Associates degree | Some college | High school graduate | Vocational/ Technical training | Bachelors degree | Graduate/ Post-graduate degree |
|---|---|---|---|---|---|---|
| Associates degree | - | | | | | |
| Some college | *** | - | | | | |
| High school graduate | *** | *** | - | | | |
| Vocational/Technical training | *** | *** | *** | - | | |
| Bachelors degree | *** | *** | *** | *** | - | |
| Graduate/Post-graduate degree | *** | *** | *** | *** | *** | - |

Figure A.15: Significant code: Degree of speeding across education level

## A.4 Algorithm

### A.4.1 Trip Generation

```
# Trip generation. Divide the GPS data into trip data.
# When the gap time is more than 300 seconds, divide the data.

import csv, dateutil, os
from itertools import islice
from dateutil import parser



tripEndTimeList = []
title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)']

def getTimeElapsed(time1, time2):
        t1 = parser.parse(time1)
        t2 = parser.parse(time2)
        return (t2 - t1).seconds

def countTrips():
        count = 0
        line = 0
        root_path = "/Users/toshiyokoo/Desktop/result1"
        dir_path = os.path.join(root_path,filename[0:12])
        #os.mkdir(dir_path)
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        fileCount=len(inputFile.readlines())
        inputFile.seek(0)
        oldTime = None
```

```
oldDate = None
for row in islice(reader, 1, fileCount-1):
        line +=1
        if oldDate == None:
                oldDate = row[7]
                oldTime = row[8]
                tripFile = open(os.path.join(root_path,
                    filename[0:12]+'_'+str(count+1))+'.
                    csv','a')
                print >>tripFile,','.join(title)
                print >>tripFile,','.join(row)
                tripFile.close()
                continue
        else:
                newDate = row[7]
                newTime = row[8]
                oldline = line
                if line < fileCount-2:
                        if newDate != oldDate:
                                tripEndTimeList.append(
                                    oldTime)
                                oldDate = newDate
                                oldTime = newTime
                                count +=1
                                tripFile = open(os.path
                                    .join(root_path,
                                    filename[0:12]+'_'+
                                    str(count+1))+'.csv
                                    ','a')
                                print >>tripFile,','.
                                    join(title)
```

```
                print >>tripFile,','.
                    join(row)
                tripFile.close()
                continue
        if newDate == oldDate:
                if getTimeElapsed(
                    oldTime, newTime)
                    >=300:
                        tripEndTimeList
                            .append(
                            oldTime)
                        oldDate =
                            newDate
                        oldTime =
                            newTime
                        count += 1
                        tripFile = open
                            (os.path.
                            join(
                            root_path,
                            filename
                            [0:12]+'_'+
                            str(count+1)
                            )+'.csv','a
                            ')
                        print >>
                            tripFile
                            ,',',.join(
                            title)
```

```
                print >>
                    tripFile
                    ,',',.join(
                    row)
                tripFile.close
                    ()
                continue
            else:
                oldDate =
                    newDate
                oldTime =
                    newTime
                tripFile = open
                    (os.path.
                    join(
                    root_path,
                    filename
                    [0:12]+'_'+
                    str(count+1)
                    )+'.csv','a
                    ')
                print >>
                    tripFile
                    ,',',.join(
                    row)
                tripFile.close
                continue
    if line == fileCount-2:
            tripFile = open(os.path.join(
                root_path,filename[0:12]+'_
                '+str(count+1))+'.csv','a')
            print >>tripFile,',',.join(row)
```

```
                                        tripFile.close
                                        count +=1
                                        tripEndTimeList.append(row[8])
        print filename[0:12], count

path = "/Users/toshiyokoo/Desktop/aaaac"
for filename in os.listdir(path):
        global filename
        countTrips()
```

### A.4.2 Remove outside Minneapolis - St. Paul region (7 counties)

```
# Remove GPS points which are outside Minneapolis.
# Longitude: -94.0123 <= Minneapolis <= -92.7397
# Latitude: 44.4714 <= Minneapolis <= 45.4139

import csv,os
from itertools import islice

title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)']
def insideMap():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root_path = "/Users/toshiyokoo/Desktop/02_result"
        #dir_path = os.path.join(root_path,directory)
        #if not os.path.exists( dir_path):
        #       os.mkdir( dir_path)

        count = 0
        all_count = 0
```

```
#writer = csv.writer(outputFile, lineterminator='\n')
for row in islice(reader,1,None):
        if ( -94.0123 <= float(row[0]) <= -92.7397) and
            ( 44.4714 <= float(row[1]) <= 45.4139) :
                count += 1
                all_count += 1
        else:
                all_count += 1

inputFile.seek(0)
if count == all_count:
        print filename[:-4] + 'is full'
        outputFile = open(os.path.join(root_path,
           filename[:-4]+'.csv'),'a')
        print >> outputFile, ','.join(title)
        for row in islice(reader,1,None):
                print >> outputFile, ','.join(row)
                #writer.writerow(row)
        outputFile.close()
elif 0 < count < all_count:
        print filename[:-4] + 'is modified'
        outputFile = open(os.path.join(root_path,
           filename[:-4]+'.csv'),'a')
        print >> outputFile, ','.join(title)
        for row in islice(reader,1,None):
                if ( -94.0123 <= float(row[0]) <=
                   -92.7397) and ( 44.4714 <= float(row
                   [1]) <= 45.4139):
                        print >> outputFile, ','.join(
                           row)
                        #writer.writerow(row)
        outputFile.close()
```

```
        elif count == 0:
                print filename[:-4] + 'is empty'
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaaau"
for filename in os.listdir(path):
        global filename
        insideMap()
```

### A.4.3 Remove speed that is less than 1 km/h

```
# Remove data whose speed is less than 1km/h.
import csv,os
from itertools import islice

title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)']
def smallspeed():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root_path = "/Users/toshiyokoo/Desktop/03_result"
        #dir_path = os.path.join(root_path,directory)
        #if not os.path.exists(dir_path):
        #       os.mkdir(dir_path)

        count = 0
        all_count = 0
        for row in islice(reader,1,None):
                if int(row[2]) >= 1:
                        count += 1
                        all_count += 1
```

```
                else:
                        all_count += 1

        inputFile.seek(0)
        if count == all_count:
                print filename[:-4] + 'is full'
                outputFile = open(os.path.join(root_path,
                   filename[:-4]+'.csv'),'a')
                writer = csv.writer(outputFile)
                print >> outputFile, ','.join(title)
                for row in islice(reader,1,None):
                        print >> outputFile, ','.join(row)
                outputFile.close()
        elif 0 < count < all_count:
                print filename[:-4] + 'is modified'
                outputFile = open(os.path.join(root_path,
                   filename[:-4]+'.csv'),'a')
                writer = csv.writer(outputFile)
                print >> outputFile, ','.join(title)
                for row in islice(reader,1,None):
                        if int(row[2]) >= 1:
                                print >> outputFile, ','.join(
                                   row)
                outputFile.close()
        elif count == 0:
                print filename[:-4] + 'is empty'
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaadw"
for filename in os.listdir(path):
        global filename
        smallspeed()
```

### A.4.4 Mode classification

```
# This program is mode classification. Five kinds of modes are
    found in the trips.
# Walk, off-network public transport mode, bus, bicycle and car
    .

import csv, dateutil, os, math
from scipy import stats
from itertools import islice
from dateutil import parser

def distanceBus():
        busStopFile = open("/Users/toshiyokoo/Desktop/5133/
            bus_stop.csv",'rU')
        busStop = csv.reader(busStopFile)
        inputFile.seek(0)
        GPSLonS = 0
        GPSLatS = 0
        startBusstop = 0
        for row in islice(reader,1,None):
                if int(row[2])>= 10: #speed is larger than 10km
                    /h
                        changeLonS = float(row[0])
                        changeLatS = float(row[1])
                        Longi_rad = changeLonS * math.pi / 180
                                    #Longitude_radians
                        Lati_rad = changeLatS * math.pi / 180
                                     #Latitude_radians
                        f = 1 / 298.257223563
                                                 #inverse
                            flattening
```

```
n = f / (2 - f)
a = 6378137

                #equatorial radius (meter)
A = (a / (1 + n)) * (1 + (n ** 2) / 4 +
    (n ** 4) / 64)
alpha1 = 0.5 * n - 0.666666667 * n ** 2
    + 0.3125 * 1.0 * n ** 3 #1/2, 2/3,
   5/16
alpha2 = 0.270833333 * n ** 2 - 0.6 * n
    ** 3     #13/48, 3/5
alpha3 = 0.254166667 * n ** 3    #61/240
t = math.sinh(math.atanh(math.sin(
   Lati_rad)) - ((2 * n ** 0.5) / (1 +
   n)) * math.atanh(((2 * n ** 0.5) /
   (1 + n))* math.sin(Lati_rad)))
Initial_deg = 15 * 6 - 183
                                #zone15
Initial_rad = Initial_deg * math.pi /
   180
xi = math.atan(t / math.cos(Longi_rad -
    Initial_rad))
eta = math.atanh(math.sin(Longi_rad -
   Initial_rad) / ((1 + t ** 2) ** 0.5)
   )
E0 = 500000

              #initial (meter)
k0 = 0.9996
```

```
                    E = E0 + k0 * A * (eta + alpha1 * math.
                        cos(2 * xi) * math.sinh(2 * eta) +
                        alpha2 * math.cos(4 * xi) * math.
                        sinh(4 * eta) + alpha3 * math.cos(6
                        * xi) * math.sinh(6 * eta))
                    N = k0 * A * (xi + alpha1 * math.sin(2
                        * xi) * math.cosh(2 * eta) + alpha2
                        * math.sin(4 * xi) * math.cosh(4 *
                        eta) + alpha3 * math.sin(6 * xi) *
                        math.cosh(6 * eta))
                    GPSLonS = float(E)
                    GPSLatS = float(N)
                    #print GPSLonS, GPSLatS
                    break
            else:
                    GPSLonS = 0
                    GPSLatS = 0
                    startBusstop = 0

i = 1
GPSLonE = 0
GPSLatE = 0
endBusstop = 0
for row in reversed(list(reader)):
        if i < fileCount - 2:
                if int(row[2])> 10: #speed is greater
                    than 10km/h
                        changeLonE = float(row[0])
                        changeLatE = float(row[1])
                        Longi_rad = changeLonE * math.
                            pi / 180          #
                            Longitude_radians
```

```
Lati_rad = changeLatE * math.pi
    / 180           #
    Latitude_radians
f = 1 / 298.257223563


          #inverse flattening
n = f / (2 - f)
a = 6378137


                              #
    equatorial radius (meter)
A = (a / (1 + n)) * (1 + (n **
    2) / 4 + (n ** 4) / 64)
alpha1 = 0.5 * n - 0.666666667
    * n ** 2 + 0.3125 * 1.0 * n
    ** 3 #1/2, 2/3, 5/16
alpha2 = 0.270833333 * n ** 2 -
     0.6 * n ** 3     #13/48, 3/5
alpha3 = 0.254166667 * n ** 3
       #61/240
t = math.sinh(math.atanh(math.
    sin(Lati_rad)) - ((2 * n **
    0.5) / (1 + n)) * math.atanh
    (((2 * n ** 0.5) / (1 + n))*
     math.sin(Lati_rad)))
Initial_deg = 15 * 6 - 183


          #zone15
Initial_rad = Initial_deg *
    math.pi / 180
xi = math.atan(t / math.cos(
    Longi_rad - Initial_rad))
```

```
                    eta = math.atanh(math.sin(
                        Longi_rad - Initial_rad) /
                        ((1 + t ** 2) ** 0.5))
                    E0 = 500000

                                                #
                        initial (meter)
                    k0 = 0.9996
                    E = E0 + k0 * A * (eta + alpha1
                         * math.cos(2 * xi) * math.
                        sinh(2 * eta) + alpha2 *
                        math.cos(4 * xi) * math.sinh
                        (4 * eta) + alpha3 * math.
                        cos(6 * xi) * math.sinh(6 *
                        eta))
                    N = k0 * A * (xi + alpha1 *
                        math.sin(2 * xi) * math.cosh
                        (2 * eta) + alpha2 * math.
                        sin(4 * xi) * math.cosh(4 *
                        eta) + alpha3 * math.sin(6 *
                         xi) * math.cosh(6 * eta))
                    GPSLonE = float(E)
                    GPSLatE = float(N)
                    #print GPSLonE, GPSLatE
                    break
                else:
                    GPSLonE = 0
                    GPSLatE = 0
                    endBusstop = 0
            else:
                GPSLonE = 0
                GPSLatE = 0
```

```
                endBusstop = 0
        i +=1

if (GPSLonS != 0 and GPSLatS != 0 and GPSLonE != 0 and
    GPSLatE != 0 and GPSLonS != GPSLonE and GPSLatS !=
    GPSLatE):
        for row in islice(busStop, 1, None):
                busLonS = float(row[0])
                busLatS = float(row[1])
                dS = ((GPSLonS - busLonS) ** 2 + (
                    GPSLatS - busLatS) ** 2) ** 0.5
                #print 'busS=', dS
                if dS <=50: #distance less than 50
                    meters
                        startBusstop = 1
                        #print startBusstop
                        break
                else:
                        startBusstop = 0

        busStopFile.seek(0)
        for row in islice(busStop, 1, None):
                busLonE = float(row[0])
                busLatE = float(row[1])
                dE = ((GPSLonE - busLonE) ** 2 + (
                    GPSLatE - busLatE) ** 2) ** 0.5
                #print 'busE=', dE
                if dE <=50: #distance less than 50
                    meters
                        endBusstop = 1
                        break
                else:
```

```
                                endBusstop = 0
        else:
                startBusstop = 0
                endBusstop = 0

        if (startBusstop == 1 and endBusstop == 1):
                distanceBus = 1
        else:
                distanceBus = 0

        return distanceBus

def distanceTrain():
        trainStopFile = open("/Users/toshiyokoo/Desktop/5133/
           train_station.csv",'rU')
        trainStop = csv.reader(trainStopFile)
        inputFile.seek(0)
        GPSLonS = 0
        GPSLatS = 0
        startTrainstop = 0
        for row in islice(reader,1,None):
                if int(row[2])>= 10: #speed is larger than 10km
                   /h
                        TchangeLonS = float(row[0])
                        TchangeLatS = float(row[1])
                        Longi_rad = TchangeLonS * math.pi / 180
                                    #Longitude_radians
                        Lati_rad = TchangeLatS * math.pi / 180
                                     #Latitude_radians
                        f = 1 / 298.257223563
                                                  #inverse
                           flattening
```

```
n = f / (2 - f)
a = 6378137

                #equatorial radius (meter)
A = (a / (1 + n)) * (1 + (n ** 2) / 4 +
    (n ** 4) / 64)
alpha1 = 0.5 * n - 0.666666667 * n ** 2
    + 0.3125 * 1.0 * n ** 3 #1/2, 2/3,
   5/16
alpha2 = 0.270833333 * n ** 2 - 0.6 * n
    ** 3     #13/48, 3/5
alpha3 = 0.254166667 * n ** 3    #61/240
t = math.sinh(math.atanh(math.sin(
   Lati_rad)) - ((2 * n ** 0.5) / (1 +
   n)) * math.atanh(((2 * n ** 0.5) /
   (1 + n))* math.sin(Lati_rad)))
Initial_deg = 15 * 6 - 183
                                #zone15
Initial_rad = Initial_deg * math.pi /
   180
xi = math.atan(t / math.cos(Longi_rad -
    Initial_rad))
eta = math.atanh(math.sin(Longi_rad -
   Initial_rad) / ((1 + t ** 2) ** 0.5)
   )
E0 = 500000

              #initial (meter)
k0 = 0.9996
```

```
                    E = E0 + k0 * A * (eta + alpha1 * math.
                        cos(2 * xi) * math.sinh(2 * eta) +
                        alpha2 * math.cos(4 * xi) * math.
                        sinh(4 * eta) + alpha3 * math.cos(6
                        * xi) * math.sinh(6 * eta))
                    N = k0 * A * (xi + alpha1 * math.sin(2
                        * xi) * math.cosh(2 * eta) + alpha2
                        * math.sin(4 * xi) * math.cosh(4 *
                        eta) + alpha3 * math.sin(6 * xi) *
                        math.cosh(6 * eta))
                    GPSLonS = float(E)
                    GPSLatS = float(N)
                    break
            else:
                    GPSLonS = 0
                    GPSLatS = 0
                    startTrainstop = 0

i = 1
GPSLonE = 0
GPSLatE = 0
endTrainstop = 0
for row in reversed(list(reader)):
        if i < fileCount - 2:
                if int(row[2])> 10: #speed is greater
                    than 10km/h
                        TchangeLonE = float(row[0])
                        TchangeLatE = float(row[1])
                        Longi_rad = TchangeLonE * math.
                            pi / 180          #
                            Longitude_radians
```

```
Lati_rad = TchangeLatE * math.
    pi / 180          #
    Latitude_radians
f = 1 / 298.257223563


          #inverse flattening
n = f / (2 - f)
a = 6378137


                              #
    equatorial radius (meter)
A = (a / (1 + n)) * (1 + (n **
    2) / 4 + (n ** 4) / 64)
alpha1 = 0.5 * n - 0.666666667
    * n ** 2 + 0.3125 * 1.0 * n
    ** 3 #1/2, 2/3, 5/16
alpha2 = 0.270833333 * n ** 2 -
     0.6 * n ** 3     #13/48, 3/5
alpha3 = 0.254166667 * n ** 3
       #61/240
t = math.sinh(math.atanh(math.
    sin(Lati_rad)) - ((2 * n **
    0.5) / (1 + n)) * math.atanh
    (((2 * n ** 0.5) / (1 + n))*
     math.sin(Lati_rad)))
Initial_deg = 15 * 6 - 183


          #zone15
Initial_rad = Initial_deg *
    math.pi / 180
xi = math.atan(t / math.cos(
    Longi_rad - Initial_rad))
```

```
                    eta = math.atanh(math.sin(
                        Longi_rad - Initial_rad) /
                        ((1 + t ** 2) ** 0.5))
                    E0 = 500000

                                                #
                        initial (meter)
                    k0 = 0.9996
                    E = E0 + k0 * A * (eta + alpha1
                         * math.cos(2 * xi) * math.
                        sinh(2 * eta) + alpha2 *
                        math.cos(4 * xi) * math.sinh
                        (4 * eta) + alpha3 * math.
                        cos(6 * xi) * math.sinh(6 *
                        eta))
                    N = k0 * A * (xi + alpha1 *
                        math.sin(2 * xi) * math.cosh
                        (2 * eta) + alpha2 * math.
                        sin(4 * xi) * math.cosh(4 *
                        eta) + alpha3 * math.sin(6 *
                         xi) * math.cosh(6 * eta))
                    GPSLonE = float(E)
                    GPSLatE = float(N)
                    break
                else:
                    GPSLonE = 0
                    GPSLatE = 0
                    endTrainstop = 0
            else:
                GPSLonE = 0
                GPSLatE = 0
                endTrainstop = 0
```

```
            i +=1

    if (GPSLonS != 0 and GPSLatS != 0 and GPSLonE != 0 and
       GPSLatE != 0 and GPSLonS != GPSLonE and GPSLatS !=
       GPSLatE):
            for row in islice(trainStop, 1, None):
                    trainLonS = float(row[0])
                    trainLatS = float(row[1])
                    dS = ((GPSLonS - trainLonS) ** 2 + (
                       GPSLatS - trainLatS) ** 2) ** 0.5
                    #print 'trainS=', dS
                    if dS <=150: #distance less than 150
                       meters
                            startTrainstop = 1
                            break
                    else:
                            startTrainstop = 0

            trainStopFile.seek(0)
            for row in islice(trainStop, 1, None):
                    trainLonE = float(row[0])
                    trainLatE = float(row[1])
                    dE = ((GPSLonE - trainLonE) ** 2 + (
                       GPSLatE - trainLatE) ** 2) ** 0.5
                    #print 'trainE=', dE
                    if dE <=150: #distance less than 150
                       meters
                            endTrainstop = 1
                            break
                    else:
                            endTrainstop = 0
    else:
```

```
                startTrainstop = 0
                endTrainstop = 0

        if (startTrainstop == 1 and endTrainstop == 1):
                distanceTrain = 1
        else:
                distanceTrain = 0

        return distanceTrain

title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)']
def checkMode():
        fileNumber = 1
        numberWalk = 0
        numberBike = 0
        numberTrain = 0
        numberBus = 0
        numberDrive = 0
        numberError = 0
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        fileCount = len(inputFile.readlines())
        global inputFile,reader,fileCount
        inputFile.seek(0)
        count = 0
        speed = []
        for row in islice(reader, 1, None):
                count += 1
                speed.append(int(row[2]))
                if (count == 1):
```

```
#                       startSpeed = (int(row[2]))
                        startTime = parser.parse(row[8])
                if (count == fileCount-1):
                        endTime = parser.parse(row[8])
        Cspeed = (sum((speed)) / count)
        print filename[:-4], Cspeed, stats.scoreatpercentile(
           speed, 85)
        inputFile.seek(0)
        if (endTime - startTime).seconds >=60:
                if ((sum((speed)) / count) <= 6 and stats.
                   scoreatpercentile(speed, 85) <= 10 and max(
                   speed) <=20):
                        root_walk = '/Users/toshiyokoo/Desktop
                           /04_result_walk'
                        outputFile_walk = open(os.path.join(
                           root_walk, filename[:-4] + '.csv'),
                           'a')
                        print >> outputFile_walk, ','.join(
                           title)
                        for row in islice(reader,1,None):
                                print >> outputFile_walk, ','.
                                   join(row)
                        outputFile_walk.close()
                        numberWalk +=1
                elif (distanceTrain()==1 and (sum((speed)) /
                   count) >= 10):
                        root_train = '/Users/toshiyokoo/Desktop
                           /04_result_train'
                        outputFile_train = open(os.path.join(
                           root_train, filename[:-4] + '.csv'),
                            'a')
```

```
            print >> outputFile_train, ','.join(
                title)
            inputFile.seek(0)
            for row in islice(reader,1,None):
                    print >> outputFile_train, ','.
                        join(row)
            outputFile_train.close()
            numberTrain +=1
    elif (distanceBus()==1 and (sum((speed)) /
        count) >= 10):
            root_bus = '/Users/toshiyokoo/Desktop
                /04_result_bus'
            outputFile_bus = open(os.path.join(
                root_bus, filename[:-4] + '.csv'), '
                a')
            print >> outputFile_bus, ','.join(title
                )
            inputFile.seek(0)
            for row in islice(reader,1,None):
                    print >> outputFile_bus, ','.
                        join(row)
            outputFile_bus.close()
            numberBus +=1
    elif (stats.scoreatpercentile(speed, 85) > 10
        and stats.scoreatpercentile(speed, 85) < 20
        and max(speed)<=30):
            root_bike = '/Users/toshiyokoo/Desktop
                /04_result_bike'
            outputFile_bike = open(os.path.join(
                root_bike, filename[:-4] + '.csv'),
                'a')
```

```
            print >> outputFile_bike, ','.join(
                title)
            inputFile.seek(0)
            for row in islice(reader,1,None):
                    print >> outputFile_bike, ','.
                        join(row)
            outputFile_bike.close()
            numberBike +=1
    elif ((sum((speed)) / count) >= 10):
            root_drive = '/Users/toshiyokoo/Desktop
                /04_result_drive'
            outputFile_drive = open(os.path.join(
                root_drive, filename[:-4] + '.csv'),
                 'a')
            print >> outputFile_drive, ','.join(
                title)
            inputFile.seek(0)
            for row in islice(reader,1,None):
                    print >> outputFile_drive, ','.
                        join(row)
            outputFile_drive.close()
            numberDrive +=1
    else:
            root_error = '/Users/toshiyokoo/Desktop
                /04_result_error'
            outputFile_error = open(os.path.join(
                root_error, filename[:-4] + '.csv'),
                 'a')
            print >> outputFile_error, ','.join(
                title)
            inputFile.seek(0)
            for row in islice(reader,1,None):
```

```
                                print >> outputFile_error, ','.
                                    join(row)
                        outputFile_error.close()
                        numberError +=1
        else:
                root_error = '/Users/toshiyokoo/Desktop/04
                    _result_error'
                outputFile_error = open(os.path.join(root_error
                    , filename[:-4] + '.csv'), 'a')
                print >> outputFile_error, ','.join(title)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile_error, ','.join(row
                            )
                outputFile_error.close()
                numberError +=1

path = "/Users/toshiyokoo/Desktop/aaadx"
for filename in os.listdir(path):
        global filename
        checkMode()
```

### A.4.5 Remove speed that is less than or equal to 5 km/h

```
# Remove data whose speed is less than 5km/h.
import csv,os
from itertools import islice

title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)']
def smallspeed():
        inputFile = open(path +'/' +filename,'rU')
```

```
reader = csv.reader(inputFile)
inputFile.seek(0)
root_path = "/Users/toshiyokoo/Desktop/05_result"
#dir_path = os.path.join(root_path,directory)
#if not os.path.exists(dir_path):
#       os.mkdir(dir_path)

count = 0
all_count = 0
for row in islice(reader,1,None):
        if int(row[2]) > 5:
                count += 1
                all_count += 1
        else:
                all_count += 1

inputFile.seek(0)
if count == all_count:
        print filename[:-4] + 'is full'
        outputFile = open(os.path.join(root_path,
           filename[:-4]+'.csv'),'a')
        writer = csv.writer(outputFile)
        print >> outputFile, ','.join(title)
        for row in islice(reader,1,None):
                print >> outputFile, ','.join(row)
        outputFile.close()
elif 0 < count < all_count:
        print filename[:-4] + 'is modified'
        outputFile = open(os.path.join(root_path,
           filename[:-4]+'.csv'),'a')
        writer = csv.writer(outputFile)
        print >> outputFile, ','.join(title)
```

```
                for row in islice(reader,1,None):
                        if int(row[2]) > 5:
                                print >> outputFile, ','.join(
                                    row)
                outputFile.close()
        elif count == 0:
                print filename[:-4] + 'is empty'
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaady"
for filename in os.listdir(path):
        global filename
        smallspeed()
```

### A.4.6 Convert coordinate system

```
# convert latitude & longitude to UTM data

import csv,os
from itertools import islice
import math

title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)','GX','GY']
def convert():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root_path = "/Users/toshiyokoo/Desktop/06_result"
        #dir_path = os.path.join(root_path,directory)
        #if not os.path.exists( dir_path):
        #       os.mkdir( dir_path)
```

```
outputFile = open(os.path.join(root_path,filename
    [:-4]+'.csv'),'a')
writer = csv.writer(outputFile)
print >> outputFile, ','.join(title)

for row in islice(reader, 1, None):
        Longi_deg = float(row[0])
                            #Longitude_degree
        Lati_deg = float(row[1])
                            #Latitude_degree
        Longi_rad = Longi_deg * math.pi / 180
                  #Longitude_radians
        Lati_rad = Lati_deg * math.pi / 180
                  #Latitude_radians
        f = 1 / 298.257223563
                            #inverse flattening
        n = f / (2 - f)
        a = 6378137
                                              #equatorial
            radius (meter)
        A = (a / (1 + n)) * (1 + (n ** 2) / 4 + (n **
            4) / 64)
        alpha1 = 0.5 * n - 0.666666667 * n ** 2 +
            0.3125 * 1.0 * n ** 3 #1/2, 2/3, 5/16
        alpha2 = 0.270833333 * n ** 2 - 0.6 * n ** 3
                #13/48, 3/5
        alpha3 = 0.254166667 * n ** 3    #61/240
        t = math.sinh(math.atanh(math.sin(Lati_rad)) -
            ((2 * n ** 0.5) / (1 + n)) * math.atanh(((2
            * n ** 0.5) / (1 + n))* math.sin(Lati_rad)))
        Initial_deg = 15 * 6 - 183
                            #zone15
```

```
                Initial_rad = Initial_deg * math.pi / 180
                xi = math.atan(t / math.cos(Longi_rad -
                    Initial_rad))
                eta = math.atanh(math.sin(Longi_rad -
                    Initial_rad) / ((1 + t ** 2) ** 0.5))
                E0 = 500000
                                                #initial (
                    meter)
                k0 = 0.9996
                E = E0 + k0 * A * (eta + alpha1 * math.cos(2 *
                    xi) * math.sinh(2 * eta) + alpha2 * math.cos
                    (4 * xi) * math.sinh(4 * eta) + alpha3 *
                    math.cos(6 * xi) * math.sinh(6 * eta))
                N = k0 * A * (xi + alpha1 * math.sin(2 * xi) *
                    math.cosh(2 * eta) + alpha2 * math.sin(4 *
                    xi) * math.cosh(4 * eta) + alpha3 * math.sin
                    (6 * xi) * math.cosh(6 * eta))
                East = str(E)
                North = str(N)
                row.extend([East,North])
                print >> outputFile, ','.join(row)
        print filename[:-4]
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaaea"
for filename in os.listdir(path):
        global filename
        convert()
```

### A.4.7 Matching of person ID

```
# extract file that match tripid of TBI data
```

```
import csv,os
from itertools import islice
import math
import shutil

title = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)','GX','GY']
def matching():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        tripid = filename[0:6] + filename[7:9] + filename
            [10:12]
        TBIFile = open("/Users/toshiyokoo/Desktop/5133/TBIdata.
            csv",'rU')
        idlist = csv.reader(TBIFile)
        TBIFile.seek(0)
        id = 0
        for row in islice(idlist, 1, None):
                if int(row[3]) == int(tripid):
                        root_path = '/Users/toshiyokoo/Desktop
                            /07_result'
                        outputFile = open(os.path.join(
                            root_path, filename[:-4] + '.csv'),
                            'a')
                        print >> outputFile, ','.join(title)
                        inputFile.seek(0)
                        for row in islice(reader,1,None):
                                print >> outputFile, ','.join(
                                    row)
                        outputFile.close()
                        print filename[:-4], "match"
```

```
                    id = 1

        if id == 0:
                root_error = '/Users/toshiyokoo/Desktop/07
                    _result_error'
                outputFile_error = open(os.path.join(root_error
                    , filename[:-4] + '.csv'), 'a')
                print >> outputFile_error, ','.join(title)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile_error, ','.join(row
                            )
                outputFile_error.close()
                print filename[:-4], "mismatch"
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaaee"
for filename in os.listdir(path):
        global filename
        matching()
```

### A.4.8 Remove matching error and Map-matching

```
#load GPS
#--- 1 Set pathname here
InDrPth1='Users/toshiyokoo/Desktop/aaadm/'

#--- 2  Set pathname here
InDrPth2='Users/toshiyokoo/Desktop/aaadn/'

#execute algorithm_sub0
execfile('Users/toshiyokoo/Desktop/Matching/sub0_matching.py')

import csv,os, numpy, math
```

```
from itertools import islice
import processing
import time

def matching():
        #load GIS map
        #--- 1 Load map layer
        InGIS = "/Users/toshiyokoo/Desktop/Buffer0318/10
            m_buffer.shp"
        vlayer = QgsVectorLayer(InGIS, "10m_buffer", "ogr")

        #--- 2 Confirm something is loaded and valid
        vlayer.isValid()

        #--- 3 Display the layer into QGIS
        QgsMapLayerRegistry.instance().addMapLayer(vlayer)

        #Part1
        #load GPS data
        #--- 2 Set file name here
        InFlnm1=filename
        #--- 3 Build file name an path for uri
        InFlPth1="file:///"+InDrPth1+InFlnm1
        #gps_File = open(InDrPth1 +InFlnm1,'rU')
        #gps_File.seek(0)
        #--- 4 Set import Sting here note only need to set x
            and y other come for free
        uri1 = InFlPth1+"?delimiter=%s&xField=%s&yField=%s" %
            (",", "GX", "GY")
        # "Longitude", "Latitude" -> "GX", "GY"
        #--- 5 Load point layer
        bh1 = QgsVectorLayer(uri1, InFlnm1, "delimitedtext")
```

```
#—— 6 Confirm something is loaded and valid
bh1.isValid()
#—— 7 Set CRS
bh1.setCrs(QgsCoordinateReferenceSystem(26915,
    QgsCoordinateReferenceSystem.EpsgCrsId))
#4326: WGS84, 26915: NAD83/UTM zone 15N
#—— 8 Display the layer into QGIS
QgsMapLayerRegistry.instance().addMapLayer(bh1)
#—— select by location
processing.runalg("qgis:selectbylocation", vlayer, uri1
    , 0)
#——save vector layer as csv (True: selected, False:
    all)
cLayer = qgis.utils.iface.mapCanvas().layers()[1]
gi1 = 'GIS1_'+InFlnm1
gis1 = 'Users/toshiyokoo/Desktop/5536/'+gi1
QgsVectorFileWriter.writeAsVectorFormat(cLayer, gis1, "
    CP1250", None, "CSV", True)
global InDrPth2, gis1, InFlnm1
#execute algorithm_sub1
#remove intersection & missing link GPS data
execfile('Users/toshiyokoo/Desktop/Matching/
    sub1_remove_intersect.py')
#Part2
#load GPS data
#—— 1 Set file name here
InFlnm2=InFlnm1
#—— 2  Set pathname here
#InDrPth2='Users/toshiyokoo/Desktop/5880/'
#—— 3 Build file name an path for uri
InFlPth2="file:///"+InDrPth2+InFlnm2
```

```
#--- 4 Set import Sting here note only need to set x
    and y other come for free
uri2 = InFlPth2+"?delimiter=%s&xField=%s&yField=%s" %
    (",", "GX", "GY")
# "Longitude", "Latitude" -> "GX", "GY"
#--- 5 Load point layer
bh2 = QgsVectorLayer(uri2, InFlnm2, "delimitedtext")
#--- 6 Confirm something is loaded and valid
bh2.isValid()
#--- 7 Set CRS
bh2.setCrs(QgsCoordinateReferenceSystem(26915,
    QgsCoordinateReferenceSystem.EpsgCrsId))
#4326: WGS84, 26915: NAD83/UTM zone 15N
#--- 8 Display the layer into QGIS
QgsMapLayerRegistry.instance().addMapLayer(bh2)
#--- select by location
processing.runalg("qgis:selectbylocation", vlayer, uri2
    , 0)
#---save vector layer as csv (True: selected, False:
    all)
cLayer = qgis.utils.iface.mapCanvas().layers()[2]
gi2 = 'GIS2_'+filename
gis2 = 'Users/toshiyokoo/Desktop/5536/'+gi2
global gis2, InFlnm2
QgsVectorFileWriter.writeAsVectorFormat(cLayer, gis2, "
    CP1250", None, "CSV", True)
#execute algorithm_sub2
# find nearlestlink between GPS point and GIS map
# calculate minimum height
# link nearlest(Streetname and Speedlimit)
execfile('Users/toshiyokoo/Desktop/Matching/
    sub2_matching.py')
```

```
        #remove GPS layer
        QgsMapLayerRegistry.instance().removeMapLayer( bh1.id()
            )
        QgsMapLayerRegistry.instance().removeMapLayer( bh2.id()
            )
        QgsMapLayerRegistry.instance().removeMapLayer( vlayer.
           id() )
        #reset selected layer
        #cLayer.setSelectedFeatures([0])
        #-- 1 Set file name here
        InFlnm3=InFlnm2
        InDrPth3='/Users/toshiyokoo/Desktop/box/'+InFlnm3
        #move file
        os.rename(gis1, "/Users/toshiyokoo/Desktop/comp1/"+gi1)
        os.rename(gis2, "/Users/toshiyokoo/Desktop/comp1/"+gi2)
        os.rename(InDrPth3 , "/Users/toshiyokoo/Desktop/comp1
           /"+InFlnm3)
        #gps_File.close()

for filename in os.listdir(InDrPth1):
        #global filename, InFlnm2, gi1, gi2, gis1, gis2
        matching()

# remove intersection & missing link GPS data

import csv,os, numpy, math
from itertools import islice
import time

#start = time.time()

global title1
```

```
title1 = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)','GX','GY']

def intersect():
        gps_inputFile1 = open(InDrPth1 +InFlnm1,'rU')
        gps_reader1 = csv.reader(gps_inputFile1)
        gps_inputFile1.seek(0)
        root_path1 = InDrPth2
        outputFile1 = open(os.path.join(root_path1,InFlnm1
            [:-4]+'.csv'),'a')
        writer1 = csv.writer(outputFile1, lineterminator='\n')
        print >> outputFile1, ','.join(title1)
        gis_inputFile1 = open(gis1,'rU')
        gis_reader1 = csv.reader(gis_inputFile1)
        min_height1 = 99 #dummy
        speedlimit1 = 99        #dummy
        for row1 in islice(gps_reader1, 1, None):
                list1 = []
                x_gps1 = float(row1[10])
                y_gps1 = float(row1[11])
                gis_inputFile1.seek(0)
                count1 = 0
                for j in islice(gis_reader1, 1, None):
                        direction1 = float(j[42])
                        ox_gis1 = float(j[43]) #OX
                        oy_gis1 = float(j[44]) #OY
                        dx_gis1 = float(j[45]) #DX
                        dy_gis1 = float(j[46]) #DY
                        if direction1 == 1:
                                if (ox_gis1 - 10<= x_gps1 <=
                                    dx_gis1 + 10):
```

```
                                        if (oy_gis1 - 10) <=
                                           y_gps1 <= (dy_gis1 +
                                            10):
                                                count1 += 1
                                        elif (dy_gis1 - 10) <=
                                           y_gps1 <= (oy_gis1 +
                                            10):
                                                count1 += 1
                                elif (dx_gis1 - 10 <= x_gps1 <=
                                    ox_gis1 + 10):
                                        if (oy_gis1 - 10) <=
                                           y_gps1 <= (dy_gis1 +
                                            10):
                                                count1 += 1
                                        elif (dy_gis1 - 10) <=
                                           y_gps1 <= (oy_gis1 +
                                            10):
                                                count1 += 1

                        if count1 == 1:
                                print >> outputFile1, ','.join(row1)

                gis_inputFile1.close()
                gps_inputFile1.close()
                print InFlnm1
                #elapsed_time = time.time() - start
                #print("elapsed_time:{0}".format(elapsed_time))

#global filename, InDrPth2, InFlnm1, InFlnm2, gi1, gi2, gis1,
    gis2
intersect()
```

```
# find nearlestlink between GPS point and GIS map
# calculate minimum height
# link nearlest(Streetname and Speedlimit)

import csv,os, numpy, math
from itertools import islice
import time

#start = time.time()

global title2
title2 = ['Longitude','Latitude','Speed(kilometer)','DD/MM/YY
    ','HH:MM:SS','GX','GY','Min_hight','Streetname', '
    Streetlength', 'Roadtype','Speedlimit(mph)']

def nearlink():
        gps_inputFile = open(InDrPth2 +InFlnm2,'rU')
        gps_reader = csv.reader(gps_inputFile)
        gps_inputFile.seek(0)
        root_path = '/Users/toshiyokoo/Desktop/box'
        outputFile = open(os.path.join(root_path,InFlnm2
            [:-4]+'.csv'),'a')
        writer = csv.writer(outputFile, lineterminator='\n')
        print >> outputFile, ','.join(title2)
        gis_inputFile = open(gis2,'rU')
        gis_reader = csv.reader(gis_inputFile)
        min_height = 99 #dummy
        speedlimit = 99 #dummy
        for row in islice(gps_reader, 1, None):
                list = []
                x_gps = float(row[10])
                y_gps = float(row[11])
```

```
longitude = str(row[0])
latitude = str(row[1])
speed = str(row[2])
date = str(row[7])
hms = str(row[8])
GX = str(row[10])
GY = str(row[11])
gis_inputFile.seek(0)
for i in islice(gis_reader, 1, None):
        direction = float(i[42])
        ox_gis = float(i[43]) #OX
        oy_gis = float(i[44]) #OY
        dx_gis = float(i[45]) #DX
        dy_gis = float(i[46]) #DY
        if direction == 1:
                if (ox_gis <= x_gps <= dx_gis):
                        if (oy_gis - 30) <=
                           y_gps <= (oy_gis +
                           30):
                                length = ((
                                   ox_gis -
                                   dx_gis) ** 2
                                    + (oy_gis -
                                    dy_gis) **
                                   2) ** 0.5
                                gis_o = numpy.
                                   array([
                                   ox_gis,
                                   oy_gis])
```

```
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height

          streetlength
              =
             str(
             i
             [0])
```

```
                    streetname
                        =
                        str(
                        i
                        [5])
                    roadtype
                        =
                        str(
                        i
                        [27])

                    speedlimit
                        =
                        str(
                        i
                        [31])

                    #print
                        '1',

                        min_height

elif (oy_gis <= y_gps
  <= dy_gis):       #
  road x-y & curve
        length = ((
            ox_gis -
            dx_gis) ** 2
             + (oy_gis -
             dy_gis) **
            2) ** 0.5
```

```
gis_o = numpy.
    array([
    ox_gis,
    oy_gis])
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height
```

```
                streetlength
                    =
                    str(
                    i
                    [0])
                streetname
                    =
                    str(
                    i
                    [5])
                roadtype
                    =
                    str(
                    i
                    [27])

                speedlimit
                    =
                    str(
                    i
                    [31])

            elif (dy_gis <= y_gps
              <= oy_gis):       #
               road x-y & curve
                    length = ((
                       ox_gis -
                       dx_gis) ** 2
                        + (oy_gis -
                        dy_gis) **
                       2) ** 0.5
```

```
gis_o = numpy.
    array([
    ox_gis,
    oy_gis])
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height
```

```
                                        streetlength
                                            =
                                            str(
                                            i
                                            [0])
                                        streetname
                                            =
                                            str(
                                            i
                                            [5])
                                        roadtype
                                            =
                                            str(
                                            i
                                            [27])

                                        speedlimit
                                            =
                                            str(
                                            i
                                            [31])

            elif (dx_gis <= x_gps <= ox_gis
                ):
                    if (oy_gis - 30) <=
                        y_gps <= (oy_gis +
                        30):
```

```
length = ((
    ox_gis -
    dx_gis) ** 2
     + (oy_gis -
     dy_gis) **
    2) ** 0.5
gis_o = numpy.
    array([
    ox_gis,
    oy_gis])
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
```

```
min_height
    =
    height

streetlength
    =
    str(
    i
    [0])
streetname
    =
    str(
    i
    [5])
roadtype
    =
    str(
    i
    [27])

speedlimit
    =
    str(
    i
    [31])

#print
    '1',

    min_height
```

```
elif (oy_gis <= y_gps
   <= dy_gis):        #
   road x-y & curve
         length = ((
            ox_gis -
            dx_gis) ** 2
             + (oy_gis -
             dy_gis) **
            2) ** 0.5
         gis_o = numpy.
            array([
            ox_gis,
            oy_gis])
         gis_d = numpy.
            array([
            dx_gis,
            dy_gis])
         gps = numpy.
            array([x_gps
            ,y_gps])
         v1 = gps -
            gis_o
         v2 = gis_d -
            gis_o
         Area = numpy.
            cross(v1, v2
            )
         D = abs(Area)
         height = D /
            length
```

```
                    if
                        min_height >
                         height:
                              min_height
                                  =
                                 height

                              streetlength
                                  =
                                 str(
                                 i
                                 [0])
                              streetname
                                  =
                                 str(
                                 i
                                 [5])
                              roadtype
                                  =
                                 str(
                                 i
                                 [27])

                              speedlimit
                                  =
                                 str(
                                 i
                                 [31])

                elif (dy_gis <= y_gps
                   <= oy_gis):       #
                    road x-y & curve
```

```
length = ((
    ox_gis -
    dx_gis) ** 2
     + (oy_gis -
     dy_gis) **
    2) ** 0.5
gis_o = numpy.
    array([
    ox_gis,
    oy_gis])
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
```

```
                min_height
                    =
                    height

                streetlength
                    =
                    str(
                    i
                    [0])
                streetname
                    =
                    str(
                    i
                    [5])
                roadtype
                    =
                    str(
                    i
                    [27])

                speedlimit
                    =
                    str(
                    i
                    [31])

        elif (oy_gis <= y_gps <= dy_gis
            ):
```

```
if (ox_gis - 30) <=
    x_gps <= (ox_gis +
    30):      #road y-
    axis, interstate
    range 50m
          length = ((
              ox_gis -
              dx_gis) ** 2
               + (oy_gis -
               dy_gis) **
              2) ** 0.5
          gis_o = numpy.
              array([
              ox_gis,
              oy_gis])
          gis_d = numpy.
              array([
              dx_gis,
              dy_gis])
          gps = numpy.
              array([x_gps
              ,y_gps])
          v1 = gps -
              gis_o
          v2 = gis_d -
              gis_o
          Area = numpy.
              cross(v1, v2
              )
          D = abs(Area)
          height = D /
              length
```

```
                                    if
                                        min_height >
                                         height:
                                              min_height
                                                  =
                                                 height

                                              streetlength
                                                  =
                                                 str(
                                                 i
                                                 [0])
                                              streetname
                                                  =
                                                 str(
                                                 i
                                                 [5])
                                              roadtype
                                                  =
                                                 str(
                                                 i
                                                 [27])

                                              speedlimit
                                                  =
                                                 str(
                                                 i
                                                 [31])

                                elif (dy_gis <= y_gps <= oy_gis
                                    ):
```

```
if (ox_gis - 30) <=
    x_gps <= (ox_gis +
    30):      #road y-
    axis, interstate
    range 50m
          length = ((
              ox_gis -
              dx_gis) ** 2
               + (oy_gis -
               dy_gis) **
              2) ** 0.5
          gis_o = numpy.
              array([
              ox_gis,
              oy_gis])
          gis_d = numpy.
              array([
              dx_gis,
              dy_gis])
          gps = numpy.
              array([x_gps
              ,y_gps])
          v1 = gps -
              gis_o
          v2 = gis_d -
              gis_o
          Area = numpy.
              cross(v1, v2
              )
          D = abs(Area)
          height = D /
              length
```

```
                                        if
                                            min_height >
                                             height:
                                                 min_height
                                                     =
                                                    height

                                                 streetlength
                                                     =
                                                    str(
                                                    i
                                                    [0])
                                                 streetname
                                                     =
                                                    str(
                                                    i
                                                    [5])
                                                 roadtype
                                                     =
                                                    str(
                                                    i
                                                    [27])

                                                 speedlimit
                                                     =
                                                    str(
                                                    i
                                                    [31])

                                    else:
                                            pass
```

```
                if min_height < 30:
                        #streetlength = 99
                        #streetname = 99         #dummy
                        #roadtype = 99           #dummy
                        #speedlimit = 99         #dummy
                        Min_height = str(min_height)
                        Streetname = str(streetname)
                        Streetlength = str(streetlength)
                        Roadtype = str(roadtype)
                        Speedlimit = str(speedlimit)
                        list.extend([longitude, latitude, speed
                            , date, hms, GX, GY, Min_height,
                            Streetname, Streetlength, Roadtype,
                            Speedlimit])
                        writer.writerow(list)
                        #print >> outputFile, ','.join(row)
                        min_height = 99 #dummy
                        speedlimit = 99 #dummy
        gis_inputFile.close()
        gps_inputFile.close()

        print InFlnm2

        #elapsed_time = time.time() - start
        #print("elapsed_time:{0}".format(elapsed_time))

#global filename, InDrPth2, InFlnm2, gi1, gi2, gis1, gis2
nearlink()
```

### A.4.9 Remove data (age is less than 16)

```
import csv,os
```

```
from itertools import islice
import math

title = ['Longitude','Latitude','Speed(kilometer)','DD/MM/YY','
    HH:MM:SS','GX','GY','Min_hight','Streetname', 'Streetlength
    ', 'Roadtype','Speedlimit(mph)']
def remove():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        tripid = filename[0:6] + filename[7:9] + filename
            [10:12]
        TBIFile = open("/Users/toshiyokoo/Desktop/5133/
            TBIdata_age.csv",'rU')
        idlist = csv.reader(TBIFile)
        TBIFile.seek(0)
        id = 0
        for row in islice(idlist, 1, None):
                if int(row[3]) == int(tripid):
                        root_error = '/Users/toshiyokoo/Desktop
                            /09_result_error'
                        outputFile_error = open(os.path.join(
                            root_error, filename[:-4] + '.csv'),
                             'a')
                        print >> outputFile_error, ','.join(
                            title)
                        inputFile.seek(0)
                        for row in islice(reader,1,None):
                                print >> outputFile_error, ','.
                                    join(row)
                        outputFile_error.close()
                        print filename[:-4], "age_young"
                        id = 1
```

```
        if id == 0:
                root_path = '/Users/toshiyokoo/Desktop/09
                    _result'
                outputFile = open(os.path.join(root_path,
                    filename[:-4] + '.csv'), 'a')
                print >> outputFile, ','.join(title)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile, ','.join(row)
                outputFile.close()
                print filename[:-4], "no_error"
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaafn"
for filename in os.listdir(path):
        global filename
        remove()
```

### A.4.10 Remove speed limit that is 0 mph

```
# Remove data whose speed limit is 0 km/h.
import csv,os
from itertools import islice

title = ['Longitude','Latitude','Speed(kilometer)','DD/MM/YY','
    HH:MM:SS','GX','GY','Min_hight','Streetname', 'Streetlength
    ', 'Roadtype','Speedlimit(mph)']
def removespeed():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root1_path = "/Users/toshiyokoo/Desktop/10_result"
```

```
root2_path = "/Users/toshiyokoo/Desktop/10_result_error
    "
count = 0
all_count = 0
for row in islice(reader,1,None):
        if int(row[11]) > 0:
                count += 1
                all_count += 1
        else:
                all_count += 1

inputFile.seek(0)
if count == 0:
        print filename[:-4] + 'is empty'
        outputFile = open(os.path.join(root2_path,
            filename[:-4]+'.csv'),'a')
        writer = csv.writer(outputFile)
        print >> outputFile, ','.join(title)
        for row in islice(reader,1,None):
                print >> outputFile, ','.join(row)
        outputFile.close()
elif count == all_count:
        print filename[:-4] + 'is full'
        outputFile = open(os.path.join(root1_path,
            filename[:-4]+'.csv'),'a')
        writer = csv.writer(outputFile)
        print >> outputFile, ','.join(title)
        for row in islice(reader,1,None):
                print >> outputFile, ','.join(row)
        outputFile.close()
elif 0 < count < all_count:
        print filename[:-4] + 'is modified'
```

```
                outputFile = open(os.path.join(root1_path,
                    filename[:-4]+'.csv'),'a')
                writer = csv.writer(outputFile)
                print >> outputFile, ','.join(title)
                for row in islice(reader,1,None):
                        if int(row[11]) > 0:
                                print >> outputFile, ','.join(
                                    row)
                outputFile.close()
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaafw"
for filename in os.listdir(path):
        global filename
        removespeed()
```

### A.4.11 Create variables for statistical analysis

```
import csv,os, numpy, math
from itertools import islice
import datetime

title = ['filename', 'tripid', 'date', 'time', 'speed(kilometer
    )', 'streetlength', 'speedlimit(mph)', 'speedlimit(km)', '
    speed_limit_compliance', 'speed2', 'hour', 'speed_25', '
    speed_30', 'speed_35', 'speed_40', 'speed_45', 'speed_50', '
    speed_55', 'speed_60', 'speed_65', 'morning', 'afternoon', '
    night', 'o_gender', 'o_edu', 'o_age', 'gender', 'education_1
    ', 'education_2', 'age_25_to_34', 'age_35_to_44', '
    age_45_to_54', 'age_55_to_64', 'age_65_to_74', 'age_75_to_84
    ', 'age_85_and_over']

def summarize():
```

```
inputFile = open(path +'/' +filename,'rU')
reader = csv.reader(inputFile)
inputFile.seek(0)
root_path = '/Users/toshiyokoo/Desktop/11_result'
outputFile = open(os.path.join(root_path, filename[:-4]
    + '.csv'), 'a')
writer = csv.writer(outputFile, lineterminator='\n')
print >> outputFile, ','.join(title)
for row in islice(reader, 1, None):
        list = []
        Filename = filename[:-4]
        tripid = filename[0:6] + filename[7:9] +
            filename[10:12]
        speed = str(row[2])
        date = str(row[3])
        hms = str(row[4])
        streetlength = str(row[9])
        speedlimit = str(row[11])
        drive_speed = float(row[2])
        if date[-4:] == '2011':
                d = datetime.datetime.strptime(date, '%
                    d/%m/%Y')
        else:
                d = datetime.datetime.strptime(date, '%
                    d/%m/%y')

        #convert the unit of speedlimit from mile to
            kilometer
        mile = float(row[11])
        conv_fac = 1.609344
        kilometer = mile * conv_fac
```

```
#calculate speed limit compliance
speed_limit_compliance = drive_speed /
    kilometer

#whether the speeding or not
if (drive_speed - kilometer) > 0:
        speed2 = 1
else:
        speed2 = 0

#change the hour from UTC to MN (Time in GPS is
     Greenwich Mean Time (GMT))
f = datetime.datetime.strptime(hms, '%H:%M:%S')
hour = f.hour
date1 = datetime.datetime(2011, 3, 13)
date2 = datetime.datetime(2011, 11, 5)
#Daylight Saving Time
if date1 <= d <= date2:
        cal_hour = hour - 5
        fix_hour = cal_hour
        if cal_hour < 0:
                fix_hour = cal_hour + 24
#Standard Time
else:
        cal_hour = hour - 6
        fix_hour = cal_hour
        if cal_hour < 0:
                fix_hour = cal_hour + 24

#Dummy variable (Speed limit)
num_speedlimit = int(row[11])
speed_25 = 0
```

```
speed_30 = 0
speed_35 = 0
speed_40 = 0
speed_45 = 0
speed_50 = 0
speed_55 = 0
speed_60 = 0
speed_65 = 0
if num_speedlimit <= 25:
        speed_25 = 1
elif num_speedlimit == 30:
        speed_30 = 1
elif num_speedlimit == 35:
        speed_35 = 1
elif num_speedlimit == 40:
        speed_40 = 1
elif num_speedlimit == 45:
        speed_45 = 1
elif num_speedlimit == 50:
        speed_50 = 1
elif num_speedlimit == 55:
        speed_55 = 1
elif num_speedlimit == 60:
        speed_60 = 1
elif num_speedlimit == 65:
        speed_65 = 1
elif num_speedlimit == 70:
        dummy = 0
else:
        speed_25 = 99
        speed_30 = 99
        speed_35 = 99
```

```
            speed_40 = 99
            speed_45 = 99
            speed_50 = 99
            speed_55 = 99
            speed_60 = 99
            speed_65 = 99

        #Dummy variable (morning, afternoon, night)
        morning = 0
        afternoon = 0
        night = 0
        dummy_hour = int(fix_hour)
        if 6 <= dummy_hour < 12:
            morning = 1
        elif 12 <= dummy_hour < 18:
            afternoon = 1
        elif 18 <= dummy_hour < 24:
            night = 1
        elif 0 <= dummy_hour < 6:
            dummy = 0
        else:
            morning = 99
            afternoon = 99
            night = 99

        #Input Household survey data
        #Match Tripid between GPS file and Household
           survey data
        TBIFile = open("/Users/toshiyokoo/Desktop/5133/
           TBIdata.csv",'rU')
        idlist = csv.reader(TBIFile)
        TBIFile.seek(0)
```

```
gender = 0
age_25_to_34 = 0
age_35_to_44 = 0
age_45_to_54 = 0
age_55_to_64 = 0
age_65_to_74 = 0
age_75_to_84 = 0
age_85_and_over = 0
education_1 = 0
education_2 = 0
for dem in islice(idlist, 1, None):
        if int(dem[3]) == int(tripid):
#Dummy variable (Gender)
                o_gender = dem[4]
                if dem[4] == 'Male':
                        gender = 1
                elif dem[4] == 'Female':
                        gender = 0
                else:
                        gender = 99
#Dummy variable (Education)
                o_education = dem[5]
                if dem[5] == 'Daycare / Pre-
                   school':
                        dummy = 0
                elif dem[5] == 'Less than high
                   school':
                        dummy = 0
                elif dem[5] == 'High school
                   graduate':
                        dummy = 0
                elif dem[5] == 'Some college':
```

```
                education_1 = 1
        elif dem[5] == 'Vocational/
            Technical training':
                dummy = 0
        elif dem[5] == 'Associates
            degree':
                education_1 = 1
        elif dem[5] == 'Bachelors
            degree':
                education_2 = 1
        elif dem[5] == 'Graduate/Post-
            graduate degree':
                education_2 = 1
        else:
                education_1 = 99
                education_2 = 99
#Dummy variable (Age)
        o_age = dem[6]
        if      int(dem[6]) == 6:
                age_25_to_34 = 1
        elif    int(dem[6]) == 7:
                age_35_to_44 = 1
        elif    int(dem[6]) == 8:
                age_45_to_54 = 1
        elif    int(dem[6]) == 9:
                age_55_to_64 = 1
        elif    int(dem[6]) == 10:
                age_65_to_74 = 1
        elif    int(dem[6]) == 11:
                age_75_to_84 = 1
        elif    int(dem[6]) == 12:
                age_85_and_over = 1
```

```
                elif    int(dem[6]) == 5:
                        dummy = 0
                else:
                        age_25_to_34 = 99
                        age_35_to_44 = 99
                        age_45_to_54 = 99
                        age_55_to_64 = 99
                        age_65_to_74 = 99
                        age_75_to_84 = 99
                        age_85_and_over = 99

                list.extend([Filename, tripid, date, hms, speed
                    , streetlength, speedlimit, kilometer,
                    speed_limit_compliance, speed2, fix_hour,
                    speed_25, speed_30, speed_35, speed_40,
                    speed_45, speed_50, speed_55, speed_60,
                    speed_65, morning, afternoon, night,
                    o_gender, o_education, o_age, gender,
                    education_1, education_2, age_25_to_34,
                    age_35_to_44, age_45_to_54, age_55_to_64,
                    age_65_to_74, age_75_to_84, age_85_and_over
                    ])
                writer.writerow(list)
        inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaaif"
for filename in os.listdir(path):
        global filename
        summarize()
```

### A.4.12 Compile data

```
# Remove data whose speed limit is 0 km/h.
import csv,os
from itertools import islice

title = ['filename', 'tripid', 'date', 'time', 'speed(kilometer
    )', 'streetlength', 'speedlimit(mph)', 'speedlimit(km)', '
    speed_limit_compliance', 'speed2', 'hour', 'speed_25', '
    speed_30', 'speed_35', 'speed_40', 'speed_45', 'speed_50', '
    speed_55', 'speed_60', 'speed_65', 'morning', 'afternoon', '
    night', 'o_gender', 'o_edu', 'o_age', 'gender', 'education_1
    ', 'education_2', 'age_25_to_34', 'age_35_to_44', '
    age_45_to_54', 'age_55_to_64', 'age_65_to_74', 'age_75_to_84
    ', 'age_85_and_over']
def compile():
        inputFile = open(path +'/' +filename,'rU')
        top = filename[0:2]
        top2 = int(top)
        if top2 == 10:
                reader = csv.reader(inputFile)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile1, ','.join(row)
                inputFile.close()
        elif top2 == 11:
                reader = csv.reader(inputFile)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile2, ','.join(row)
                inputFile.close()
        elif top2 == 12:
                reader = csv.reader(inputFile)
                inputFile.seek(0)
```

```
        for row in islice(reader,1,None):
                print >> outputFile3, ','.join(row)
        inputFile.close()
elif top2 == 13:
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                print >> outputFile4, ','.join(row)
        inputFile.close()
elif top2 == 14:
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                print >> outputFile5, ','.join(row)
        inputFile.close()
elif top2 == 15:
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                print >> outputFile6, ','.join(row)
        inputFile.close()
elif top2 == 16:
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                print >> outputFile7, ','.join(row)
        inputFile.close()
elif top2 == 17:
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                print >> outputFile8, ','.join(row)
```

```
                inputFile.close()
        elif top2 == 18:
                reader = csv.reader(inputFile)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile9, ','.join(row)
                inputFile.close()
        elif top2 == 19:
                reader = csv.reader(inputFile)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile10, ','.join(row)
                inputFile.close()
        elif top2 == 20:
                reader = csv.reader(inputFile)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile11, ','.join(row)
                inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaaig"
root_path = "/Users/toshiyokoo/Desktop/12_result"
outputFile1 = open(os.path.join(root_path,"summary_10")+'.csv
    ','a')
outputFile2 = open(os.path.join(root_path,"summary_11")+'.csv
    ','a')
outputFile3 = open(os.path.join(root_path,"summary_12")+'.csv
    ','a')
outputFile4 = open(os.path.join(root_path,"summary_13")+'.csv
    ','a')
```

```
outputFile5 = open(os.path.join(root_path,"summary_14")+'.csv
    ','a')
outputFile6 = open(os.path.join(root_path,"summary_15")+'.csv
    ','a')
outputFile7 = open(os.path.join(root_path,"summary_16")+'.csv
    ','a')
outputFile8 = open(os.path.join(root_path,"summary_17")+'.csv
    ','a')
outputFile9 = open(os.path.join(root_path,"summary_18")+'.csv
    ','a')
outputFile10 = open(os.path.join(root_path,"summary_19")+'.csv
    ','a')
outputFile11 = open(os.path.join(root_path,"summary_20")+'.csv
    ','a')
writer = csv.writer(outputFile1)
writer = csv.writer(outputFile2)
writer = csv.writer(outputFile3)
writer = csv.writer(outputFile4)
writer = csv.writer(outputFile5)
writer = csv.writer(outputFile6)
writer = csv.writer(outputFile7)
writer = csv.writer(outputFile8)
writer = csv.writer(outputFile9)
writer = csv.writer(outputFile10)
writer = csv.writer(outputFile11)
print >> outputFile1, ','.join(title)
print >> outputFile2, ','.join(title)
print >> outputFile3, ','.join(title)
print >> outputFile4, ','.join(title)
print >> outputFile5, ','.join(title)
print >> outputFile6, ','.join(title)
print >> outputFile7, ','.join(title)
```

```
print >> outputFile8, ','.join(title)
print >> outputFile9, ','.join(title)
print >> outputFile10, ','.join(title)
print >> outputFile11, ','.join(title)
for filename in os.listdir(path):
        global filename
        compile()
outputFile1.close()
outputFile2.close()
outputFile3.close()
outputFile4.close()
outputFile5.close()
outputFile6.close()
outputFile7.close()
outputFile8.close()
outputFile9.close()
outputFile10.close()
outputFile11.close()
```

### A.4.13 t-test: time

```
# compute t-test and average

import csv,os, pandas
import pandas as pd
from scipy import stats
from itertools import islice
import math
import numpy

path = "/Users/toshiyokoo/Desktop/aaals"
count = 1
for filename in os.listdir(path):
```

```
if count == 1:
        data = pandas.read_csv(path +'/' +filename)
        am0 = data[data['hour'] == 0]['
            speed_limit_compliance']
        am1 = data[data['hour'] == 1]['
            speed_limit_compliance']
        am2 = data[data['hour'] == 2]['
            speed_limit_compliance']
        am3 = data[data['hour'] == 3]['
            speed_limit_compliance']
        am4 = data[data['hour'] == 4]['
            speed_limit_compliance']
        am5 = data[data['hour'] == 5]['
            speed_limit_compliance']
        am6 = data[data['hour'] == 6]['
            speed_limit_compliance']
        am7 = data[data['hour'] == 7]['
            speed_limit_compliance']
        am8 = data[data['hour'] == 8]['
            speed_limit_compliance']
        am9 = data[data['hour'] == 9]['
            speed_limit_compliance']
        am10 = data[data['hour'] == 10]['
            speed_limit_compliance']
        am11 = data[data['hour'] == 11]['
            speed_limit_compliance']
        pm12 = data[data['hour'] == 12]['
            speed_limit_compliance']
        pm1 = data[data['hour'] == 13]['
            speed_limit_compliance']
        pm2 = data[data['hour'] == 14]['
            speed_limit_compliance']
```

```
            pm3 = data[data['hour'] == 15]['
                speed_limit_compliance']
            pm4 = data[data['hour'] == 16]['
                speed_limit_compliance']
            pm5 = data[data['hour'] == 17]['
                speed_limit_compliance']
            pm6 = data[data['hour'] == 18]['
                speed_limit_compliance']
            pm7 = data[data['hour'] == 19]['
                speed_limit_compliance']
            pm8 = data[data['hour'] == 20]['
                speed_limit_compliance']
            pm9 = data[data['hour'] == 21]['
                speed_limit_compliance']
            pm10 = data[data['hour'] == 22]['
                speed_limit_compliance']
            pm11 = data[data['hour'] == 23]['
                speed_limit_compliance']
            print filename + "1"
    else:
            data = pandas.read_csv(path +'/' +filename)
            add0 = data[data['hour'] == 0]['
                speed_limit_compliance']
            add1 = data[data['hour'] == 1]['
                speed_limit_compliance']
            add2 = data[data['hour'] == 2]['
                speed_limit_compliance']
            add3 = data[data['hour'] == 3]['
                speed_limit_compliance']
            add4 = data[data['hour'] == 4]['
                speed_limit_compliance']
```

```
add5 = data[data['hour'] == 5]['
    speed_limit_compliance']
add6 = data[data['hour'] == 6]['
    speed_limit_compliance']
add7 = data[data['hour'] == 7]['
    speed_limit_compliance']
add8 = data[data['hour'] == 8]['
    speed_limit_compliance']
add9 = data[data['hour'] == 9]['
    speed_limit_compliance']
add10 = data[data['hour'] == 10]['
    speed_limit_compliance']
add11 = data[data['hour'] == 11]['
    speed_limit_compliance']
pdd12 = data[data['hour'] == 12]['
    speed_limit_compliance']
pdd1 = data[data['hour'] == 13]['
    speed_limit_compliance']
pdd2 = data[data['hour'] == 14]['
    speed_limit_compliance']
pdd3 = data[data['hour'] == 15]['
    speed_limit_compliance']
pdd4 = data[data['hour'] == 16]['
    speed_limit_compliance']
pdd5 = data[data['hour'] == 17]['
    speed_limit_compliance']
pdd6 = data[data['hour'] == 18]['
    speed_limit_compliance']
pdd7 = data[data['hour'] == 19]['
    speed_limit_compliance']
pdd8 = data[data['hour'] == 20]['
    speed_limit_compliance']
```

```
            pdd9 = data[data['hour'] == 21]['
                speed_limit_compliance']
            pdd10 = data[data['hour'] == 22]['
                speed_limit_compliance']
            pdd11 = data[data['hour'] == 23]['
                speed_limit_compliance']
            am0 = pd.concat([am0,add0])
            am1 = pd.concat([am1,add1])
            am2 = pd.concat([am2,add2])
            am3 = pd.concat([am3,add3])
            am4 = pd.concat([am4,add4])
            am5 = pd.concat([am5,add5])
            am6 = pd.concat([am6,add6])
            am7 = pd.concat([am7,add7])
            am8 = pd.concat([am8,add8])
            am9 = pd.concat([am9,add9])
            am10 = pd.concat([am10,add10])
            am11 = pd.concat([am11,add11])
            pm12 = pd.concat([pm12,pdd12])
            pm1 = pd.concat([pm1,pdd1])
            pm2 = pd.concat([pm2,pdd2])
            pm3 = pd.concat([pm3,pdd3])
            pm4 = pd.concat([pm4,pdd4])
            pm5 = pd.concat([pm5,pdd5])
            pm6 = pd.concat([pm6,pdd6])
            pm7 = pd.concat([pm7,pdd7])
            pm8 = pd.concat([pm8,pdd8])
            pm9 = pd.concat([pm9,pdd9])
            pm10 = pd.concat([pm10,pdd10])
            pm11 = pd.concat([pm11,pdd11])
            print filename + "2"
    if count == 11:
```

```
                print "am4, am7", stats.ttest_ind(am4,am7,
                    equal_var = False)
                print "pm12, pm2", stats.ttest_ind(pm12,pm2,
                    equal_var = False)
                print numpy.mean(am0)
                print numpy.mean(am1)
                print numpy.mean(am2)
                print numpy.mean(am3)
                print numpy.mean(am4)
                print numpy.mean(am5)
                print numpy.mean(am6)
                print numpy.mean(am7)
                print numpy.mean(am8)
                print numpy.mean(am9)
                print numpy.mean(am10)
                print numpy.mean(am11)
                print numpy.mean(pm12)
                print numpy.mean(pm1)
                print numpy.mean(pm2)
                print numpy.mean(pm3)
                print numpy.mean(pm4)
                print numpy.mean(pm5)
                print numpy.mean(pm6)
                print numpy.mean(pm7)
                print numpy.mean(pm8)
                print numpy.mean(pm9)
                print numpy.mean(pm10)
                print numpy.mean(pm11)
        count += 1
```

### A.4.14 t-test: gender

```
# compute t-test and average
```

```
import csv,os, pandas
import pandas as pd
from scipy import stats
from itertools import islice
import math
import numpy

path = "/Users/toshiyokoo/Desktop/aaals"
count = 1
for filename in os.listdir(path):
        if count == 1:
                data = pandas.read_csv(path +'/' +filename)
                female = data[data['gender'] == 0]['
                   speed_limit_compliance']
                male = data[data['gender'] == 1]['
                   speed_limit_compliance']
                print filename + "1"
        else:
                data = pandas.read_csv(path +'/' +filename)
                add1 = data[data['gender'] == 0]['
                   speed_limit_compliance']
                add2 = data[data['gender'] == 1]['
                   speed_limit_compliance']
                female = pd.concat([female,add1])
                male = pd.concat([male,add2])
                print filename + "2"

        if count == 11:
                print stats.ttest_ind(female, male, equal_var =
                    False)
                print numpy.mean(female)
```

```
                print numpy.mean(male)
        count += 1
```

### A.4.15 t-test: age

```
# extract file that match tripid of TBI data

import csv,os, pandas
import pandas as pd
from scipy import stats
from itertools import islice
import math
import numpy

path = "/Users/toshiyokoo/Desktop/aaals"
count = 1
for filename in os.listdir(path):
        if count == 1:
                data = pandas.read_csv(path +'/' +filename)
                age5 = data[data['o_age'] == 5]['
                   speed_limit_compliance']
                age6 = data[data['o_age'] == 6]['
                   speed_limit_compliance']
                age7 = data[data['o_age'] == 7]['
                   speed_limit_compliance']
                age8 = data[data['o_age'] == 8]['
                   speed_limit_compliance']
                age9 = data[data['o_age'] == 9]['
                   speed_limit_compliance']
                age10 = data[data['o_age'] == 10]['
                   speed_limit_compliance']
                age11 = data[data['o_age'] == 11]['
                   speed_limit_compliance']
```

```
            age12 = data[data['o_age'] == 12]['
                speed_limit_compliance']
            print filename + "1"
    else:
            data = pandas.read_csv(path +'/' +filename)
            add5 = data[data['o_age'] == 5]['
                speed_limit_compliance']
            add6 = data[data['o_age'] == 6]['
                speed_limit_compliance']
            add7 = data[data['o_age'] == 7]['
                speed_limit_compliance']
            add8 = data[data['o_age'] == 8]['
                speed_limit_compliance']
            add9 = data[data['o_age'] == 9]['
                speed_limit_compliance']
            add10 = data[data['o_age'] == 10]['
                speed_limit_compliance']
            add11 = data[data['o_age'] == 11]['
                speed_limit_compliance']
            add12 = data[data['o_age'] == 12]['
                speed_limit_compliance']
            age5 = pd.concat([age5,add5])
            age6 = pd.concat([age6,add6])
            age7 = pd.concat([age7,add7])
            age8 = pd.concat([age8,add8])
            age9 = pd.concat([age9,add9])
            age10 = pd.concat([age10,add10])
            age11 = pd.concat([age11,add11])
            age12 = pd.concat([age12,add12])
            print filename + "2"
    if count == 11:
```

```
                print "age6, age7", stats.ttest_ind(age6,age7,
                    equal_var = False)
                print "age7, age8", stats.ttest_ind(age7,age8,
                    equal_var = False)
                print numpy.mean(age5)
                print numpy.mean(age6)
                print numpy.mean(age7)
                print numpy.mean(age8)
                print numpy.mean(age9)
                print numpy.mean(age10)
                print numpy.mean(age11)
                print numpy.mean(age12)
        count += 1
```

### A.4.16 t-test: education

```
# extract file that match tripid of TBI data

import csv,os, pandas
import pandas as pd
from scipy import stats
from itertools import islice
import math
import numpy

path = "/Users/toshiyokoo/Desktop/aaals"
count = 1
for filename in os.listdir(path):
        if count == 1:
                data = pandas.read_csv(path +'/' +filename)
                asso = data[data['o_edu'] == 'Associates degree
                    ']['speed_limit_compliance']
```

```
            some = data[data['o_edu'] == 'Some college']['
                speed_limit_compliance']
            high = data[data['o_edu'] == 'High school
                graduate']['speed_limit_compliance']
            voca= data[data['o_edu'] == 'Vocational/
                Technical training']['speed_limit_compliance
                ']
            bach = data[data['o_edu'] == 'Bachelors degree
                ']['speed_limit_compliance']
            grad = data[data['o_edu'] == 'Graduate/Post-
                graduate degree']['speed_limit_compliance']
            print filename + "1"
    else:
            data = pandas.read_csv(path +'/' +filename)
            add0 = data[data['o_edu'] == 'Associates degree
                ']['speed_limit_compliance']
            add1 = data[data['o_edu'] == 'Some college']['
                speed_limit_compliance']
            add2 = data[data['o_edu'] == 'High school
                graduate']['speed_limit_compliance']
            add3 = data[data['o_edu'] == 'Vocational/
                Technical training']['speed_limit_compliance
                ']
            add4 = data[data['o_edu'] == 'Bachelors degree
                ']['speed_limit_compliance']
            add5 = data[data['o_edu'] == 'Graduate/Post-
                graduate degree']['speed_limit_compliance']
            asso = pd.concat([asso,add0])
            some = pd.concat([some,add1])
            high = pd.concat([high,add2])
            voca = pd.concat([voca,add3])
            bach = pd.concat([bach,add4])
```

```
                grad = pd.concat([grad,add5])
                print filename + "2"
        if count == 11:
                print "high, some", stats.ttest_ind(high,some,
                    equal_var = False)
                print "grad, some", stats.ttest_ind(grad,some,
                    equal_var = False)
                print numpy.mean(asso)
                print numpy.mean(bach)
                print numpy.mean(grad)
                print numpy.mean(high)
                print numpy.mean(some)
                print numpy.mean(voca)
        count += 1
```

### A.4.17 Speeding difference depending on driver

```
#inputfile is from csv file after the algorithm 10_removeSL0km

import csv,os, numpy, math
from itertools import islice

title = ['filename', 'tripid', 'date', 'time', 'speed(kilometer
    )', 'streetlength', 'speedlimit(mph)', 'speedlimit(km)', '
    speed_limit_compliance', 'speed2']

path = "/Users/toshiyokoo/Desktop/aaahu"
newname = 99
for filename in os.listdir(path):
        inputFile = open(path +'/' +filename,'rU')
        root_path = '/Users/toshiyokoo/Desktop/14_result'
        outputFile = open(os.path.join(root_path, filename
            [0:12] + '.csv'), 'a')
```

```
writer = csv.writer(outputFile, lineterminator='\n')
Filename = filename[:-4]
trip = filename[0:6] + filename[7:9] + filename[10:12]
tripid = int(trip)
list = []
if tripid == newname:
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                speed_km = str(row[2])
                date = str(row[3])
                time = str(row[4])
                street_length = str(row[9])
                speedlimit_mph = str(row[11])
                drive_speed = float(row[2])
                #convert the unit of speedlimit from
                   mile to kilometer
                mile = float(row[11])
                conv_fac = 1.609344
                speedlimit_km = mile * conv_fac

                #calculate speed limit compliance
                speed_limit_compliance = drive_speed /
                   speedlimit_km

                #whether the speeding or not
                if (drive_speed - speedlimit_km) > 0:
                        speed2 = 1
                else:
                        speed2 = 0
```

```
                list.extend([Filename, trip, date, time
                    , speed_km, street_length,
                    speedlimit_mph, speedlimit_km,
                    speed_limit_compliance, speed2])
                writer.writerow(list)
                list = []

            newname = tripid
            inputFile.close()
            outputFile.close()
            print filename, "1"

    else:
            print >> outputFile, ','.join(title)
            reader = csv.reader(inputFile)
            inputFile.seek(0)
            for row in islice(reader,1,None):
                    speed_km = str(row[2])
                    date = str(row[3])
                    time = str(row[4])
                    street_length = str(row[9])
                    speedlimit_mph = str(row[11])
                    drive_speed = float(row[2])
                    #convert the unit of speedlimit from
                        mile to kilometer
                    mile = float(row[11])
                    conv_fac = 1.609344
                    speedlimit_km = mile * conv_fac

                    #calculate speed limit compliance
                    speed_limit_compliance = drive_speed /
                        speedlimit_km
```

```
                        #whether the speeding or not
                        if (drive_speed - speedlimit_km) > 0:
                                speed2 = 1
                        else:
                                speed2 = 0

                        list.extend([Filename, trip, date, time
                            , speed_km, street_length,
                            speedlimit_mph, speedlimit_km,
                            speed_limit_compliance, speed2])
                        writer.writerow(list)
                        list = []

                newname = tripid
                inputFile.close()
                outputFile.close()
                print filename, "2"


import csv,os
from itertools import islice

title = ['tripid', 'total_time', 'total_speeding', 'percentage
    ', 'gender', 'education', 'age']
def howmuch():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        count = 0
        num_speed = 0
        list = []
        for row in islice(reader,1,None):
```

```
            count += 1
            tripid = int(row[1])
            speeding = int(row[9])
            if speeding == 1:
                    num_speed +=1

            if count == 1:
                    #Input Household survey data
                    #Match Tripid between GPS file and
                       Household survey data
                    TBIFile = open("/Users/toshiyokoo/
                       Desktop/5133/TBIdata.csv",'rU')
                    idlist = csv.reader(TBIFile)
                    TBIFile.seek(0)
                    for dem in islice(idlist, 1, None):
                            if int(dem[3]) == int(tripid):
                                    o_gender = dem[4]
                                    o_education = dem[5]
                                    o_age = dem[6]

    time = count
    speed = num_speed
    total_time = float(time)
    total_speeding = float(speed)

    #percentage of speeding depends on individuals
    percentage = float(total_speeding / total_time)

    list.extend([tripid, total_time, total_speeding,
       percentage, o_gender, o_education, o_age])
    writer.writerow(list)
    list = []
```

```
        inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaaie"
root_path = '/Users/toshiyokoo/Desktop/15_result'
outputFile = open(os.path.join(root_path,"summary_percentage")
   +'.csv','a')
writer = csv.writer(outputFile, lineterminator='\n')
print >> outputFile, ','.join(title)
for filename in os.listdir(path):
        global filename
        howmuch()
outputFile.close()
```

### A.4.18 Fixed effects estimation with integrated GPS points

```
#load GPS
#--- 1 Set pathname here
InDrPth1='Users/toshiyokoo/Desktop/aaadq/'

#--- 2  Set pathname here
InDrPth2='Users/toshiyokoo/Desktop/aaadr/'

#execute algorithm_sub0
execfile('Users/toshiyokoo/Desktop/Matching/sub0_matching.py')

import csv,os, numpy, math
from itertools import islice
import processing
import time

def matching():
        #load GIS map
```

```
#--- 1 Load map layer
InGIS = "/Users/toshiyokoo/Desktop/Buffer0318/10
    m_buffer.shp"
vlayer = QgsVectorLayer(InGIS, "10m_buffer", "ogr")

#--- 2 Confirm something is loaded and valid
vlayer.isValid()

#--- 3 Display the layer into QGIS
QgsMapLayerRegistry.instance().addMapLayer(vlayer)

#Part1
#load GPS data
#--- 2 Set file name here
InFlnm1=filename
#--- 3 Build file name an path for uri
InFlPth1="file:///"+InDrPth1+InFlnm1
#gps_File = open(InDrPth1 +InFlnm1,'rU')
#gps_File.seek(0)
#--- 4 Set import Sting here note only need to set x
    and y other come for free
uri1 = InFlPth1+"?delimiter=%s&xField=%s&yField=%s" %
    (",", "GX", "GY")
# "Longitude", "Latitude" -> "GX", "GY"
#--- 5 Load point layer
bh1 = QgsVectorLayer(uri1, InFlnm1, "delimitedtext")
#--- 6 Confirm something is loaded and valid
bh1.isValid()
#--- 7 Set CRS
bh1.setCrs(QgsCoordinateReferenceSystem(26915,
    QgsCoordinateReferenceSystem.EpsgCrsId))
#4326: WGS84, 26915: NAD83/UTM zone 15N
```

```
#—— 8 Display the layer into QGIS
QgsMapLayerRegistry.instance().addMapLayer(bh1)
#—— select by location
processing.runalg("qgis:selectbylocation", vlayer, uri1
    , 0)
#——save vector layer as csv (True: selected, False:
    all)
cLayer = qgis.utils.iface.mapCanvas().layers()[1]
gi1 = 'GIS1_'+InFlnm1
gis1 = 'Users/toshiyokoo/Desktop/5536/'+gi1
QgsVectorFileWriter.writeAsVectorFormat(cLayer, gis1, "
    CP1250", None, "CSV", True)
global InDrPth2, gis1, InFlnm1
#execute algorithm_sub1
#remove intersection & missing link GPS data
execfile('Users/toshiyokoo/Desktop/Matching/
    sub1_remove_intersect.py')
#Part2
#load GPS data
#—— 1 Set file name here
InFlnm2=InFlnm1
#—— 2  Set pathname here
#InDrPth2='Users/toshiyokoo/Desktop/5880/'
#—— 3 Build file name an path for uri
InFlPth2="file:///"+InDrPth2+InFlnm2
#—— 4 Set import Sting here note only need to set x
    and y other come for free
uri2 = InFlPth2+"?delimiter=%s&xField=%s&yField=%s" %
    (",", "GX", "GY")
# "Longitude", "Latitude" -> "GX", "GY"
#—— 5 Load point layer
bh2 = QgsVectorLayer(uri2, InFlnm2, "delimitedtext")
```

```
#--- 6 Confirm something is loaded and valid
bh2.isValid()
#--- 7 Set CRS
bh2.setCrs(QgsCoordinateReferenceSystem(26915,
    QgsCoordinateReferenceSystem.EpsgCrsId))
#4326: WGS84, 26915: NAD83/UTM zone 15N
#--- 8 Display the layer into QGIS
QgsMapLayerRegistry.instance().addMapLayer(bh2)
#--- select by location
processing.runalg("qgis:selectbylocation", vlayer, uri2
    , 0)
#----save vector layer as csv (True: selected, False:
    all)
cLayer = qgis.utils.iface.mapCanvas().layers()[2]
gi2 = 'GIS2_'+filename
gis2 = 'Users/toshiyokoo/Desktop/5536/'+gi2
global gis2, InFlnm2
QgsVectorFileWriter.writeAsVectorFormat(cLayer, gis2, "
    CP1250", None, "CSV", True)
#execute algorithm_sub2
# find nearlestlink between GPS point and GIS map
# calculate minimum height
# link nearlest(Streetname and Speedlimit)
execfile('Users/toshiyokoo/Desktop/Matching/
    sub2_matching.py')
#remove GPS layer
QgsMapLayerRegistry.instance().removeMapLayer( bh1.id()
    )
QgsMapLayerRegistry.instance().removeMapLayer( bh2.id()
    )
QgsMapLayerRegistry.instance().removeMapLayer( vlayer.
    id() )
```

```
        #reset selected layer
        #cLayer.setSelectedFeatures([0])
        #--- 1 Set file name here
        InFlnm3=InFlnm2
        InDrPth3='/Users/toshiyokoo/Desktop/box/'+InFlnm3
        #move file
        os.rename(gis1, "/Users/toshiyokoo/Desktop/comp1/"+gi1)
        os.rename(gis2, "/Users/toshiyokoo/Desktop/comp1/"+gi2)
        os.rename(InDrPth3 , "/Users/toshiyokoo/Desktop/comp1
            /"+InFlnm3)
        #gps_File.close()

for filename in os.listdir(InDrPth1):
        #global filename, InFlnm2, gi1, gi2, gis1, gis2
        matching()

# remove intersection & missing link GPS data

import csv,os, numpy, math
from itertools import islice
import time

#start = time.time()

global title1
title1 = ['Longitude','Latitude','Speed(kilometer)','Course(
    degrees)','NumberOfSatellites','HDOP','Altitude(meters)','DD
    /MM/YY','HH:MM:SS','Distance(meters)','GX','GY']

def intersect():
        gps_inputFile1 = open(InDrPth1 +InFlnm1,'rU')
        gps_reader1 = csv.reader(gps_inputFile1)
```

```
gps_inputFile1.seek(0)
root_path1 = InDrPth2
outputFile1 = open(os.path.join(root_path1,InFlnm1
    [:-4]+'.csv'),'a')
writer1 = csv.writer(outputFile1, lineterminator='\n')
print >> outputFile1, ','.join(title1)
gis_inputFile1 = open(gis1,'rU')
gis_reader1 = csv.reader(gis_inputFile1)
min_height1 = 99 #dummy
speedlimit1 = 99          #dummy
for row1 in islice(gps_reader1, 1, None):
        list1 = []
        x_gps1 = float(row1[10])
        y_gps1 = float(row1[11])
        gis_inputFile1.seek(0)
        count1 = 0
        for j in islice(gis_reader1, 1, None):
                direction1 = float(j[42])
                ox_gis1 = float(j[43]) #OX
                oy_gis1 = float(j[44]) #OY
                dx_gis1 = float(j[45]) #DX
                dy_gis1 = float(j[46]) #DY
                if direction1 == 1:
                        if (ox_gis1 - 10<= x_gps1 <=
                            dx_gis1 + 10):
                                if (oy_gis1 - 10) <=
                                    y_gps1 <= (dy_gis1 +
                                     10):
                                        count1 += 1
                                elif (dy_gis1 - 10) <=
                                    y_gps1 <= (oy_gis1 +
                                     10):
```

```
                                        count1 += 1
                                elif (dx_gis1 - 10 <= x_gps1 <=
                                    ox_gis1 + 10):
                                        if (oy_gis1 - 10) <=
                                            y_gps1 <= (dy_gis1 +
                                             10):
                                                count1 += 1
                                        elif (dy_gis1 - 10) <=
                                            y_gps1 <= (oy_gis1 +
                                             10):
                                                count1 += 1

                        if count1 == 1:
                                print >> outputFile1, ','.join(row1)

                gis_inputFile1.close()
                gps_inputFile1.close()
                print InFlnm1
                #elapsed_time = time.time() - start
                #print("elapsed_time:{0}".format(elapsed_time))

#global filename, InDrPth2, InFlnm1, InFlnm2, gi1, gi2, gis1,
    gis2
intersect()

# find nearlestlink between GPS point and GIS map
# calculate minimum height
# link nearlest(Streetname and Speedlimit)

import csv,os, numpy, math
from itertools import islice
import time
```

```
#start = time.time()

global title2
title2 = ['Longitude','Latitude','Speed(kilometer)','DD/MM/YY
    ','HH:MM:SS','GX','GY','Min_hight','Streetname',  '
    Streetlength',  'Roadtype','Speedlimit(mph)',  'LinkID']

def nearlink():
        gps_inputFile = open(InDrPth2 +InFlnm2,'rU')
        gps_reader = csv.reader(gps_inputFile)
        gps_inputFile.seek(0)
        root_path = '/Users/toshiyokoo/Desktop/box'
        outputFile = open(os.path.join(root_path,InFlnm2
            [:-4]+'.csv'),'a')
        writer = csv.writer(outputFile, lineterminator='\n')
        print >> outputFile, ','.join(title2)
        gis_inputFile = open(gis2,'rU')
        gis_reader = csv.reader(gis_inputFile)
        min_height = 99 #dummy
        speedlimit = 99 #dummy
        for row in islice(gps_reader, 1, None):
                list = []
                x_gps = float(row[10])
                y_gps = float(row[11])
                longitude = str(row[0])
                latitude = str(row[1])
                speed = str(row[2])
                date = str(row[7])
                hms = str(row[8])
                GX = str(row[10])
                GY = str(row[11])
```

```
gis_inputFile.seek(0)
for i in islice(gis_reader, 1, None):
        direction = float(i[42])
        ox_gis = float(i[43]) #OX
        oy_gis = float(i[44]) #OY
        dx_gis = float(i[45]) #DX
        dy_gis = float(i[46]) #DY
        if direction == 1:
                if (ox_gis <= x_gps <= dx_gis):
                        if (oy_gis - 30) <=
                           y_gps <= (oy_gis +
                           30):
                                length = ((
                                   ox_gis -
                                   dx_gis) ** 2
                                    + (oy_gis -
                                    dy_gis) **
                                   2) ** 0.5
                                gis_o = numpy.
                                   array([
                                   ox_gis,
                                   oy_gis])
                                gis_d = numpy.
                                   array([
                                   dx_gis,
                                   dy_gis])
                                gps = numpy.
                                   array([x_gps
                                   ,y_gps])
                                v1 = gps -
                                   gis_o
```

```
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
              height

          streetlength
              =
              str(
              i
              [0])
          streetname
              =
              str(
              i
              [5])
          roadtype
              =
              str(
              i
              [27])
```

```
                                speedlimit
                                    =
                                    str(
                                    i
                                    [31])

                                linkID
                                   =
                                    str(
                                    i
                                    [41])

                                #print
                                    '1',

                                    min_height

            elif (oy_gis <= y_gps
               <= dy_gis):        #
               road x-y & curve
                     length = ((
                        ox_gis -
                        dx_gis) ** 2
                         + (oy_gis -
                         dy_gis) **
                        2) ** 0.5
                    gis_o = numpy.
                        array([
                        ox_gis,
                        oy_gis])
```

```
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height

          streetlength
              =
             str(
             i
             [0])
```

```
                    streetname
                        =
                        str(
                        i
                        [5])
                    roadtype
                        =
                        str(
                        i
                        [27])

                    speedlimit
                        =
                        str(
                        i
                        [31])

                    linkID
                       =
                        str(
                        i
                        [41])

            elif (dy_gis <= y_gps
               <= oy_gis):       #
                road x-y & curve
                     length = ((
                        ox_gis -
                        dx_gis) ** 2
                         + (oy_gis -
                         dy_gis) **
                        2) ** 0.5
```

```
gis_o = numpy.
    array([
    ox_gis,
    oy_gis])
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height
```

```
                streetlength
                    =
                    str(
                    i
                    [0])
                streetname
                    =
                    str(
                    i
                    [5])
                roadtype
                    =
                    str(
                    i
                    [27])

                speedlimit
                    =
                    str(
                    i
                    [31])

                linkID
                   =
                    str(
                    i
                    [41])

        elif (dx_gis <= x_gps <= ox_gis
            ):
```

```
if (oy_gis - 30) <=
    y_gps <= (oy_gis +
    30):
        length = ((
            ox_gis -
            dx_gis) ** 2
             + (oy_gis -
             dy_gis) **
            2) ** 0.5
        gis_o = numpy.
            array([
            ox_gis,
            oy_gis])
        gis_d = numpy.
            array([
            dx_gis,
            dy_gis])
        gps = numpy.
            array([x_gps
            ,y_gps])
        v1 = gps -
            gis_o
        v2 = gis_d -
            gis_o
        Area = numpy.
            cross(v1, v2
            )
        D = abs(Area)
        height = D /
            length
```

```
if
    min_height >
     height:
          min_height
              =
              height

          streetlength
              =
              str(
              i
              [0])
          streetname
              =
              str(
              i
              [5])
          roadtype
              =
              str(
              i
              [27])

          speedlimit
              =
              str(
              i
              [31])
```

```
                    linkID
                        =
                        str(
                        i
                        [41])

                    #print
                        '1',

                        min_height

elif (oy_gis <= y_gps
    <= dy_gis):         #
    road x-y & curve
            length = ((
                ox_gis -
                dx_gis) ** 2
                 + (oy_gis -
                 dy_gis) **
                2) ** 0.5
            gis_o = numpy.
                array([
                ox_gis,
                oy_gis])
            gis_d = numpy.
                array([
                dx_gis,
                dy_gis])
            gps = numpy.
                array([x_gps
                ,y_gps])
```

```
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height

          streetlength
              =
             str(
             i
             [0])
          streetname
              =
             str(
             i
             [5])
```

```
                roadtype
                    =
                    str(
                    i
                    [27])

                speedlimit
                    =
                    str(
                    i
                    [31])

                linkID
                    =
                    str(
                    i
                    [41])

        elif (dy_gis <= y_gps
            <= oy_gis):       #
            road x-y & curve
                length = ((
                    ox_gis -
                    dx_gis) ** 2
                     + (oy_gis -
                     dy_gis) **
                    2) ** 0.5
                gis_o = numpy.
                    array([
                    ox_gis,
                    oy_gis])
```

```
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
          min_height
              =
             height

          streetlength
              =
             str(
             i
             [0])
```

```
                                    streetname
                                        =
                                        str(
                                        i
                                        [5])
                                    roadtype
                                        =
                                        str(
                                        i
                                        [27])

                                    speedlimit
                                        =
                                        str(
                                        i
                                        [31])

                                    linkID
                                       =
                                        str(
                                        i
                                        [41])

            elif (oy_gis <= y_gps <= dy_gis
                ):
                    if (ox_gis - 30) <=
                        x_gps <= (ox_gis +
                        30):     #road y-
                        axis, interstate
                        range 50m
```

```
length = ((
    ox_gis -
    dx_gis) ** 2
     + (oy_gis -
     dy_gis) **
    2) ** 0.5
gis_o = numpy.
    array([
    ox_gis,
    oy_gis])
gis_d = numpy.
    array([
    dx_gis,
    dy_gis])
gps = numpy.
    array([x_gps
    ,y_gps])
v1 = gps -
    gis_o
v2 = gis_d -
    gis_o
Area = numpy.
    cross(v1, v2
    )
D = abs(Area)
height = D /
    length
if
    min_height >
     height:
```

```
min_height
    =
    height

streetlength
    =
    str(
    i
    [0])
streetname
    =
    str(
    i
    [5])
roadtype
    =
    str(
    i
    [27])

speedlimit
    =
    str(
    i
    [31])

linkID
   =
    str(
    i
    [41])
```

```
elif (dy_gis <= y_gps <= oy_gis
    ):
        if (ox_gis - 30) <=
            x_gps <= (ox_gis +
            30):      #road y-
            axis, interstate
            range 50m
                length = ((
                    ox_gis -
                    dx_gis) ** 2
                     + (oy_gis -
                     dy_gis) **
                    2) ** 0.5
                gis_o = numpy.
                    array([
                    ox_gis,
                    oy_gis])
                gis_d = numpy.
                    array([
                    dx_gis,
                    dy_gis])
                gps = numpy.
                    array([x_gps
                    ,y_gps])
                v1 = gps -
                    gis_o
                v2 = gis_d -
                    gis_o
                Area = numpy.
                    cross(v1, v2
                    )
                D = abs(Area)
```

```
height = D /
    length
if
    min_height >
     height:
          min_height
              =
              height

          streetlength
              =
              str(
              i
              [0])
          streetname
              =
              str(
              i
              [5])
          roadtype
              =
              str(
              i
              [27])

          speedlimit
              =
              str(
              i
              [31])
```

```
                                                    linkID
                                                        =
                                                        str(
                                                        i
                                                        [41])

                        else:
                                pass

                if min_height < 30:
                        #streetlength = 99
                        #streetname = 99         #dummy
                        #roadtype = 99           #dummy
                        #speedlimit = 99         #dummy
                        Min_height = str(min_height)
                        Streetname = str(streetname)
                        Streetlength = str(streetlength)
                        Roadtype = str(roadtype)
                        Speedlimit = str(speedlimit)
                        LinkID = str(linkID)
                        list.extend([longitude, latitude, speed
                            , date, hms, GX, GY, Min_height,
                            Streetname, Streetlength, Roadtype,
                            Speedlimit, LinkID])
                        writer.writerow(list)
                        #print >> outputFile, ','.join(row)
                        min_height = 99 #dummy
                        speedlimit = 99 #dummy
gis_inputFile.close()
gps_inputFile.close()

print InFlnm2
```

```
        #elapsed_time = time.time() - start
        #print("elapsed_time:{0}".format(elapsed_time))

#global filename, InDrPth2, InFlnm2, gi1, gi2, gis1, gis2
nearlink()

# extract file that match tripid of TBI data

import csv,os
from itertools import islice
import math

title = ['Longitude','Latitude','Speed(kilometer)','DD/MM/YY','
   HH:MM:SS','GX','GY','Min_hight','Streetname', 'Streetlength
   ', 'Roadtype','Speedlimit(mph)', 'LinkID']

def remove():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        tripid = filename[0:6] + filename[7:9] + filename
           [10:12]
        TBIFile = open("/Users/toshiyokoo/Desktop/5133/
           TBIdata_age.csv",'rU')
        idlist = csv.reader(TBIFile)
        TBIFile.seek(0)
        id = 0
        for row in islice(idlist, 1, None):
                if int(row[3]) == int(tripid):
                        root_error = '/Users/toshiyokoo/Desktop
                           /09_result_error'
```

```
                        outputFile_error = open(os.path.join(
                            root_error, filename[:-4] + '.csv'),
                             'a')
                        print >> outputFile_error, ','.join(
                            title)
                        inputFile.seek(0)
                        for row in islice(reader,1,None):
                                print >> outputFile_error, ','.
                                    join(row)
                        outputFile_error.close()
                        print filename[:-4], "age_young"
                        id = 1

        if id == 0:
                root_path = '/Users/toshiyokoo/Desktop/09
                    _result'
                outputFile = open(os.path.join(root_path,
                    filename[:-4] + '.csv'), 'a')
                print >> outputFile, ','.join(title)
                inputFile.seek(0)
                for row in islice(reader,1,None):
                        print >> outputFile, ','.join(row)
                outputFile.close()
                print filename[:-4], "no_error"
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaaef"
for filename in os.listdir(path):
        global filename
        remove()

# Remove data whose speed limit is 0 km/h.
```

```
import csv,os
from itertools import islice

title = ['Longitude','Latitude','Speed(kilometer)','DD/MM/YY','
    HH:MM:SS','GX','GY','Min_hight','Streetname', 'Streetlength
    ', 'Roadtype','Speedlimit(mph)', 'LinkID']
def removespeed():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root1_path = "/Users/toshiyokoo/Desktop/10_result"
        root2_path = "/Users/toshiyokoo/Desktop/10_result_error
            "
        count = 0
        all_count = 0
        for row in islice(reader,1,None):
                if int(row[11]) > 0:
                        count += 1
                        all_count += 1
                else:
                        all_count += 1

        inputFile.seek(0)
        if count == 0:
                print filename[:-4] + 'is empty'
                outputFile = open(os.path.join(root2_path,
                    filename[:-4]+'.csv'),'a')
                writer = csv.writer(outputFile)
                print >> outputFile, ','.join(title)
                for row in islice(reader,1,None):
                        print >> outputFile, ','.join(row)
                outputFile.close()
```

```
        elif count == all_count:
                print filename[:-4] + 'is full'
                outputFile = open(os.path.join(root1_path,
                    filename[:-4]+'.csv'),'a')
                writer = csv.writer(outputFile)
                print >> outputFile, ','.join(title)
                for row in islice(reader,1,None):
                        print >> outputFile, ','.join(row)
                outputFile.close()
        elif 0 < count < all_count:
                print filename[:-4] + 'is modified'
                outputFile = open(os.path.join(root1_path,
                    filename[:-4]+'.csv'),'a')
                writer = csv.writer(outputFile)
                print >> outputFile, ','.join(title)
                for row in islice(reader,1,None):
                        if int(row[11]) > 0:
                                print >> outputFile, ','.join(
                                    row)
                outputFile.close()
        inputFile.close()

path = "/Users/toshiyokoo/Desktop/aaaeg"
for filename in os.listdir(path):
        global filename
        removespeed()

import csv,os, numpy, math
from itertools import islice
import datetime
```

```
title = ['PersonID', 'TripID', 'LinkID', 'Longitude','Latitude
    ','Speed(kilometer)','DD/MM/YY','HH:MM:SS','GX','GY', '
    Streetname', 'Streetlength', 'Speedlimit(mph)', 'Speedlimit(
    km/h)', 'speeding']

def addinfo():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root_path = '/Users/toshiyokoo/Desktop/11_result'
        outputFile = open(os.path.join(root_path, filename[:-4]
            + '.csv'), 'a')
        writer = csv.writer(outputFile, lineterminator='\n')
        print >> outputFile, ','.join(title)
        for row in islice(reader, 1, None):
                list = []
                Filename = filename[:-4]
                personid = filename[0:6] + filename[7:9] +
                    filename[10:12]
                longitude = str(row[0])
                latitude = str(row[1])
                speed = str(row[2])
                date = str(row[3])
                hms = str(row[4])
                GX = str(row[5])
                GY = str(row[6])
                streetname = str(row[8])
                streetlength = str(row[9])
                speedlimit = str(row[11])
                linkID = str(row[12])
                drive_speed = float(row[2])
```

```
                #convert the unit of speedlimit from mile to
                    kilometer
                mile = float(row[11])
                conv_fac = 1.609344
                kilometer = mile * conv_fac

                #whether the speeding or not
                if (drive_speed - kilometer) > 0:
                        speed2 = 1
                else:
                        speed2 = 0

                list.extend([personid, Filename, linkID,
                    longitude, latitude, speed, date, hms, GX,
                    GY, streetname, streetlength, speedlimit,
                    kilometer, speed2])
                writer.writerow(list)
        inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaaei"
for filename in os.listdir(path):
        global filename
        addinfo()

#integrate multiple GPS points of one link into one GPS point

import csv,os, numpy, math
from itertools import islice
import datetime
```

```
title = ['PersonID', 'Tripid', 'LinkID', 'streetname', 'hour',
    'streetlength', 'Speedlimit', 'Sum_drivespeed(kilometer)', '
    sum_speedlimit(kilometer)', 'sum_speed', 'sum_time', 'degree
    ','percentage']

def calculate():
        link_new = 999999 #Errornumber
        count = 0
        all_count = 0
        time = 0
        drive = 0
        speedlimit = 0
        speeding = 0
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader, 1, None):
                all_count += 1

        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader, 1, None):
                root_path = '/Users/toshiyokoo/Desktop/12
                    _result'
                outputFile = open(os.path.join(root_path,
                    filename[:-4] + '.csv'), 'a')
                writer = csv.writer(outputFile, lineterminator
                    ='\n')
                Tripname = filename[:-4]
                personid = filename[0:6] + filename[7:9] +
                    filename[10:12]
                date = str(row[6])
```

```
hms = str(row[7])
streetname = str(row[10])
streetlength = str(row[11])
linkID = str(row[2])
speedmph = str(row[12])
list = []

if link_new == 999999:
        print >> outputFile, ','.join(title)
        time += 1
        drive += float(row[5]) #drive_speed(
            kilometer)
        speedlimit += float(row[13]) #
            speedlimit(kilometer)
        speeding += float(row[14]) #whether
            speeding or not
        link_new = linkID
        old_date = date
        old_hms = hms
        old_streetname = streetname
        old_streetlength = streetlength
        old_linkID = linkID
        old_speedlimit = speedmph
        count += 1
        continue

elif link_new == linkID:
        time += 1
        drive += float(row[5])
        speedlimit += float(row[13])
        speeding += float(row[14])
        #old_streetname = streetname
```

```
#old_streetlength = streetlength
#old_linkID = linkID
count += 1
if count == all_count:
        sum_drive = drive
        sum_speedlimit = speedlimit
        sum_speed = speeding
        sum_time = time
        degree = sum_drive /
            sum_speedlimit
        percentage = sum_speed /
            sum_time
        if old_date[-4:] == '2011':
                d = datetime.datetime.
                    strptime(old_date,
                    '%d/%m/%Y')
        else:
                d = datetime.datetime.
                    strptime(old_date,
                    '%d/%m/%y')

        #change the hour from UTC to MN
             (Time in GPS is Greenwich
            Mean Time (GMT))
        f = datetime.datetime.strptime(
            old_hms, '%H:%M:%S')
        hour = f.hour
        date1 = datetime.datetime(2011,
             3, 13)
        date2 = datetime.datetime(2011,
             11, 5)
        #Daylight Saving Time
```

```
                    if date1 <= d <= date2:
                            cal_hour = hour - 5
                            fix_hour = cal_hour
                            if cal_hour < 0:
                                    fix_hour =
                                       cal_hour +
                                       24
                    #Standard Time
                    else:
                            cal_hour = hour - 6
                            fix_hour = cal_hour
                            if cal_hour < 0:
                                    fix_hour =
                                       cal_hour +
                                       24

                    list.extend([personid, Tripname
                       , old_linkID, old_streetname
                       , fix_hour, old_streetlength
                       , old_speedlimit, sum_drive,
                        sum_speedlimit, sum_speed,
                       sum_time, degree, percentage
                       ])
                    writer.writerow(list)
                    break
            continue

    elif link_new != linkID:
            sum_drive = drive
            sum_speedlimit = speedlimit
            sum_speed = speeding
            sum_time = time
```

```
degree = sum_drive / sum_speedlimit
percentage = sum_speed / sum_time
if old_date[-4:] == '2011':
        d = datetime.datetime.strptime(
            old_date, '%d/%m/%Y')
else:
        d = datetime.datetime.strptime(
            old_date, '%d/%m/%y')

#change the hour from UTC to MN (Time
    in GPS is Greenwich Mean Time (GMT))
f = datetime.datetime.strptime(old_hms,
    '%H:%M:%S')
hour = f.hour
date1 = datetime.datetime(2011, 3, 13)
date2 = datetime.datetime(2011, 11, 5)
#Daylight Saving Time
if date1 <= d <= date2:
        cal_hour = hour - 5
        fix_hour = cal_hour
        if cal_hour < 0:
                fix_hour = cal_hour +
                    24
#Standard Time
else:
        cal_hour = hour - 6
        fix_hour = cal_hour
        if cal_hour < 0:
                fix_hour = cal_hour +
                    24
```

```
list.extend([personid, Tripname,
    old_linkID, old_streetname, fix_hour
    , old_streetlength, old_speedlimit,
    sum_drive, sum_speedlimit, sum_speed
    , sum_time, degree, percentage])
writer.writerow(list)
list = []
time = 1
drive = float(row[5])
speedlimit = float(row[13])
speeding = float(row[14])
link_new = linkID
count += 1
old_date = date
old_hms = hms
old_streetname = streetname
old_streetlength = streetlength
old_linkID = linkID
old_speedlimit = speedmph
if count == all_count:
        sum_drive = drive
        sum_speedlimit = speedlimit
        sum_speed = speeding
        sum_time = time
        degree = sum_drive /
            sum_speedlimit
        percentage = sum_speed /
            sum_time
        if old_date[-4:] == '2011':
                d = datetime.datetime.
                    strptime(old_date,
                    '%d/%m/%Y')
```

```
else:
        d = datetime.datetime.
            strptime(old_date,
            '%d/%m/%y')

#change the hour from UTC to MN
    (Time in GPS is Greenwich
   Mean Time (GMT))
f = datetime.datetime.strptime(
   old_hms, '%H:%M:%S')
hour = f.hour
date1 = datetime.datetime(2011,
    3, 13)
date2 = datetime.datetime(2011,
    11, 5)
#Daylight Saving Time
if date1 <= d <= date2:
        cal_hour = hour - 5
        fix_hour = cal_hour
        if cal_hour < 0:
                fix_hour =
                   cal_hour +
                   24
#Standard Time
else:
        cal_hour = hour - 6
        fix_hour = cal_hour
        if cal_hour < 0:
                fix_hour =
                   cal_hour +
                   24
```

```
                                        list.extend([personid, Tripname
                                            , old_linkID, old_streetname
                                            , fix_hour, old_streetlength
                                            , old_speedlimit, sum_drive,
                                             sum_speedlimit, sum_speed,
                                            sum_time, degree, percentage
                                            ])
                                        writer.writerow(list)
                                        break

        print filename
        inputFile.close()
        outputFile.close()

path = "/Users/toshiyokoo/Desktop/aaafh"
for filename in os.listdir(path):
        global filename
        calculate()

import csv,os
from itertools import islice

title = ['PersonID', 'Tripid', 'LinkID', 'streetname', 'hour',
    'streetlength', 'Speedlimit', 'Sum_drivespeed(kilometer)', '
    sum_speedlimit(kilometer)', 'sum_speed', 'sum_time', 'degree
    ','percentage']
def compile():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        for row in islice(reader,1,None):
                print >> outputFile, ','.join(row)
```

```
        inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaafi"
root_path = "/Users/toshiyokoo/Desktop/13_result"
outputFile = open(os.path.join(root_path,"summary_13")+'.csv','
    a')
writer = csv.writer(outputFile)
print >> outputFile, ','.join(title)
for filename in os.listdir(path):
        global filename
        compile()
outputFile.close()

import csv,os, numpy, math
from itertools import islice
import datetime

title = ['PersonID', 'TripID', 'LinkID', 'Hour', 'streetlength
    ', 'Speedlimit', 'degree', 'percentage', 'speed_25', '
    speed_30', 'speed_35', 'speed_40', 'speed_45', 'speed_50', '
    speed_55', 'speed_60', 'speed_65', 'morning', 'afternoon', '
    night', 'o_gender', 'o_edu', 'o_age', 'gender', 'education_1
    ', 'education_2', 'age_25_to_34', 'age_35_to_44', '
    age_45_to_54', 'age_55_to_64', 'age_65_to_74', 'age_75_to_84
    ', 'age_85_and_over']

def summarize():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root_path = '/Users/toshiyokoo/Desktop/14_result'
```

```
outputFile = open(os.path.join(root_path,"summary_14")
    +'.csv','a')
writer = csv.writer(outputFile, lineterminator='\n')
print >> outputFile, ','.join(title)
for row in islice(reader, 1, None):
        list = []
        personid = str(row[0])
        tripid = str(row[1])
        linkid = str(row[2])
        hour = str(row[4])
        streetlength = str(row[5])
        speedlimit = str(row[6])
        degree = str(row[11])
        percentage = str(row[12])

        #Dummy variable (Speed limit)
        num_speedlimit = int(row[6])
        speed_25 = 0
        speed_30 = 0
        speed_35 = 0
        speed_40 = 0
        speed_45 = 0
        speed_50 = 0
        speed_55 = 0
        speed_60 = 0
        speed_65 = 0
        if num_speedlimit <= 25:
                speed_25 = 1
        elif num_speedlimit == 30:
                speed_30 = 1
        elif num_speedlimit == 35:
                speed_35 = 1
```

```
elif num_speedlimit == 40:
        speed_40 = 1
elif num_speedlimit == 45:
        speed_45 = 1
elif num_speedlimit == 50:
        speed_50 = 1
elif num_speedlimit == 55:
        speed_55 = 1
elif num_speedlimit == 60:
        speed_60 = 1
elif num_speedlimit == 65:
        speed_65 = 1
elif num_speedlimit == 70:
        dummy = 0
else:
        speed_25 = 99
        speed_30 = 99
        speed_35 = 99
        speed_40 = 99
        speed_45 = 99
        speed_50 = 99
        speed_55 = 99
        speed_60 = 99
        speed_65 = 99

#Dummy variable (morning, afternoon, night)
morning = 0
afternoon = 0
night = 0
dummy_hour = int(hour)
if 6 <= dummy_hour < 12:
        morning = 1
```

```
elif 12 <= dummy_hour < 18:
        afternoon = 1
elif 18 <= dummy_hour < 24:
        night = 1
elif 0 <= dummy_hour < 6:
        dummy = 0
else:
        morning = 99
        afternoon = 99
        night = 99

#Input Household survey data
#Match Tripid between GPS file and Household
   survey data
TBIFile = open("/Users/toshiyokoo/Desktop/5133/
   TBIdata.csv",'rU')
idlist = csv.reader(TBIFile)
TBIFile.seek(0)
gender = 0
age_25_to_34 = 0
age_35_to_44 = 0
age_45_to_54 = 0
age_55_to_64 = 0
age_65_to_74 = 0
age_75_to_84 = 0
age_85_and_over = 0
education_1 = 0
education_2 = 0
for dem in islice(idlist, 1, None):
        if int(dem[3]) == int(personid):
#Dummy variable (Gender)
                o_gender = dem[4]
```

```
                    if dem[4] == 'Male':
                            gender = 1
                    elif dem[4] == 'Female':
                            gender = 0
                    else:
                            gender = 99
#Dummy variable (Education)
                    o_education = dem[5]
                    if dem[5] == 'Daycare / Pre-
                       school':
                            dummy = 0
                    elif dem[5] == 'Less than high
                       school':
                            dummy = 0
                    elif dem[5] == 'High school
                       graduate':
                            dummy = 0
                    elif dem[5] == 'Some college':
                            education_1 = 1
                    elif dem[5] == 'Vocational/
                       Technical training':
                            dummy = 0
                    elif dem[5] == 'Associates
                       degree':
                            education_1 = 1
                    elif dem[5] == 'Bachelors
                       degree':
                            education_2 = 1
                    elif dem[5] == 'Graduate/Post-
                       graduate degree':
                            education_2 = 1
                    else:
```

```
                    education_1 = 99
                    education_2 = 99
#Dummy variable (Age)
                o_age = dem[6]
                if      int(dem[6]) == 6:
                    age_25_to_34 = 1
                elif    int(dem[6]) == 7:
                    age_35_to_44 = 1
                elif    int(dem[6]) == 8:
                    age_45_to_54 = 1
                elif    int(dem[6]) == 9:
                    age_55_to_64 = 1
                elif    int(dem[6]) == 10:
                    age_65_to_74 = 1
                elif    int(dem[6]) == 11:
                    age_75_to_84 = 1
                elif    int(dem[6]) == 12:
                    age_85_and_over = 1
                elif    int(dem[6]) == 5:
                    dummy = 0
                else:
                    age_25_to_34 = 99
                    age_35_to_44 = 99
                    age_45_to_54 = 99
                    age_55_to_64 = 99
                    age_65_to_74 = 99
                    age_75_to_84 = 99
                    age_85_and_over = 99
```

```
                list.extend([personid, tripid, linkid, hour,
                    streetlength, speedlimit, degree, percentage
                    , speed_25, speed_30, speed_35, speed_40,
                    speed_45, speed_50, speed_55, speed_60,
                    speed_65, morning, afternoon, night,
                    o_gender, o_education, o_age, gender,
                    education_1, education_2, age_25_to_34,
                    age_35_to_44, age_45_to_54, age_55_to_64,
                    age_65_to_74, age_75_to_84, age_85_and_over
                    ])
                writer.writerow(list)
        inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaafj"
for filename in os.listdir(path):
        global filename
        summarize()

import csv,os, numpy, math
from itertools import islice
import datetime
```

```
title = ['PersonID', 'TripID', 'LinkID', 'Hour', 'streetlength
    ', 'Speedlimit', 'degree', 'percentage', 'speed_25', '
    speed_30', 'speed_35', 'speed_40', 'speed_45', 'speed_50', '
    speed_55', 'speed_60', 'speed_65', 'morning', 'afternoon', '
    night', 'o_gender', 'o_edu', 'o_age', 'gender', 'education_1
    ', 'education_2', 'age_25_to_34', 'age_35_to_44', '
    age_45_to_54', 'age_55_to_64', 'age_65_to_74', 'age_75_to_84
    ', 'age_85_and_over', 'person1', 'person2', 'person3', '
    person4', 'person5', 'person6', 'person7', 'person8', '
    person9', 'person10', 'person11', 'person12', 'person13', '
    person14', 'person15', 'person16', 'person17', 'person18', '
    person19', 'person20', 'person21', 'person22', 'person23', '
    person24', 'person25', 'person26', 'person27', 'person28', '
    person29', 'person30', 'person31', 'person32', 'person33', '
    person34', 'person35', 'person36', 'person37', 'person38', '
    person39', 'person40', 'person41', 'person42', 'person43', '
    person44', 'person45', 'person46', 'person47', 'person48', '
    person49', 'person50', 'person51', 'person52', 'person53', '
    person54', 'person55', 'person56', 'person57', 'person58', '
    person59', 'person60', 'person61', 'person62', 'person63', '
    person64', 'person65', 'person66', 'person67', 'person68', '
    person69', 'person70', 'person71', 'person72', 'person73', '
    person74', 'person75', 'person76', 'person77', 'person78', '
    person79', 'person80', 'person81', 'person82', 'person83', '
    person84', 'person85', 'person86', 'person87', 'person88', '
    person89', 'person90', 'person91', 'person92', 'person93', '
    person94', 'person95', 'person96', 'person97', 'person98', '
    person99', 'person100', 'person101', 'person102', 'person103
    ', 'person104', 'person105', 'person106', 'person107', '
    person108', 'person109', 'person110', 'person111', '
    person112', 'person113', 'person114', 'person115', '
    person116', 'person117', 'person118', 'person119', '
    person120', 'person121', 'person122', 'person123', '
    person124', 'person125', 'person126', 'person127', '
    person128', 'person129', 'person130', 'person131', '
    person132', 'person133', 'person134', 'person135', '
    person136', 'person137', 'person138', 'person139', '
    person140', 'person141', 'person142', 'person143', '
```

```
def summarize():
        inputFile = open(path +'/' +filename,'rU')
        reader = csv.reader(inputFile)
        inputFile.seek(0)
        root_path = '/Users/toshiyokoo/Desktop/15_result'
        outputFile = open(os.path.join(root_path,"summary_15")
           +'.csv','a')
        writer = csv.writer(outputFile, lineterminator='\n')
        print >> outputFile, ','.join(title)
        for row in islice(reader, 1, None):
                list = []
                personid = str(row[0])
                person = int(row[0])
                tripid = str(row[1])
                linkid = str(row[2])
                hour = str(row[3])
                streetlength = str(row[4])
                speedlimit = str(row[5])
                degree = str(row[6])
                percentage = str(row[7])
                speed_25 = str(row[8])
                speed_30 = str(row[9])
                speed_35 = str(row[10])
                speed_40 = str(row[11])
                speed_45 = str(row[12])
                speed_50 = str(row[13])
                speed_55 = str(row[14])
                speed_60 = str(row[15])
                speed_65 = str(row[16])
                morning = str(row[17])
                afternoon = str(row[18])
```

```
night = str(row[19])
o_gender = str(row[20])
o_education = str(row[21])
o_age = str(row[22])
gender = str(row[23])
education_1 = str(row[24])
education_2 = str(row[25])
age_25_to_34 = str(row[26])
age_35_to_44 = str(row[27])
age_45_to_54 = str(row[28])
age_55_to_64 = str(row[29])
age_65_to_74 = str(row[30])
age_75_to_84 = str(row[31])
age_85_and_over = str(row[32])

#person
person1 = 0
person2 = 0
person3 = 0
person4 = 0
person5 = 0
person6 = 0
person7 = 0
person8 = 0
person9 = 0
person10 = 0
person11 = 0
person12 = 0
person13 = 0
person14 = 0
person15 = 0
person16 = 0
```

$$
\begin{aligned}
\text{person17} &= 0 \\
\text{person18} &= 0 \\
\text{person19} &= 0 \\
\text{person20} &= 0 \\
\text{person21} &= 0 \\
\text{person22} &= 0 \\
\text{person23} &= 0 \\
\text{person24} &= 0 \\
\text{person25} &= 0 \\
\text{person26} &= 0 \\
\text{person27} &= 0 \\
\text{person28} &= 0 \\
\text{person29} &= 0 \\
\text{person30} &= 0 \\
\text{person31} &= 0 \\
\text{person32} &= 0 \\
\text{person33} &= 0 \\
\text{person34} &= 0 \\
\text{person35} &= 0 \\
\text{person36} &= 0 \\
\text{person37} &= 0 \\
\text{person38} &= 0 \\
\text{person39} &= 0 \\
\text{person40} &= 0 \\
\text{person41} &= 0 \\
\text{person42} &= 0 \\
\text{person43} &= 0 \\
\text{person44} &= 0 \\
\text{person45} &= 0 \\
\text{person46} &= 0 \\
\text{person47} &= 0 \\
\text{person48} &= 0
\end{aligned}
$$

$$
\begin{aligned}
\text{person49} &= 0 \\
\text{person50} &= 0 \\
\text{person51} &= 0 \\
\text{person52} &= 0 \\
\text{person53} &= 0 \\
\text{person54} &= 0 \\
\text{person55} &= 0 \\
\text{person56} &= 0 \\
\text{person57} &= 0 \\
\text{person58} &= 0 \\
\text{person59} &= 0 \\
\text{person60} &= 0 \\
\text{person61} &= 0 \\
\text{person62} &= 0 \\
\text{person63} &= 0 \\
\text{person64} &= 0 \\
\text{person65} &= 0 \\
\text{person66} &= 0 \\
\text{person67} &= 0 \\
\text{person68} &= 0 \\
\text{person69} &= 0 \\
\text{person70} &= 0 \\
\text{person71} &= 0 \\
\text{person72} &= 0 \\
\text{person73} &= 0 \\
\text{person74} &= 0 \\
\text{person75} &= 0 \\
\text{person76} &= 0 \\
\text{person77} &= 0 \\
\text{person78} &= 0 \\
\text{person79} &= 0 \\
\text{person80} &= 0
\end{aligned}
$$

person81 = 0
person82 = 0
person83 = 0
person84 = 0
person85 = 0
person86 = 0
person87 = 0
person88 = 0
person89 = 0
person90 = 0
person91 = 0
person92 = 0
person93 = 0
person94 = 0
person95 = 0
person96 = 0
person97 = 0
person98 = 0
person99 = 0
person100 = 0
person101 = 0
person102 = 0
person103 = 0
person104 = 0
person105 = 0
person106 = 0
person107 = 0
person108 = 0
person109 = 0
person110 = 0
person111 = 0
person112 = 0

person113 = 0
person114 = 0
person115 = 0
person116 = 0
person117 = 0
person118 = 0
person119 = 0
person120 = 0
person121 = 0
person122 = 0
person123 = 0
person124 = 0
person125 = 0
person126 = 0
person127 = 0
person128 = 0
person129 = 0
person130 = 0
person131 = 0
person132 = 0
person133 = 0
person134 = 0
person135 = 0
person136 = 0
person137 = 0
person138 = 0
person139 = 0
person140 = 0
person141 = 0
person142 = 0
person143 = 0
person144 = 0

```
person145 = 0
person146 = 0
person147 = 0
person148 = 0
person149 = 0
person150 = 0
person151 = 0
#person152 = 0

#if person == 1013360102:
#       person1 = 1
#elif person == 1022380102:
#       person2 = 1

#Input personID data
PersonFile = open("/Users/toshiyokoo/Desktop
   /5133/Persondata.csv",'rU')
idlist = csv.reader(PersonFile)
PersonFile.seek(0)
for dem in islice(idlist, 0, None):
        if person == int(dem[0]):
                person1 = 1
        elif person == int(dem[1]):
                person2 = 1
        elif person == int(dem[2]):
                person3 = 1
        elif person == int(dem[3]):
                person4 = 1
        elif person == int(dem[4]):
                person5 = 1
        elif person == int(dem[5]):
                person6 = 1
```

```
elif person == int(dem[6]):
        person7 = 1
elif person == int(dem[7]):
        person8 = 1
elif person == int(dem[8]):
        person9 = 1
elif person == int(dem[9]):
        person10 = 1
elif person == int(dem[10]):
        person11 = 1
elif person == int(dem[11]):
        person12 = 1
elif person == int(dem[12]):
        person13 = 1
elif person == int(dem[13]):
        person14 = 1
elif person == int(dem[14]):
        person15 = 1
elif person == int(dem[15]):
        person16 = 1
elif person == int(dem[16]):
        person17 = 1
elif person == int(dem[17]):
        person18 = 1
elif person == int(dem[18]):
        person19 = 1
elif person == int(dem[19]):
        person20 = 1
elif person == int(dem[20]):
        person21 = 1
elif person == int(dem[21]):
        person22 = 1
```

```
elif person == int(dem[22]):
        person23 = 1
elif person == int(dem[23]):
        person24 = 1
elif person == int(dem[24]):
        person25 = 1
elif person == int(dem[25]):
        person26 = 1
elif person == int(dem[26]):
        person27 = 1
elif person == int(dem[27]):
        person28 = 1
elif person == int(dem[28]):
        person29 = 1
elif person == int(dem[29]):
        person30 = 1
elif person == int(dem[30]):
        person31 = 1
elif person == int(dem[31]):
        person32 = 1
elif person == int(dem[32]):
        person33 = 1
elif person == int(dem[33]):
        person34 = 1
elif person == int(dem[34]):
        person35 = 1
elif person == int(dem[35]):
        person36 = 1
elif person == int(dem[36]):
        person37 = 1
elif person == int(dem[37]):
        person38 = 1
```

```
elif person == int(dem[38]):
        person39 = 1
elif person == int(dem[39]):
        person40 = 1
elif person == int(dem[40]):
        person41 = 1
elif person == int(dem[41]):
        person42 = 1
elif person == int(dem[42]):
        person43 = 1
elif person == int(dem[43]):
        person44 = 1
elif person == int(dem[44]):
        person45 = 1
elif person == int(dem[45]):
        person46 = 1
elif person == int(dem[46]):
        person47 = 1
elif person == int(dem[47]):
        person48 = 1
elif person == int(dem[48]):
        person49 = 1
elif person == int(dem[49]):
        person50 = 1
elif person == int(dem[50]):
        person51 = 1
elif person == int(dem[51]):
        person52 = 1
elif person == int(dem[52]):
        person53 = 1
elif person == int(dem[53]):
        person54 = 1
```

```
elif person == int(dem[54]):
        person55 = 1
elif person == int(dem[55]):
        person56 = 1
elif person == int(dem[56]):
        person57 = 1
elif person == int(dem[57]):
        person58 = 1
elif person == int(dem[58]):
        person59 = 1
elif person == int(dem[59]):
        person60 = 1
elif person == int(dem[60]):
        person61 = 1
elif person == int(dem[61]):
        person62 = 1
elif person == int(dem[62]):
        person63 = 1
elif person == int(dem[63]):
        person64 = 1
elif person == int(dem[64]):
        person65 = 1
elif person == int(dem[65]):
        person66 = 1
elif person == int(dem[66]):
        person67 = 1
elif person == int(dem[67]):
        person68 = 1
elif person == int(dem[68]):
        person69 = 1
elif person == int(dem[69]):
        person70 = 1
```

```
elif person == int(dem[70]):
        person71 = 1
elif person == int(dem[71]):
        person72 = 1
elif person == int(dem[72]):
        person73 = 1
elif person == int(dem[73]):
        person74 = 1
elif person == int(dem[74]):
        person75 = 1
elif person == int(dem[75]):
        person76 = 1
elif person == int(dem[76]):
        person77 = 1
elif person == int(dem[77]):
        person78 = 1
elif person == int(dem[78]):
        person79 = 1
elif person == int(dem[79]):
        person80 = 1
elif person == int(dem[80]):
        person81 = 1
elif person == int(dem[81]):
        person82 = 1
elif person == int(dem[82]):
        person83 = 1
elif person == int(dem[83]):
        person84 = 1
elif person == int(dem[84]):
        person85 = 1
elif person == int(dem[85]):
        person86 = 1
```

```
elif person == int(dem[86]):
        person87 = 1
elif person == int(dem[87]):
        person88 = 1
elif person == int(dem[88]):
        person89 = 1
elif person == int(dem[89]):
        person90 = 1
elif person == int(dem[90]):
        person91 = 1
elif person == int(dem[91]):
        person92 = 1
elif person == int(dem[92]):
        person93 = 1
elif person == int(dem[93]):
        person94 = 1
elif person == int(dem[94]):
        person95 = 1
elif person == int(dem[95]):
        person96 = 1
elif person == int(dem[96]):
        person97 = 1
elif person == int(dem[97]):
        person98 = 1
elif person == int(dem[98]):
        person99 = 1
elif person == int(dem[99]):
        person100 = 1
elif person == int(dem[100]):
        person101 = 1
elif person == int(dem[101]):
        person102 = 1
```

```
elif person == int(dem[102]):
        person103 = 1
elif person == int(dem[103]):
        person104 = 1
elif person == int(dem[104]):
        person105 = 1
elif person == int(dem[105]):
        person106 = 1
elif person == int(dem[106]):
        person107 = 1
elif person == int(dem[107]):
        person108 = 1
elif person == int(dem[108]):
        person109 = 1
elif person == int(dem[109]):
        person110 = 1
elif person == int(dem[110]):
        person111 = 1
elif person == int(dem[111]):
        person112 = 1
elif person == int(dem[112]):
        person113 = 1
elif person == int(dem[113]):
        person114 = 1
elif person == int(dem[114]):
        person115 = 1
elif person == int(dem[115]):
        person116 = 1
elif person == int(dem[116]):
        person117 = 1
elif person == int(dem[117]):
        person118 = 1
```

```
elif person == int(dem[118]):
        person119 = 1
elif person == int(dem[119]):
        person120 = 1
elif person == int(dem[120]):
        person121 = 1
elif person == int(dem[121]):
        person122 = 1
elif person == int(dem[122]):
        person123 = 1
elif person == int(dem[123]):
        person124 = 1
elif person == int(dem[124]):
        person125 = 1
elif person == int(dem[125]):
        person126 = 1
elif person == int(dem[126]):
        person127 = 1
elif person == int(dem[127]):
        person128 = 1
elif person == int(dem[128]):
        person129 = 1
elif person == int(dem[129]):
        person130 = 1
elif person == int(dem[130]):
        person131 = 1
elif person == int(dem[131]):
        person132 = 1
elif person == int(dem[132]):
        person133 = 1
elif person == int(dem[133]):
        person134 = 1
```

```
elif person == int(dem[134]):
        person135 = 1
elif person == int(dem[135]):
        person136 = 1
elif person == int(dem[136]):
        person137 = 1
elif person == int(dem[137]):
        person138 = 1
elif person == int(dem[138]):
        person139 = 1
elif person == int(dem[139]):
        person140 = 1
elif person == int(dem[140]):
        person141 = 1
elif person == int(dem[141]):
        person142 = 1
elif person == int(dem[142]):
        person143 = 1
elif person == int(dem[143]):
        person144 = 1
elif person == int(dem[144]):
        person145 = 1
elif person == int(dem[145]):
        person146 = 1
elif person == int(dem[146]):
        person147 = 1
elif person == int(dem[147]):
        person148 = 1
elif person == int(dem[148]):
        person149 = 1
elif person == int(dem[149]):
        person150 = 1
```

```
elif person == int(dem[150]):
        person151 = 1
#elif person == int(dem[151]):
#        person152 = 1
```

```
list.extend([personid, tripid, linkid, hour,
    streetlength, speedlimit, degree, percentage
    , speed_25, speed_30, speed_35, speed_40,
    speed_45, speed_50, speed_55, speed_60,
    speed_65, morning, afternoon, night,
    o_gender, o_education, o_age, gender,
    education_1, education_2, age_25_to_34,
    age_35_to_44, age_45_to_54, age_55_to_64,
    age_65_to_74, age_75_to_84, age_85_and_over,
     person1, person2, person3, person4, person5
    , person6, person7, person8, person9,
    person10, person11, person12, person13,
    person14, person15, person16, person17,
    person18, person19, person20, person21,
    person22, person23, person24, person25,
    person26, person27, person28, person29,
    person30, person31, person32, person33,
    person34, person35, person36, person37,
    person38, person39, person40, person41,
    person42, person43, person44, person45,
    person46, person47, person48, person49,
    person50, person51, person52, person53,
    person54, person55, person56, person57,
    person58, person59, person60, person61,
    person62, person63, person64, person65,
    person66, person67, person68, person69,
    person70, person71, person72, person73,
    person74, person75, person76, person77,
    person78, person79, person80, person81,
    person82, person83, person84, person85,
    person86, person87, person88, person89,
    person90, person91, person92, person93,
    person94, person95, person96, person97,
    person98, person99, person100, person101,
    person102, person103, person104, person105,
    person106, person107, person108, person109,
    person110, person111, person112, person113,
    person114, person115, person116, person117,
```

```
                writer.writerow(list)
        inputFile.close()
        print filename

path = "/Users/toshiyokoo/Desktop/aaafm"
for filename in os.listdir(path):
        global filename
        summarize()
```

### A.4.19 Computing and counting number in *R*

```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.65 (6833) x86_64-apple-darwin10.8.0]

[Workspace restored from /Users/toshiyokoo/.RData]
```

```
[History restored from /Users/toshiyokoo/.Rapp.history]

> fnames <- dir(pattern=".csv")
> csvlist <- lapply(fnames, read.csv)
> names(csvlist) <- fnames
> n <- length(csvlist)
> n
[1] 11
> temp <- csvlist[[1]]
> for (i in 2:n) {temp <- merge(temp, csvlist[[i]], all=T)}
> res <- temp
> summary(res)
             filename           tripid                     date
                                time          speed.kilometer.
 187010_02_02_1  :   7283   Min.   :1.013e+09   11/10/2011:
    27531   21:42:58:      54   Min.   :  6.00
 183019_02_02_21 :   3544   1st Qu.:1.229e+09   10/10/2011:
    26916   21:43:22:      53   1st Qu.: 31.00
 186319_02_02_44 :   3063   Median :1.505e+09   28/9/2011 :
    23472   21:43:21:      52   Median : 48.00
 184208_02_02_44 :   3048   Mean   :1.489e+09   22/7/2011 :
    19902   21:42:59:      51   Mean   : 51.99
 121926_02_02_25 :   2536   3rd Qu.:1.766e+09   12/10/2011:
    19845   21:44:20:      51   3rd Qu.: 70.00
 152077_01_02_112:   2480   Max.   :2.091e+09   29/9/2011 :
    18262   21:43:0 :      50   Max.   :391.00
 (Other)          :1150149                       (Other)
    :1036175   (Other) :1171792
  streetlength      speedlimit.mph. speedlimit.km.
     speed_limit_compliance      speed2             hour
 Min.   :  16.74   Min.   : 5.00   Min.   :  8.047   Min.
    :0.05736         Min.   :0.0000   Min.   :  0.00
```

```
1st Qu.: 136.60   1st Qu.:35.00   1st Qu.: 56.327   1st Qu
   .:0.53852         1st Qu.:0.0000   1st Qu.:10.00
Median : 231.35   Median :40.00   Median : 64.374   Median
   :0.80778          Median :0.0000   Median :14.00
Mean   : 362.13   Mean   :41.08   Mean   : 66.116   Mean
   :0.76621          Mean   :0.2329   Mean   :13.29
3rd Qu.: 442.00   3rd Qu.:40.00   3rd Qu.: 64.374   3rd Qu
   .:0.99419         3rd Qu.:0.0000   3rd Qu.:17.00
Max.   :4841.32   Max.   :70.00   Max.   :112.654   Max.
   :7.58073          Max.   :1.0000   Max.   :23.00

   speed_25          speed_30          speed_35
      speed_40          speed_45          speed_50
Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.
   :0.0000   Min.   :0.0000   Min.   :0.00000
1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu
   .:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
Median :0.00000   Median :0.0000   Median :0.0000   Median
   :0.0000   Median :0.0000   Median :0.00000
Mean   :0.02054   Mean   :0.1138   Mean   :0.3225   Mean
   :0.2959   Mean   :0.0193   Mean   :0.01706
3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu
   .:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.
   :1.0000   Max.   :1.0000   Max.   :1.00000

   speed_55          speed_60          speed_65
      morning          afternoon          night
Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.
   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu
   .:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
```

```
Median :0.0000    Median :0.00000    Median :0.00000    Median
   :0.0000    Median :0.0000    Median :0.0000
Mean    :0.1008    Mean    :0.07778    Mean    :0.02424    Mean
   :0.3494    Mean    :0.4653    Mean    :0.1701
3rd Qu.:0.0000    3rd Qu.:0.00000    3rd Qu.:0.00000    3rd Qu
   .:1.0000    3rd Qu.:1.0000    3rd Qu.:0.0000
Max.    :1.0000    Max.    :1.00000    Max.    :1.00000    Max.
   :1.0000    Max.    :1.0000    Max.    :1.0000

  o_gender                                    o_edu
     o_age             gender         education_1
Female:666621    Associates degree             : 48153    Min.
   : 5.000    Min.    :0.0000    Min.    :0.0000
Male  :505482    Bachelors degree             :428046    1st Qu
   .: 7.000    1st Qu.:0.0000    1st Qu.:0.0000
                 Graduate/Post-graduate degree:343870    Median
                    : 9.000    Median :0.0000    Median :0.0000
                 High school graduate         :140378    Mean
                    : 8.576    Mean    :0.4313    Mean    :0.1845
                 Some college                 :168149    3rd Qu
                    .:10.000    3rd Qu.:1.0000    3rd Qu.:0.0000
                 Vocational/Technical training: 43507    Max.
                    :12.000    Max.    :1.0000    Max.    :1.0000

 education_2       age_25_to_34       age_35_to_44
    age_45_to_54       age_55_to_64       age_65_to_74
Min.    :0.0000    Min.    :0.00000    Min.    :0.0000    Min.
   :0.0000    Min.    :0.0000    Min.    :0.0000
1st Qu.:0.0000    1st Qu.:0.00000    1st Qu.:0.0000    1st Qu
   .:0.0000    1st Qu.:0.0000    1st Qu.:0.0000
Median :1.0000    Median :0.00000    Median :0.0000    Median
   :0.0000    Median :0.0000    Median :0.0000
```

```
 Mean   :0.6586   Mean   :0.06539   Mean   :0.1697   Mean
    :0.1787   Mean   :0.3082   Mean   :0.1808
 3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu
    .:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.
    :1.0000   Max.   :1.0000   Max.   :1.0000

  age_75_to_84      age_85_and_over
 Min.   :0.00000   Min.   :0.0000
 1st Qu.:0.00000   1st Qu.:0.0000
 Median :0.00000   Median :0.0000
 Mean   :0.05869   Mean   :0.0209
 3rd Qu.:0.00000   3rd Qu.:0.0000
 Max.   :1.00000   Max.   :1.0000

> #Table
> #Percentage of speeding across speed limit zone
> table(res$speedlimit.mph., res$speed2)

        0      1
 5    145    614
 10  1468    896
 20     9     71
 25  5581  15286
 30 119937  13402
 35 339354  38606
 40 272477  74399
 45  15470   7154
 50  15303   4698
 55  60023  58137
 60  44765  46399
 65  17589  10827
```

```
  70   7025   2468
> x <- table(res$speedlimit.mph., res$speed2)
> write.csv (x, "x_csv.csv")
>
> #count the number of individuals who used specific speed
   limit
> r <- table(res$tripid, res$speedlimit.mph.)
> write.csv (r, "r_csv.csv")
>
> #Percentage of speeding by time of day
> y <- table(res$hour, res$speed2)
> write.csv (y, "y_csv.csv")
>
> #count the number of individuals who drove specific hour
> s <- table(res$tripid, res$hour)
> write.csv (s, "s_csv.csv")
>
> #Speeding by gender
> z <- table(res$gender, res$speed2)
> write.csv (z, "z_csv.csv")
>
> #count the number of driver depends on gender
> t <- table(res$tripid, res$gender)
> write.csv (t, "t_csv.csv")
>
> #Speeding by age
> e <- table(res$o_age, res$speed2)
> write.csv (e, "e_csv.csv")
>
> #count the number of driver depends on age
> u <- table(res$tripid, res$o_age)
> write.csv (u, "u_csv.csv")
```

```
>
> #Speeding by education level
> f <- table(res$o_edu, res$speed2)
> write.csv (f, "f_csv.csv")
>
> #count the number of driver depends on education level
> v <- table(res$tripid, res$education_1)
> write.csv (v, "v_csv.csv")
> w <- table(res$tripid, res$education_2)
> write.csv (w, "w_csv.csv")
>
```

### A.4.20 Box plot in *R*

```
> fnames <- dir(pattern=".csv")
> csvlist <- lapply(fnames, read.csv)
> names(csvlist) <- fnames
> n <- length(csvlist)
> n
[1] 11
> temp <- csvlist[[1]]
> for (i in 2:n) {temp <- merge(temp, csvlist[[i]], all=T)}
> res <- temp
>
> #Speed limit
> boxplot(speed_limit_compliance ~ speedlimit.mph., data=res,
   ylim=c(0, 8), yaxp=c(0, 8, 8), xlab="Speed limit(mph)", ylab
   ="driving speed/speed limit")
>
> #Time
> boxplot(speed_limit_compliance ~ hour, data=res, ylim=c(0, 8)
   , yaxp=c(0, 8, 8), xlab="Time", ylab="driving speed/speed
   limit")
```

```
>
> #Education
> boxplot(speed_limit_compliance ~ o_edu, data=res, ylim=c(0,
    8), yaxp=c(0, 8, 8), xlab="Education", ylab="driving speed/
    speed limit")
>
> #Age
> boxplot(speed_limit_compliance ~ o_age, data=res, ylim=c(0,
    8), yaxp=c(0, 8, 8), xlab="Age", ylab="driving speed/speed
    limit")
>
> #Gender
> boxplot(speed_limit_compliance ~ o_gender, data=res, ylim=c
    (0, 8), yaxp=c(0, 8, 8), xlab="Gender", ylab="driving speed/
    speed limit")
>
```

### A.4.21 Graph in *R*

```
> fnames <- dir(pattern=".csv")
> csvlist <- lapply(fnames, read.csv)
> names(csvlist) <- fnames
> n <- length(csvlist)
> n
[1] 11
> temp <- csvlist[[1]]
> for (i in 2:n) {temp <- merge(temp, csvlist[[i]], all=T)}
> res <- temp
> #Plot between street length and speeding
> street_length <- res$streetlength
> speeding <- res$speed_limit_compliance
```

```
> plot(street_length, speeding,pch=".", ylim=c(0, 8), yaxp=c(0,
    8, 8), xlab="Link length", ylab="driving speed/speed limit
    ")
> lm.obj<-lm(speeding~street_length)
> abline(lm.obj,col=2)
> summary(lm.obj)

Call:
lm(formula = speeding ~ street_length)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4096 -0.2039  0.0400  0.1957  6.8803

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.679e-01  4.162e-04  1604.8   <2e-16 ***
street_length 2.716e-04  8.212e-07   330.7   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3152 on 1172101 degrees of freedom
Multiple R-squared:  0.08536,    Adjusted R-squared:  0.08536
F-statistic: 1.094e+05 on 1 and 1172101 DF,  p-value: < 2.2e-16

> mtext("N=1,172,103, R^2=0.0854, y=0.6679+0.0003x", side=3)
>
> #Link length is less than 1,000m
> length1000 <- subset(res, res$streetlength <= 1000)
> street_length_1 <- length1000$streetlength
> speeding_1 <- length1000$speed_limit_compliance
```

```
> lm1000.obj<-lm(speeding_1~street_length_1)
> summary(lm1000.obj)

Call:
lm(formula = speeding_1 ~ street_length_1)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0270 -0.1976  0.0317  0.1914  6.9144

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     6.078e-01  5.075e-04  1197.5   <2e-16 ***
street_length_1 4.881e-04  1.363e-06   358.2   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3124 on 1112317 degrees of freedom
Multiple R-squared:  0.1034,    Adjusted R-squared:  0.1034
F-statistic: 1.283e+05 on 1 and 1112317 DF,  p-value: < 2.2e-16

> plot(street_length_1, speeding_1,pch=".", ylim=c(0, 8), yaxp=
    c(0, 8, 8), xlab="Link length", ylab="driving speed/speed
    limit")
> abline(lm1000.obj,col=2)
> mtext("N=1,112,319, R^2=0.103, y=0.6078+0.0005x", side=3)
>
> #polynomial model
> m2 <- lm(speeding ~ street_length + I(street_length^2))
> summary(m2)
```

```
Call:
lm(formula = speeding ~ street_length + I(street_length^2))

Residuals:
    Min      1Q  Median      3Q     Max
-1.0322 -0.1914  0.0327  0.1890  6.9124

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         6.025e-01  5.226e-04  1152.8   <2e-16 ***
street_length       5.679e-04  1.680e-06   338.1   <2e-16 ***
I(street_length^2) -1.632e-07  8.113e-10  -201.2   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3099 on 1172100 degrees of freedom
Multiple R-squared:  0.1159,    Adjusted R-squared:  0.1159
F-statistic: 7.681e+04 on 2 and 1172100 DF,  p-value: < 2.2e-16

> #polynomial model (Link length is less than 1,000m)
> m4<-lm(speeding_1~street_length_1+I(street_length_1^2))
> summary(m4)

Call:
lm(formula = speeding_1 ~ street_length_1 + I(street_length_1
    ^2))

Residuals:
    Min      1Q  Median      3Q     Max
-0.8900 -0.1926  0.0351  0.1863  6.9403
```

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              5.142e-01  8.627e-04   596.0   <2e-16 ***
street_length_1          1.142e-03  5.075e-06   225.0   <2e-16 ***
I(street_length_1^2)    -7.501e-07  5.614e-09  -133.6   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.31 on 1112316 degrees of freedom
Multiple R-squared:  0.1176,    Adjusted R-squared:  0.1176
F-statistic: 7.41e+04 on 2 and 1112316 DF,  p-value: < 2.2e-16

> #plot (speed limit: 25 mph)
> speed25 <- subset(res, res$speedlimit.mph.=="25")
> summary(speed25)
            filename          tripid                     date
                              time        speed.kilometer.
               streetlength
 126871_01_02_57:  253   Min.   :1.013e+09   11/10/2011:  597
      22:11:9 :    6   Min.   :  6.00   Min.   :  18.03
 115164_01_02_3 :  178   1st Qu.:1.219e+09   10/8/2011 :  481
      23:12:14:    6   1st Qu.:  39.00   1st Qu.: 303.12
 125425_01_02_65:  126   Median :1.373e+09   27/9/2011 :  467
      23:12:15:    6   Median :  57.00   Median : 375.95
 121926_01_02_21:  114   Mean   :1.428e+09   11/8/2011 :  448
      12:23:16:    5   Mean   :  57.02   Mean   : 375.67
 121926_01_02_18:  112   3rd Qu.:1.576e+09   9/8/2011  :  401
      12:23:17:    5   3rd Qu.:  78.00   3rd Qu.: 444.39
 176075_02_02_2 :  111   Max.   :2.091e+09   24/8/2011 :  393
      14:24:55:    5   Max.   :122.00   Max.   :1143.21
```

```
(Other)          :19973                          (Other)    :18080
     (Other) :20834
speedlimit.mph. speedlimit.km.  speed_limit_compliance
   speed2              hour            speed_25     speed_30
Min.    :25       Min.    :40.23   Min.    :0.1491          Min.
   :0.0000   Min.    : 0.00   Min.    :1    Min.    :0
1st Qu.:25       1st Qu.:40.23   1st Qu.:0.9693          1st Qu
   .:0.0000   1st Qu.: 9.00   1st Qu.:1    1st Qu.:0
Median :25       Median :40.23   Median :1.4167          Median
   :1.0000   Median :13.00   Median :1    Median :0
Mean    :25       Mean    :40.23   Mean    :1.4172          Mean
   :0.7325   Mean    :12.98   Mean    :1    Mean    :0
3rd Qu.:25       3rd Qu.:40.23   3rd Qu.:1.9387          3rd Qu
   .:1.0000   3rd Qu.:17.00   3rd Qu.:1    3rd Qu.:0
Max.    :25       Max.    :40.23   Max.    :3.0323          Max.
   :1.0000   Max.    :23.00   Max.    :1    Max.    :0


   speed_35     speed_40     speed_45     speed_50     speed_55
         speed_60     speed_65     morning
Min.    :0    Min.    :0    Min.    :0    Min.    :0    Min.    :0
   Min.    :0    Min.    :0    Min.    :0.0000
1st Qu.:0    1st Qu.:0    1st Qu.:0    1st Qu.:0    1st Qu.:0    1
   st Qu.:0    1st Qu.:0    1st Qu.:0.0000
Median :0    Median :0    Median :0    Median :0    Median :0
   Median :0    Median :0    Median :0.0000
Mean    :0    Mean    :0    Mean    :0    Mean    :0    Mean    :0
   Mean    :0    Mean    :0    Mean    :0.3736
3rd Qu.:0    3rd Qu.:0    3rd Qu.:0    3rd Qu.:0    3rd Qu.:0    3
   rd Qu.:0    3rd Qu.:0    3rd Qu.:1.0000
Max.    :0    Max.    :0    Max.    :0    Max.    :0    Max.    :0
   Max.    :0    Max.    :0    Max.    :1.0000
```

```
  afternoon          night          o_gender
                                    o_edu          o_age
Min.   :0.0000   Min.   :0.0000   Female:11322   Associates
   degree             : 575   Min.   : 5.000
1st Qu.:0.0000   1st Qu.:0.0000   Male   : 9545   Bachelors
   degree             :8330   1st Qu.: 7.000
Median :0.0000   Median :0.0000                  Graduate/Post
   -graduate degree:6325   Median : 8.000
Mean   :0.4195   Mean   :0.1756                  High school
   graduate           :2297   Mean   : 8.342
3rd Qu.:1.0000   3rd Qu.:0.0000                  Some college
                  :2138   3rd Qu.: 9.000
Max.   :1.0000   Max.   :1.0000                  Vocational/
   Technical training:1202   Max.   :12.000

    gender         education_1     education_2       age_25_to_34
          age_35_to_44     age_45_to_54      age_55_to_64
Min.   :0.0000   Min.   :0.00   Min.   :0.0000   Min.
   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:0.0000   1st Qu
   .:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
Median :0.0000   Median :0.00   Median :1.0000   Median
   :0.0000   Median :0.000   Median :0.0000   Median :0.0000
Mean   :0.4574   Mean   :0.13   Mean   :0.7023   Mean
   :0.1298   Mean   :0.161   Mean   :0.2016   Mean   :0.2786
3rd Qu.:1.0000   3rd Qu.:0.00   3rd Qu.:1.0000   3rd Qu
   .:0.0000   3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:1.0000
Max.   :1.0000   Max.   :1.00   Max.   :1.0000   Max.
   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000

 age_65_to_74      age_75_to_84      age_85_and_over
Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
```

```
 1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
 Median :0.0000   Median :0.00000   Median :0.00000
 Mean   :0.1555   Mean   :0.04179   Mean   :0.02037
 3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
 Max.   :1.0000   Max.   :1.00000   Max.   :1.00000

> plot(speed25$streetlength, speed25$speed_limit_compliance,pch
    =".", ylim=c(0, 3), yaxp=c(0, 3, 6), xlab="Link length (
    Speed limit: 25mph)", ylab="driving speed/speed limit")
> lm_25.obj<-lm(speed25$speed_limit_compliance~
    speed25$streetlength)
> summary(lm_25.obj)

Call:
lm(formula = speed25$speed_limit_compliance ~
    speed25$streetlength)

Residuals:
     Min       1Q   Median       3Q      Max
-1.89516 -0.44544  0.03034  0.45883  1.56718

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.074e+00  1.186e-02   90.52   <2e-16 ***
speed25$streetlength  9.142e-04  2.943e-05   31.06   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.6208 on 20865 degrees of freedom
Multiple R-squared:  0.04419,    Adjusted R-squared:  0.04415
F-statistic: 964.8 on 1 and 20865 DF,  p-value: < 2.2e-16
```

```
> abline(lm_25.obj, col=2)
> mtext("N=20,867, R^2=0.0442, y=1.0740+0.0009x", side=3)
>
> #plot (speed limit: 30 mph)
> speed30 <- subset(res, res$speedlimit.mph.=="30")
> summary(speed30)
                filename          tripid                     date
                                    time         speed.kilometer.
 152077_01_02_112:  1914   Min.   :1.013e+09   10/7/2011 :
    4995   16:38:20:    12   Min.   :  6.00
 125650_01_02_47 :  1520   1st Qu.:1.229e+09   24/10/2011:
    4123   16:53:31:    12   1st Qu.: 19.00
 125650_02_02_63 :  1419   Median :1.443e+09   6/9/2011  :
    3458   16:36:37:    11   Median : 30.00
 184208_02_02_36 :  1287   Mean   :1.461e+09   23/7/2011 :
    3406   16:36:41:    11   Mean   : 30.54
 184208_02_02_26 :  1192   3rd Qu.:1.761e+09   21/7/2011 :
    3338   16:36:43:    11   3rd Qu.: 43.00
 184208_02_02_44 :  1190   Max.   :2.090e+09   22/7/2011 :
    3190   16:38:21:    11   Max.   :113.00
 (Other)          :124817                       (Other)
    :110829   (Other) :133271
  streetlength     speedlimit.mph. speedlimit.km.
     speed_limit_compliance      speed2             hour
 Min.   :  21.4   Min.   :30       Min.   :48.28   Min.
    :0.1243          Min.   :0.0000   Min.   : 0.00
 1st Qu.: 102.1   1st Qu.:30       1st Qu.:48.28   1st Qu
    .:0.3935          1st Qu.:0.0000   1st Qu.:10.00
 Median : 135.5   Median :30       Median :48.28   Median
    :0.6214          Median :0.0000   Median :14.00
```

```
Mean   : 187.3   Mean   :30      Mean   :48.28   Mean
   :0.6326         Mean   :0.1005   Mean   :13.47
3rd Qu.: 201.5   3rd Qu.:30      3rd Qu.:48.28   3rd Qu
   .:0.8906         3rd Qu.:0.0000   3rd Qu.:17.00
Max.   :1244.6   Max.   :30      Max.   :48.28   Max.
   :2.3405         Max.   :1.0000   Max.   :23.00

   speed_25     speed_30     speed_35     speed_40     speed_45
        speed_50     speed_55     speed_60     speed_65
Min.   :0   Min.   :1   Min.   :0   Min.   :0   Min.   :0
   Min.   :0   Min.   :0   Min.   :0   Min.   :0
1st Qu.:0   1st Qu.:1   1st Qu.:0   1st Qu.:0   1st Qu.:0   1
   st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:0
Median :0   Median :1   Median :0   Median :0   Median :0
   Median :0   Median :0   Median :0   Median :0
Mean   :0   Mean   :1   Mean   :0   Mean   :0   Mean   :0
   Mean   :0   Mean   :0   Mean   :0   Mean   :0
3rd Qu.:0   3rd Qu.:1   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3
   rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0
Max.   :0   Max.   :1   Max.   :0   Max.   :0   Max.   :0
   Max.   :0   Max.   :0   Max.   :0   Max.   :0

   morning          afternoon          night          o_gender
                                      o_edu
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Female
   :72062   Associates degree            : 6718
1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   Male
   :61277   Bachelors degree             :41888
Median :0.0000   Median :0.0000   Median :0.0000
                    Graduate/Post-graduate degree:42775
Mean   :0.3407   Mean   :0.4798   Mean   :0.1664
                    High school graduate         : 8674
```

```
3rd Qu.:1.0000    3rd Qu.:1.0000    3rd Qu.:0.0000
                     Some college                    :30281
Max.    :1.0000    Max.    :1.0000    Max.    :1.0000
                     Vocational/Technical training: 3003

    o_age            gender          education_1       education_2
              age_25_to_34       age_35_to_44
Min.    : 5.00    Min.    :0.0000    Min.    :0.0000    Min.
   :0.0000    Min.    :0.00000    Min.    :0.0000
1st Qu.: 7.00    1st Qu.:0.0000    1st Qu.:0.0000    1st Qu
   .:0.0000    1st Qu.:0.00000    1st Qu.:0.0000
Median : 9.00    Median :0.0000    Median :0.0000    Median
   :1.0000    Median :0.00000    Median :0.0000
Mean    : 8.46    Mean    :0.4596    Mean    :0.2775    Mean
   :0.6349    Mean    :0.05434    Mean    :0.2436
3rd Qu.: 9.00    3rd Qu.:1.0000    3rd Qu.:1.0000    3rd Qu
   .:1.0000    3rd Qu.:0.00000    3rd Qu.:0.0000
Max.    :12.00    Max.    :1.0000    Max.    :1.0000    Max.
   :1.0000    Max.    :1.00000    Max.    :1.0000

 age_45_to_54      age_55_to_64      age_65_to_74
    age_75_to_84      age_85_and_over
Min.    :0.0000    Min.    :0.0000    Min.    :0.0000    Min.
   :0.00000    Min.    :0.00000
1st Qu.:0.0000    1st Qu.:0.0000    1st Qu.:0.0000    1st Qu
   .:0.00000    1st Qu.:0.00000
Median :0.0000    Median :0.0000    Median :0.0000    Median
   :0.00000    Median :0.00000
Mean    :0.1686    Mean    :0.3166    Mean    :0.1075    Mean
   :0.08826    Mean    :0.01127
3rd Qu.:0.0000    3rd Qu.:1.0000    3rd Qu.:0.0000    3rd Qu
   .:0.00000    3rd Qu.:0.00000
```

```
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.
    :1.00000   Max.   :1.00000

> plot(speed30$streetlength, speed30$speed_limit_compliance,pch
   =".", ylim=c(0, 3), yaxp=c(0, 3, 6), xlab="Link length (
   Speed limit: 30mph)", ylab="driving speed/speed limit")
> lm_30.obj<-lm(speed30$speed_limit_compliance~
   speed30$streetlength)
> summary(lm_30.obj)

Call:
lm(formula = speed30$speed_limit_compliance ~
   speed30$streetlength)

Residuals:
     Min       1Q   Median       3Q      Max
-0.50857 -0.23909 -0.01125  0.25799  1.70788

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          6.326e-01  1.430e-03 442.303   <2e-16 ***
speed30$streetlength 1.965e-07  6.224e-06   0.032    0.975
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1         1

Residual standard error: 0.3025 on 133337 degrees of freedom
Multiple R-squared:  7.476e-09, Adjusted R-squared:  -7.492e-06
F-statistic: 0.0009968 on 1 and 133337 DF,  p-value: 0.9748

> abline(lm_30.obj, col=2)
> mtext("N=133,339, R^2=7.5e-09, y=0.6326+1.97e-07x", side=3)
```

```
>
> #plot (speed limit: 35 mph)
> speed35 <- subset(res, res$speedlimit.mph.=="35")
> summary(speed35)
                filename          tripid                     date
                                    time          speed.kilometer.
 187010_02_02_1 :  2364   Min.   :1.013e+09   11/10/2011: 12223
       20:38:12:    27   Min.   :  6.00
 127709_02_02_48:  1690   1st Qu.:1.219e+09   10/10/2011:  9588
       20:38:13:    26   1st Qu.: 22.00
 127709_02_02_67:  1606   Median :1.373e+09   30/9/2011 :  8441
       20:38:10:    23   Median : 37.00
 137171_01_02_8 :  1590   Mean   :1.457e+09   28/9/2011 :  7894
       20:38:14:    23   Mean   : 35.89
 193857_01_02_69:  1510   3rd Qu.:1.721e+09   8/10/2011 :  6923
       20:38:11:    22   3rd Qu.: 48.00
 184208_02_02_31:  1508   Max.   :2.091e+09   12/10/2011:  6910
       21:25:43:    22   Max.   :150.00
 (Other)         :367692                       (Other)   :325981
       (Other) :377817
  streetlength      speedlimit.mph. speedlimit.km.
     speed_limit_compliance      speed2             hour
 Min.   :  18.05   Min.   :35      Min.   :56.33   Min.
    :0.1065          Min.   :0.0000   Min.   : 0.00
 1st Qu.: 111.10   1st Qu.:35      1st Qu.:56.33   1st Qu
    .:0.3906          1st Qu.:0.0000   1st Qu.: 9.00
 Median : 197.51   Median :35      Median :56.33   Median
    :0.6569          Median :0.0000   Median :13.00
 Mean   : 243.35   Mean   :35      Mean   :56.33   Mean
    :0.6372          Mean   :0.1021   Mean   :13.18
 3rd Qu.: 291.75   3rd Qu.:35      3rd Qu.:56.33   3rd Qu
    .:0.8522          3rd Qu.:0.0000   3rd Qu.:16.00
```

```
Max.   :3334.96   Max.   :35       Max.   :56.33   Max.
   :2.6630          Max.   :1.0000   Max.   :23.00

   speed_25     speed_30     speed_35     speed_40     speed_45
        speed_50     speed_55     speed_60     speed_65
Min.   :0   Min.   :0   Min.   :1   Min.   :0   Min.   :0
   Min.   :0   Min.   :0   Min.   :0   Min.   :0
1st Qu.:0   1st Qu.:0   1st Qu.:1   1st Qu.:0   1st Qu.:0   1
   st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:0
Median :0   Median :0   Median :1   Median :0   Median :0
   Median :0   Median :0   Median :0   Median :0
Mean   :0   Mean   :0   Mean   :1   Mean   :0   Mean   :0
   Mean   :0   Mean   :0   Mean   :0   Mean   :0
3rd Qu.:0   3rd Qu.:0   3rd Qu.:1   3rd Qu.:0   3rd Qu.:0   3
   rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0
Max.   :0   Max.   :0   Max.   :1   Max.   :0   Max.   :0
   Max.   :0   Max.   :0   Max.   :0   Max.   :0

   morning         afternoon          night          o_gender
                                      o_edu
Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Female
   :200978   Associates degree            : 13270
1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   Male
   :176982   Bachelors degree             :126144
Median :0.000   Median :0.0000   Median :0.0000
                      Graduate/Post-graduate degree:116098
Mean   :0.362   Mean   :0.4685   Mean   :0.1589
                      High school graduate         : 58226
3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:0.0000
                      Some college                 : 50968
Max.   :1.000   Max.   :1.0000   Max.   :1.0000
                      Vocational/Technical training: 13254
```

```
    o_age            gender         education_1     education_2
          age_25_to_34       age_35_to_44
Min.   : 5.000   Min.   :0.0000   Min.   :0.00   Min.
   :0.0000   Min.   :0.00000   Min.   :0.0000
1st Qu.: 7.000   1st Qu.:0.0000   1st Qu.:0.00   1st Qu
   .:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
Median : 9.000   Median :0.0000   Median :0.00   Median
   :1.0000   Median :0.00000   Median :0.0000
Mean   : 8.499   Mean   :0.4683   Mean   :0.17   Mean
   :0.6409   Mean   :0.06344   Mean   :0.1615
3rd Qu.:10.000   3rd Qu.:1.0000   3rd Qu.:0.00   3rd Qu
   .:1.0000   3rd Qu.:0.00000   3rd Qu.:0.0000
Max.   :12.000   Max.   :1.0000   Max.   :1.00   Max.
   :1.0000   Max.   :1.00000   Max.   :1.0000

 age_45_to_54     age_55_to_64     age_65_to_74
    age_75_to_84     age_85_and_over
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.
   :0.00000   Min.   :0.00000
1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu
   .:0.00000   1st Qu.:0.00000
Median :0.0000   Median :0.0000   Median :0.0000   Median
   :0.00000   Median :0.00000
Mean   :0.2094   Mean   :0.2694   Mean   :0.1947   Mean
   :0.05336   Mean   :0.01621
3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu
   .:0.00000   3rd Qu.:0.00000
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.
   :1.00000   Max.   :1.00000
```

```
> plot(speed35$streetlength, speed35$speed_limit_compliance,pch
    =".", ylim=c(0, 3), yaxp=c(0, 3, 6), xlab="Link length (
    Speed limit: 35mph)", ylab="driving speed/speed limit")
> lm_35.obj<-lm(speed35$speed_limit_compliance~
    speed35$streetlength)
> summary(lm_35.obj)

Call:
lm(formula = speed35$speed_limit_compliance ~
    speed35$streetlength)

Residuals:
      Min        1Q    Median        3Q       Max
-1.06627  -0.21815   0.01676   0.21111   2.05986

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           5.780e-01  7.266e-04   795.5   <2e-16 ***
speed35$streetlength  2.433e-04  2.260e-06   107.6   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.292 on 377958 degrees of freedom
Multiple R-squared:  0.02974,    Adjusted R-squared:  0.02974
F-statistic: 1.159e+04 on 1 and 377958 DF,  p-value: < 2.2e-16

> abline(lm_35.obj, col=2)
> mtext("N=377,960, R^2=0.0297, y=0.5780+0.0002x", side=3)
>
> #plot (speed limit: 40 mph)
> speed40 <- subset(res, res$speedlimit.mph.=="40")
```

```
> summary(speed40)
            filename          tripid                     date
                                time          speed.kilometer.
 187010_02_02_1 :  2233   Min.   :1.013e+09   10/10/2011:  9230
       21:12:31:    23   Min.   :  6.00
 183019_02_02_21:  2167   1st Qu.:1.254e+09   28/9/2011 :  8522
       22:42:22:    23   1st Qu.: 41.00
 183019_01_02_23:  1737   Median :1.520e+09   11/10/2011:  8194
       21:12:29:    22   Median : 52.00
 157551_01_02_12:  1242   Mean   :1.510e+09   29/9/2011 :  8160
       21:12:30:    22   Mean   : 50.56
 157551_01_02_30:  1239   3rd Qu.:1.830e+09   22/7/2011 :  5891
       21:12:28:    21   3rd Qu.: 61.00
 194902_02_02_40:  1087   Max.   :2.091e+09   12/10/2011:  5700
       21:12:32:    21   Max.   :391.00
 (Other)         :337171                       (Other)   :301179
       (Other) :346744
  streetlength      speedlimit.mph. speedlimit.km.
     speed_limit_compliance      speed2             hour
 Min.   :  16.74   Min.   :40      Min.   :64.37   Min.
    :0.09321        Min.   :0.0000   Min.   : 0.00
 1st Qu.: 135.87   1st Qu.:40      1st Qu.:64.37   1st Qu
    .:0.63691       1st Qu.:0.0000   1st Qu.:10.00
 Median : 203.00   Median :40      Median :64.37   Median
    :0.80778        Median :0.0000   Median :14.00
 Mean   : 284.99   Mean   :40      Mean   :64.37   Mean
    :0.78536        Mean   :0.2145   Mean   :13.45
 3rd Qu.: 344.86   3rd Qu.:40      3rd Qu.:64.37   3rd Qu
    .:0.94759       3rd Qu.:0.0000   3rd Qu.:17.00
 Max.   :3074.42   Max.   :40      Max.   :64.37   Max.
    :6.07390        Max.   :1.0000   Max.   :23.00
```

```
   speed_25      speed_30      speed_35      speed_40      speed_45
        speed_50      speed_55      speed_60      speed_65
Min.   :0   Min.   :0   Min.   :0   Min.   :1   Min.   :0
   Min.   :0   Min.   :0   Min.   :0   Min.   :0
1st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:1   1st Qu.:0   1
   st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:0
Median :0   Median :0   Median :0   Median :1   Median :0
   Median :0   Median :0   Median :0   Median :0
Mean   :0   Mean   :0   Mean   :0   Mean   :1   Mean   :0
   Mean   :0   Mean   :0   Mean   :0   Mean   :0
3rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:1   3rd Qu.:0   3
   rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0
Max.   :0   Max.   :0   Max.   :0   Max.   :1   Max.   :0
   Max.   :0   Max.   :0   Max.   :0   Max.   :0

   morning          afternoon          night          o_gender
                                        o_edu
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Female
   :214868   Associates degree             : 12172
1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   Male
   :132008   Bachelors degree              :131846
Median :0.0000   Median :0.0000   Median :0.0000
                        Graduate/Post-graduate degree:108089
Mean   :0.3391   Mean   :0.4863   Mean   :0.1673
                        High school graduate          : 33423
3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
                        Some college                  : 52847
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
                        Vocational/Technical training:  8499

    o_age             gender         education_1
       education_2       age_25_to_34       age_35_to_44
```

```
 Min.   : 5.000   Min.   :0.0000   Min.   :0.0000   Min.
    :0.0000   Min.   :0.00000   Min.   :0.0000
 1st Qu.: 8.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu
    .:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
 Median : 9.000   Median :0.0000   Median :0.0000   Median
    :1.0000   Median :0.00000   Median :0.0000
 Mean   : 8.766   Mean   :0.3806   Mean   :0.1874   Mean
    :0.6917   Mean   :0.05571   Mean   :0.1475
 3rd Qu.:10.000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu
    .:1.0000   3rd Qu.:0.00000   3rd Qu.:0.0000
 Max.   :12.000   Max.   :1.0000   Max.   :1.0000   Max.
    :1.0000   Max.   :1.00000   Max.   :1.0000

  age_45_to_54     age_55_to_64     age_65_to_74
     age_75_to_84     age_85_and_over
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.
    :0.00000   Min.   :0.00000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu
    .:0.00000   1st Qu.:0.00000
 Median :0.0000   Median :0.0000   Median :0.0000   Median
    :0.00000   Median :0.00000
 Mean   :0.1502   Mean   :0.3385   Mean   :0.2117   Mean
    :0.06489   Mean   :0.02321
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu
    .:0.00000   3rd Qu.:0.00000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.
    :1.00000   Max.   :1.00000

> plot(speed40$streetlength, speed40$speed_limit_compliance,pch
   =".", ylim=c(0, 3), yaxp=c(0, 3, 6), xlab="Link length (
   Speed limit: 40mph)", ylab="driving speed/speed limit")
```

```
> lm_40.obj<-lm(speed40$speed_limit_compliance~
    speed40$streetlength)
> abline(lm_40.obj, col=2)
> summary(lm_40.obj)

Call:
lm(formula = speed40$speed_limit_compliance ~
    speed40$streetlength)

Residuals:
     Min       1Q   Median       3Q      Max
-1.8500  -0.1463   0.0350   0.1724   5.3431

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)             6.671e-01  7.175e-04   929.7   <2e-16 ***
speed40$streetlength    4.151e-04  1.913e-06   217.0   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.2748 on 346874 degrees of freedom
Multiple R-squared:  0.1195,     Adjusted R-squared:  0.1195
F-statistic: 4.708e+04 on 1 and 346874 DF,  p-value: < 2.2e-16

> mtext("N=346,876, R^2=0.1195, y=0.6671+0.0004x", side=3)
>
```

### A.4.22 Regression analysis in *R*

```
> fnames <- dir(pattern=".csv")
> csvlist <- lapply(fnames, read.csv)
> names(csvlist) <- fnames
```

```
> n <- length(csvlist)
> n
[1] 11
> temp <- csvlist[[1]]
> for (i in 2:n) {temp <- merge(temp, csvlist[[i]], all=T)}
> res <- temp
> #Multi Linear Regression
> regression =lm(speed_limit_compliance~streetlength+speed_25+
    speed_30+speed_35+speed_40+speed_45+speed_50+speed_55+
    speed_60+speed_65+age_25_to_34+age_35_to_44+age_45_to_54+
    age_55_to_64+age_65_to_74+age_75_to_84+age_85_and_over+
    gender+education_1+education_2+morning+afternoon+night,res)
> summary(regression)

Call:
lm(formula = speed_limit_compliance ~ streetlength + speed_25 +
    speed_30 + speed_35 + speed_40 + speed_45 + speed_50 +
        speed_55 +
    speed_60 + speed_65 + age_25_to_34 + age_35_to_44 +
        age_45_to_54 +
    age_55_to_64 + age_65_to_74 + age_75_to_84 +
        age_85_and_over +
    gender + education_1 + education_2 + morning + afternoon +
    night, data = res)

Residuals:
    Min      1Q  Median      3Q     Max
-1.3491 -0.1723  0.0337  0.1780  6.2159

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      6.878e-01  4.433e-03 155.171  < 2e-16 ***
```

```
streetlength      1.650e-04  9.496e-07 173.801  < 2e-16 ***
speed_25          6.365e-01  3.659e-03 173.934  < 2e-16 ***
speed_30         -9.581e-02  3.302e-03 -29.015  < 2e-16 ***
speed_35         -1.035e-01  3.219e-03 -32.139  < 2e-16 ***
speed_40          3.614e-02  3.211e-03  11.255  < 2e-16 ***
speed_45          6.506e-02  3.683e-03  17.665  < 2e-16 ***
speed_50          4.953e-02  3.734e-03  13.265  < 2e-16 ***
speed_55          1.226e-01  3.199e-03  38.317  < 2e-16 ***
speed_60          1.166e-01  3.191e-03  36.527  < 2e-16 ***
speed_65          7.844e-02  3.478e-03  22.555  < 2e-16 ***
age_25_to_34      5.604e-02  2.406e-03  23.292  < 2e-16 ***
age_35_to_44      1.221e-02  2.235e-03   5.462 4.70e-08 ***
age_45_to_54      3.321e-02  2.218e-03  14.976  < 2e-16 ***
age_55_to_64      4.196e-02  2.180e-03  19.246  < 2e-16 ***
age_65_to_74      3.868e-02  2.222e-03  17.403  < 2e-16 ***
age_75_to_84      3.389e-03  2.387e-03   1.420   0.1557
age_85_and_over   8.744e-02  2.832e-03  30.881  < 2e-16 ***
gender           -1.341e-02  5.658e-04 -23.704  < 2e-16 ***
education_1      -4.808e-03  9.801e-04  -4.906 9.30e-07 ***
education_2      -1.705e-02  8.001e-04 -21.314  < 2e-16 ***
morning           4.021e-03  2.238e-03   1.796   0.0724 .
afternoon        -1.322e-02  2.228e-03  -5.934 2.95e-09 ***
night             1.595e-02  2.288e-03   6.972 3.12e-12 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05
      .  0.1  1

Residual standard error: 0.2898 on 1172079 degrees of freedom
Multiple R-squared:  0.2269,    Adjusted R-squared:  0.2269
F-statistic: 1.496e+04 on 23 and 1172079 DF,  p-value: < 2.2e
    -16
```

```
> write.csv (summary(regression)$coef, "regression.csv")
>
> #Regression by street length
> regression2 =lm(speed_limit_compliance~streetlength,res)
> summary(regression2)

Call:
lm(formula = speed_limit_compliance ~ streetlength, data = res)

Residuals:
     Min       1Q   Median       3Q      Max
-1.4096  -0.2039   0.0400   0.1957   6.8803

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.679e-01  4.162e-04  1604.8   <2e-16 ***
streetlength 2.716e-04  8.212e-07   330.7   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3152 on 1172101 degrees of freedom
Multiple R-squared:  0.08536,    Adjusted R-squared:  0.08536
F-statistic: 1.094e+05 on 1 and 1172101 DF,  p-value: < 2.2e-16

> write.csv (summary(regression2)$coef, "regression2.csv")
>
> #Regression by speed limit
> regression22 =lm(speed_limit_compliance~speed_25+speed_30+
   speed_35+speed_40+speed_45+speed_50+speed_55+speed_60+
   speed_65,res)
> summary(regression22)
```

```
Call:
lm(formula = speed_limit_compliance ~ speed_25 + speed_30 +
    speed_35 +
     speed_40 + speed_45 + speed_50 + speed_55 + speed_60 +
        speed_65,
     data = res)

Residuals:
     Min       1Q   Median       3Q      Max
-1.2500  -0.1756   0.0281   0.1794   6.1816

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.939460   0.003022 310.909  < 2e-16 ***
speed_25     0.459701   0.003568 128.836  < 2e-16 ***
speed_30    -0.306821   0.003127 -98.108  < 2e-16 ***
speed_35    -0.302239   0.003059 -98.792  < 2e-16 ***
speed_40    -0.154104   0.003063 -50.316  < 2e-16 ***
speed_45    -0.109113   0.003600 -30.307  < 2e-16 ***
speed_50    -0.103670   0.003669 -28.253  < 2e-16 ***
speed_55    -0.012994   0.003141  -4.137 3.52e-05 ***
speed_60     0.008102   0.003175   2.552   0.0107 *
speed_65    -0.006673   0.003490  -1.912   0.0559 .
---
Signif. codes:  0     ***     0.001     **     0.01     *     0.05
      .     0.1         1

Residual standard error: 0.2944 on 1172093 degrees of freedom
Multiple R-squared:  0.2021,     Adjusted R-squared:  0.2021
F-statistic: 3.299e+04 on 9 and 1172093 DF,  p-value: < 2.2e-16
```

```
> write.csv (summary(regression2)$coef, "regression22.csv")
>
> #Regression by age
> regression3=lm(speed_limit_compliance~age_25_to_34+
    age_35_to_44+age_45_to_54+age_55_to_64+age_65_to_74+
    age_75_to_84+age_85_and_over,res)
> summary(regression3)

Call:
lm(formula = speed_limit_compliance ~ age_25_to_34 +
    age_35_to_44 +
     age_45_to_54 + age_55_to_64 + age_65_to_74 + age_75_to_84 +
     age_85_and_over, data = res)

Residuals:
     Min       1Q   Median       3Q      Max
-0.7798  -0.2253   0.0430   0.2206   6.8169

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.702842   0.002281 308.161  < 2e-16 ***
age_25_to_34    0.117389   0.002571  45.662  < 2e-16 ***
age_35_to_44    0.038872   0.002397  16.219  < 2e-16 ***
age_45_to_54    0.049881   0.002391  20.862  < 2e-16 ***
age_55_to_64    0.082682   0.002345  35.254  < 2e-16 ***
age_65_to_74    0.060960   0.002390  25.509  < 2e-16 ***
age_75_to_84    0.013250   0.002602   5.093 3.53e-07 ***
age_85_and_over 0.139055   0.003099  44.868  < 2e-16 ***
---
Signif. codes:  0     ***     0.001     **     0.01     *     0.05
      .     0.1         1
```

```
Residual standard error: 0.3284 on 1172095 degrees of freedom
Multiple R-squared:  0.007174,  Adjusted R-squared:  0.007168
F-statistic:  1210 on 7 and 1172095 DF,  p-value: < 2.2e-16

> write.csv (summary(regression3)$coef, "regression3.csv")
>
> #Regression by gender
> reg=lm(speed_limit_compliance~gender,res)
> summary(reg)

Call:
lm(formula = speed_limit_compliance ~ gender, data = res)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7178 -0.2248  0.0327  0.2191  6.8055

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.7751797  0.0004035 1921.23   <2e-16 ***
gender      -0.0207935  0.0006144  -33.84   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3294 on 1172101 degrees of freedom
Multiple R-squared:  0.0009762, Adjusted R-squared:  0.0009754
F-statistic:  1145 on 1 and 1172101 DF,  p-value: < 2.2e-16

> write.csv (summary(reg)$coef, "regression4.csv")
>
> #Regression by education
```

```
> reg2=lm(speed_limit_compliance~education_1+education_2,res)
> summary(reg2)

Call:
lm(formula = speed_limit_compliance ~ education_1 + education_2
    ,
    data = res)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7195 -0.2265  0.0442  0.2193  6.8142

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  0.7768322  0.0007685 1010.813   <2e-16 ***
education_1 -0.0103085  0.0010453   -9.861   <2e-16 ***
education_2 -0.0132370  0.0008552  -15.479   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3296 on 1172100 degrees of freedom
Multiple R-squared:  0.0002046, Adjusted R-squared:  0.0002029
F-statistic: 119.9 on 2 and 1172100 DF,  p-value: < 2.2e-16

> write.csv (summary(reg2)$coef, "regression5.csv")
>
> #Regression by time
> reg3=lm(speed_limit_compliance~morning + afternoon + night,
    res)
> summary(reg3)
```

```
Call:
lm(formula = speed_limit_compliance ~ morning + afternoon +
    night,
     data = res)

Residuals:
     Min       1Q   Median       3Q      Max
-0.7744  -0.2234   0.0375   0.2229   6.8296

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.848985   0.002470  343.76   <2e-16 ***
morning     -0.078751   0.002523  -31.22   <2e-16 ***
afternoon   -0.097900   0.002510  -39.01   <2e-16 ***
night       -0.057023   0.002577  -22.12   <2e-16 ***
---
Signif. codes:  0     ***     0.001     **     0.01     *     0.05
      .     0.1         1

Residual standard error: 0.3291 on 1172099 degrees of freedom
Multiple R-squared:  0.003026,  Adjusted R-squared:  0.003024
F-statistic:  1186 on 3 and 1172099 DF,  p-value: < 2.2e-16

> write.csv (summary(reg3)$coef, "regression6.csv")
>
```

### A.4.23 Dummy variable regression in *R*

```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.65 (6833) x86_64-apple-darwin10.8.0]

[Workspace restored from /Users/toshiyokoo/.RData]
[History restored from /Users/toshiyokoo/.Rapp.history]

> fnames <- dir(pattern=".csv")
> csvlist <- lapply(fnames, read.csv)
> names(csvlist) <- fnames
> n <- length(csvlist)
> n
[1] 1
> temp <- csvlist[[1]]
> res <- temp
> summary(res)
     PersonID                      TripID          LinkID
                 Hour        streetlength      Speedlimit
               degree
```

```
Min.   :1.013e+09   187010_02_02_1 :   441   Min.   :   22
    Min.   : 0.00   Min.   :  16.74   Min.   : 5.00   Min.
    :0.05736
1st Qu.:1.219e+09   183019_02_02_21:   336   1st Qu.: 50947
    1st Qu.:10.00   1st Qu.: 101.03   1st Qu.:35.00   1st Qu
    .:0.59829
Median :1.477e+09   127709_02_02_67:   284   Median : 75620
    Median :13.00   Median : 165.53   Median :40.00   Median
    :0.82004
Mean   :1.481e+09   186319_02_02_44:   270   Mean   : 72883
    Mean   :13.25   Mean   : 254.53   Mean   :40.27   Mean
    :0.80202
3rd Qu.:1.761e+09   127709_02_02_48:   261   3rd Qu.: 97953
    3rd Qu.:17.00   3rd Qu.: 307.62   3rd Qu.:40.00   3rd Qu
    .:0.99419
Max.   :2.091e+09   144314_01_02_40:   255   Max.   :152636
    Max.   :23.00   Max.   :4841.32   Max.   :70.00   Max.
    :7.58073
                    (Other)         :113689
  percentage         speed_25           speed_30
     speed_35          speed_40          speed_45
     speed_50
Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.
    :0.0000   Min.   :0.0000   Min.   :0.00000   Min.
    :0.00000
1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu
    .:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu
    .:0.00000
Median :0.0000   Median :0.00000   Median :0.0000   Median
    :0.0000   Median :0.0000   Median :0.00000   Median
    :0.00000
```

```
Mean   :0.2465   Mean   :0.02546   Mean   :0.1168   Mean
    :0.3288   Mean   :0.3168   Mean   :0.01797   Mean
    :0.01333
3rd Qu.:0.4767   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu
    .:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu
    .:0.00000
Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.
    :1.0000   Max.   :1.0000   Max.   :1.00000   Max.
    :1.00000

   speed_55          speed_60          speed_65
      morning          afternoon          night
      o_gender
Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.
    :0.0000   Min.   :0.000   Min.   :0.0000   Female:66198
1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu
    .:0.0000   1st Qu.:0.000   1st Qu.:0.0000   Male  :49338
Median :0.00000   Median :0.0000   Median :0.00000   Median
    :0.0000   Median :0.000   Median :0.0000
Mean   :0.09751   Mean   :0.0579   Mean   :0.02017   Mean
    :0.3525   Mean   :0.465   Mean   :0.1665
3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu
    .:1.0000   3rd Qu.:1.000   3rd Qu.:0.0000
Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.
    :1.0000   Max.   :1.000   Max.   :1.0000

                          o_edu          o_age
                             gender         education_1
                             education_2       age_25_to_34
Associates degree            : 4238   Min.   : 5.000   Min.
    :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
```

```
Bachelors degree                :40086   1st Qu.: 8.000   1st Qu
   .:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
Graduate/Post-graduate degree:35579   Median : 9.000   Median
   :0.000   Median :0.0000   Median :1.0000   Median :0.00000
High school graduate            :14006   Mean   : 8.639   Mean
   :0.427   Mean   :0.1848   Mean   :0.6549   Mean   :0.05835
Some college                    :17112   3rd Qu.:10.000   3rd Qu
   .:1.000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.00000
Vocational/Technical training: 4515   Max.   :12.000   Max.
   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000

 age_35_to_44      age_45_to_54      age_55_to_64
    age_65_to_74      age_75_to_84      age_85_and_over
    person1
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.
   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.
   :0.00000
1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu
   .:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu
   .:0.00000
Median :0.0000   Median :0.0000   Median :0.0000   Median
   :0.0000   Median :0.00000   Median :0.00000   Median
   :0.00000
Mean   :0.1562   Mean   :0.1857   Mean   :0.3105   Mean
   :0.1869   Mean   :0.06209   Mean   :0.02317   Mean
   :0.01064
3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu
   .:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu
   .:0.00000
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.
   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.
   :1.00000
```

```
    person2             person3             person4
      person5             person6             person7
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
     :0.000000   Min.   :0.000000   Min.   :0.000000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000
Median :0.000000   Median :0.000000   Median :0.000000
   Median :0.000000   Median :0.000000   Median :0.000000
Mean   :0.007573   Mean   :0.002423   Mean   :0.004639   Mean
     :0.008932   Mean   :0.005202   Mean   :0.005868
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd
   Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000
Max.   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
     :1.000000   Max.   :1.000000   Max.   :1.000000

    person8             person9             person10
      person11            person12            person13
Min.   :0.000000   Min.   :0.000000   Min.   :0.00000   Min.
     :0.0000000   Min.   :0.000000   Min.   :0.000000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu
   .:0.0000000   1st Qu.:0.000000   1st Qu.:0.000000
Median :0.000000   Median :0.000000   Median :0.00000   Median
    :0.0000000   Median :0.000000   Median :0.000000
Mean   :0.004683   Mean   :0.003116   Mean   :0.01235   Mean
     :0.0006665   Mean   :0.004085   Mean   :0.005236
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu
   .:0.0000000   3rd Qu.:0.000000   3rd Qu.:0.000000
Max.   :1.000000   Max.   :1.000000   Max.   :1.00000   Max.
     :1.0000000   Max.   :1.000000   Max.   :1.000000
```

```
    person14          person15          person16
       person17          person18          person19
       person20
Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.
    :0.000000   Min.   :0.00000   Min.   :0.000000   Min.
    :0.000000
1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
    .:0.000000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
    .:0.000000
Median :0.00000   Median :0.00000   Median :0.000000   Median
    :0.000000   Median :0.00000   Median :0.000000   Median
    :0.000000
Mean   :0.01033   Mean   :0.00431   Mean   :0.009876   Mean
    :0.005401   Mean   :0.00309   Mean   :0.008058   Mean
    :0.007617
3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
    .:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
    .:0.000000
Max.   :1.00000   Max.   :1.00000   Max.   :1.000000   Max.
    :1.000000   Max.   :1.00000   Max.   :1.000000   Max.
    :1.000000

    person21          person22          person23
       person24          person25          person26
       person27
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
     :0.000000   Min.   :0.00000   Min.   :0.00000   Min.
    :0.000000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu
    .:0.000000
```

```
Median :0.000000    Median :0.000000    Median :0.000000
   Median :0.000000    Median :0.00000    Median :0.00000
   Median :0.000000
Mean    :0.003358    Mean    :0.004631    Mean    :0.003722    Mean
     :0.002051    Mean    :0.00592    Mean    :0.01623    Mean
   :0.009175
3rd Qu.:0.000000    3rd Qu.:0.000000    3rd Qu.:0.000000    3rd
   Qu.:0.000000    3rd Qu.:0.00000    3rd Qu.:0.00000    3rd Qu
   .:0.000000
Max.    :1.000000    Max.    :1.000000    Max.    :1.000000    Max.
     :1.000000    Max.    :1.00000    Max.    :1.00000    Max.
   :1.000000

   person28          person29          person30
      person31          person32          person33
                person34
Min.    :0.0000    Min.    :0.00000    Min.    :0.00000    Min.
   :0.000000    Min.    :0.000000    Min.    :0.000000    Min.
   :0.000000
1st Qu.:0.0000    1st Qu.:0.00000    1st Qu.:0.00000    1st Qu
   .:0.000000    1st Qu.:0.000000    1st Qu.:0.000000    1st Qu
   .:0.000000
Median :0.0000    Median :0.00000    Median :0.00000    Median
   :0.000000    Median :0.000000    Median :0.000000    Median
   :0.000000
Mean    :0.0102    Mean    :0.01147    Mean    :0.01574    Mean
   :0.005202    Mean    :0.003436    Mean    :0.002848    Mean
   :0.004406
3rd Qu.:0.0000    3rd Qu.:0.00000    3rd Qu.:0.00000    3rd Qu
   .:0.000000    3rd Qu.:0.000000    3rd Qu.:0.000000    3rd Qu
   .:0.000000
```

```
Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.
   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
   :1.000000

    person35          person36          person37
       person38          person39          person40
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
      :0.000000   Min.   :0.000000   Min.   :0.00000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.00000
Median :0.000000   Median :0.000000   Median :0.000000
   Median :0.000000   Median :0.000000   Median :0.00000
Mean   :0.008664   Mean   :0.003523   Mean   :0.002692   Mean
      :0.008543   Mean   :0.007617   Mean   :0.01186
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd
   Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.00000
Max.   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
      :1.000000   Max.   :1.000000   Max.   :1.00000

    person41          person42          person43
       person44          person45          person46
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
      :0.000000   Min.   :0.000000   Min.   :0.000000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000
Median :0.000000   Median :0.000000   Median :0.000000
   Median :0.000000   Median :0.000000   Median :0.000000
Mean   :0.008205   Mean   :0.005756   Mean   :0.008127   Mean
      :0.003523   Mean   :0.007617   Mean   :0.001471
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd
   Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000
```

```
Max.   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
      :1.000000   Max.   :1.000000   Max.   :1.000000

    person47          person48          person49
       person50          person51          person52
       person53
Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.
    :0.00000   Min.   :0.000000   Min.   :0.000000   Min.
    :0.000000
1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
    .:0.00000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu
    .:0.000000
Median :0.00000   Median :0.00000   Median :0.000000   Median
    :0.00000   Median :0.000000   Median :0.000000   Median
    :0.000000
Mean   :0.01146   Mean   :0.02476   Mean   :0.006405   Mean
    :0.00882   Mean   :0.005176   Mean   :0.007245   Mean
    :0.002216
3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
    .:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu
    .:0.000000
Max.   :1.00000   Max.   :1.00000   Max.   :1.000000   Max.
    :1.00000   Max.   :1.000000   Max.   :1.000000   Max.
    :1.000000

    person54          person55          person56
       person57          person58          person59
       person60
Min.   :0.0000   Min.   :0.000000   Min.   :0.000000   Min.
    :0.000000   Min.   :0.00000   Min.   :0.000000   Min.
    :0.000000
```

```
1st Qu.:0.0000    1st Qu.:0.000000    1st Qu.:0.000000    1st Qu
    .:0.000000    1st Qu.:0.00000    1st Qu.:0.000000    1st Qu
    .:0.000000
Median :0.0000    Median :0.000000    Median :0.000000    Median
    :0.000000    Median :0.00000    Median :0.000000    Median
    :0.000000
Mean    :0.0074    Mean    :0.007227    Mean    :0.008231    Mean
    :0.007816    Mean    :0.01136    Mean    :0.005609    Mean
    :0.002363
3rd Qu.:0.0000    3rd Qu.:0.000000    3rd Qu.:0.000000    3rd Qu
    .:0.000000    3rd Qu.:0.00000    3rd Qu.:0.000000    3rd Qu
    .:0.000000
Max.    :1.0000    Max.    :1.000000    Max.    :1.000000    Max.
    :1.000000    Max.    :1.00000    Max.    :1.000000    Max.
    :1.000000

    person61            person62            person63
       person64            person65            person66
                  person67
Min.    :0.00000    Min.    :0.00000    Min.    :0.000000    Min.
    :0.000000    Min.    :0.000000    Min.    :0.000000    Min.
    :0.000000
1st Qu.:0.00000    1st Qu.:0.00000    1st Qu.:0.000000    1st Qu
    .:0.000000    1st Qu.:0.000000    1st Qu.:0.000000    1st Qu
    .:0.000000
Median :0.00000    Median :0.00000    Median :0.000000    Median
    :0.000000    Median :0.000000    Median :0.000000    Median
    :0.000000
Mean    :0.01465    Mean    :0.01073    Mean    :0.006976    Mean
    :0.001082    Mean    :0.002882    Mean    :0.002155    Mean
    :0.002311
```

```
3rd Qu.:0.00000    3rd Qu.:0.00000    3rd Qu.:0.000000    3rd Qu
   .:0.000000    3rd Qu.:0.000000    3rd Qu.:0.000000    3rd Qu
   .:0.000000
Max.    :1.00000    Max.    :1.00000    Max.    :1.000000    Max.
   :1.000000    Max.    :1.000000    Max.    :1.000000    Max.
   :1.000000

    person68             person69             person70
       person71             person72             person73
Min.    :0.000000    Min.    :0.00000    Min.    :0.000000    Min.
      :0.00000    Min.    :0.000000    Min.    :0.000000
1st Qu.:0.000000    1st Qu.:0.00000    1st Qu.:0.000000    1st Qu
   .:0.00000    1st Qu.:0.000000    1st Qu.:0.000000
Median :0.000000    Median :0.00000    Median :0.000000    Median
    :0.00000    Median :0.000000    Median :0.000000
Mean    :0.001965    Mean    :0.00959    Mean    :0.008413    Mean
      :0.00901    Mean    :0.002804    Mean    :0.005184
3rd Qu.:0.000000    3rd Qu.:0.00000    3rd Qu.:0.000000    3rd Qu
   .:0.00000    3rd Qu.:0.000000    3rd Qu.:0.000000
Max.    :1.000000    Max.    :1.00000    Max.    :1.000000    Max.
      :1.00000    Max.    :1.000000    Max.    :1.000000

    person74             person75             person76
       person77             person78             person79
Min.    :0.0000000    Min.    :0.00000    Min.    :0.000000    Min.
      :0.000000    Min.    :0.000000    Min.    :0.000000
1st Qu.:0.0000000    1st Qu.:0.00000    1st Qu.:0.000000    1st
   Qu.:0.000000    1st Qu.:0.000000    1st Qu.:0.000000
Median :0.0000000    Median :0.00000    Median :0.000000
   Median :0.000000    Median :0.000000    Median :0.000000
Mean    :0.0008223    Mean    :0.00425    Mean    :0.006509    Mean
      :0.004345    Mean    :0.008526    Mean    :0.004302
```

```
3rd Qu.:0.0000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd
   Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000
Max.   :1.0000000   Max.   :1.00000   Max.   :1.000000   Max.
      :1.000000   Max.   :1.000000   Max.   :1.000000

   person80            person81            person82
      person83            person84            person85
      person86
Min.   :0.000000   Min.   :0.000000   Min.   :0.000   Min.
   :0.000000   Min.   :0.00000   Min.   :0.000000   Min.
   :0.00000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000   1st Qu
   .:0.000000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
   .:0.00000
Median :0.000000   Median :0.000000   Median :0.000   Median
   :0.000000   Median :0.00000   Median :0.000000   Median
   :0.00000
Mean   :0.003938   Mean   :0.005427   Mean   :0.013   Mean
   :0.008231   Mean   :0.01206   Mean   :0.006665   Mean
   :0.00927
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000   3rd Qu
   .:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
   .:0.00000
Max.   :1.000000   Max.   :1.000000   Max.   :1.000   Max.
   :1.000000   Max.   :1.00000   Max.   :1.000000   Max.
   :1.00000

   person87            person88            person89
      person90            person91            person92
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
      :0.00000   Min.   :0.000000   Min.   :0.000000
```

```
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000
Median :0.000000   Median :0.000000   Median :0.000000
   Median :0.00000   Median :0.000000   Median :0.000000
Mean   :0.003029   Mean   :0.004181   Mean   :0.005366   Mean
     :0.01441   Mean   :0.006154   Mean   :0.006266
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd
   Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000
Max.   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
     :1.00000   Max.   :1.000000   Max.   :1.000000

   person93          person94          person95
      person96          person97          person98
Min.   :0.000000   Min.   :0.00000   Min.   :0.000000   Min.
     :0.000000   Min.   :0.000000   Min.   :0.000000
1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
   .:0.000000   1st Qu.:0.000000   1st Qu.:0.000000
Median :0.000000   Median :0.00000   Median :0.000000   Median
    :0.000000   Median :0.000000   Median :0.000000
Mean   :0.004155   Mean   :0.03102   Mean   :0.007539   Mean
     :0.003488   Mean   :0.005176   Mean   :0.004864
3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
   .:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000
Max.   :1.000000   Max.   :1.00000   Max.   :1.000000   Max.
     :1.000000   Max.   :1.000000   Max.   :1.000000

   person99          person100          person101
      person102          person103          person104
      person105
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
     :0.00000   Min.   :0.000000   Min.   :0.00000   Min.
   :0.000000
```

```
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu
   .:0.000000
Median :0.000000   Median :0.000000   Median :0.000000
   Median :0.00000   Median :0.000000   Median :0.00000
   Median :0.000000
Mean   :0.007011   Mean   :0.005089   Mean   :0.004163   Mean
     :0.00521   Mean   :0.003852   Mean   :0.00438   Mean
   :0.001567
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd
   Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu
   .:0.000000
Max.   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
     :1.00000   Max.   :1.000000   Max.   :1.00000   Max.
   :1.000000

  person106          person107          person108
     person109          person110          person111
     person112
Min.   :0.000000   Min.   :0.00000   Min.   :0.00000   Min.
   :0.00000   Min.   :0.000000   Min.   :0.000000   Min.
   :0.000000
1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu
   .:0.00000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu
   .:0.000000
Median :0.000000   Median :0.00000   Median :0.00000   Median
   :0.00000   Median :0.000000   Median :0.000000   Median
   :0.000000
Mean   :0.002112   Mean   :0.00167   Mean   :0.01035   Mean
   :0.00335   Mean   :0.008422   Mean   :0.004293   Mean
   :0.002475
```

```
3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu
   .:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu
   .:0.000000
Max.   :1.000000   Max.   :1.00000   Max.   :1.00000   Max.
   :1.00000   Max.   :1.000000   Max.   :1.000000   Max.
   :1.000000

  person113          person114          person115
     person116          person117          person118
     person119
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
     :0.000000   Min.   :0.00000   Min.   :0.000000   Min.
   :0.00000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
   .:0.00000
Median :0.000000   Median :0.000000   Median :0.000000
   Median :0.000000   Median :0.00000   Median :0.000000
   Median :0.00000
Mean   :0.006457   Mean   :0.002077   Mean   :0.004899   Mean
     :0.009235   Mean   :0.00206   Mean   :0.006743   Mean
   :0.01155
3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd
   Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
   .:0.00000
Max.   :1.000000   Max.   :1.000000   Max.   :1.000000   Max.
     :1.000000   Max.   :1.00000   Max.   :1.000000   Max.
   :1.00000

  person120          person121          person122
     person123          person124          person125
     person126
```

```
Min.   :0.00000   Min.   :0.000000   Min.   :0.000000   Min.
     :0.00000   Min.   :0.00000   Min.   :0.000000   Min.
   :0.000000
1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu
   .:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
   .:0.000000
Median :0.00000   Median :0.000000   Median :0.000000   Median
    :0.00000   Median :0.00000   Median :0.000000   Median
   :0.000000
Mean   :0.01017   Mean   :0.004354   Mean   :0.009902   Mean
     :0.01721   Mean   :0.02052   Mean   :0.005133   Mean
   :0.003817
3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu
   .:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
   .:0.000000
Max.   :1.00000   Max.   :1.000000   Max.   :1.000000   Max.
     :1.00000   Max.   :1.00000   Max.   :1.000000   Max.
   :1.000000

  person127          person128          person129
     person130          person131          person132
     person133
Min.   :0.00000   Min.   :0.0e+00   Min.   :0.000000   Min.
   :0.000000   Min.   :0.00000   Min.   :0.000000   Min.
   :0.00000
1st Qu.:0.00000   1st Qu.:0.0e+00   1st Qu.:0.000000   1st Qu
   .:0.000000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu
   .:0.00000
Median :0.00000   Median :0.0e+00   Median :0.000000   Median
   :0.000000   Median :0.00000   Median :0.000000   Median
   :0.00000
```

```
Mean   :0.00682   Mean   :8.7e-06   Mean   :0.006967   Mean
    :0.004042   Mean   :0.00592   Mean   :0.004838   Mean
    :0.00901
3rd Qu.:0.00000   3rd Qu.:0.0e+00   3rd Qu.:0.000000   3rd Qu
    .:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu
    .:0.00000
Max.   :1.00000   Max.   :1.0e+00   Max.   :1.000000   Max.
    :1.000000   Max.   :1.00000   Max.   :1.000000   Max.
    :1.00000

  person134          person135          person136
     person137          person138          person139
Min.   :0.000000   Min.   :0.0000000   Min.   :0.000000   Min.
      :0.000000   Min.   :0.000000   Min.   :0.00000
1st Qu.:0.000000   1st Qu.:0.0000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.00000
Median :0.000000   Median :0.0000000   Median :0.000000
   Median :0.000000   Median :0.000000   Median :0.00000
Mean   :0.005548   Mean   :0.0004501   Mean   :0.007824   Mean
      :0.005505   Mean   :0.002397   Mean   :0.01168
3rd Qu.:0.000000   3rd Qu.:0.0000000   3rd Qu.:0.000000   3rd
   Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.00000
Max.   :1.000000   Max.   :1.0000000   Max.   :1.000000   Max.
      :1.000000   Max.   :1.000000   Max.   :1.00000

  person140          person141          person142
     person143          person144          person145
Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.
      :0.000000   Min.   :0.000000   Min.   :0.000000
1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st
   Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000
```

```
Median :0.000000    Median :0.000000    Median :0.000000
   Median :0.000000    Median :0.000000    Median :0.000000
Mean    :0.004371    Mean    :0.008145    Mean    :0.004527    Mean
     :0.004778    Mean    :0.008854    Mean    :0.006717
3rd Qu.:0.000000    3rd Qu.:0.000000    3rd Qu.:0.000000    3rd
   Qu.:0.000000    3rd Qu.:0.000000    3rd Qu.:0.000000
Max.    :1.000000    Max.    :1.000000    Max.    :1.000000    Max.
     :1.000000    Max.    :1.000000    Max.    :1.000000

  person146           person147           person148
     person149           person150           person151
Min.    :0.000000    Min.    :0.000000    Min.    :0.00000    Min.
     :0.000000    Min.    :0.000000    Min.    :0.000000
1st Qu.:0.000000    1st Qu.:0.000000    1st Qu.:0.00000    1st Qu
   .:0.000000    1st Qu.:0.000000    1st Qu.:0.000000
Median :0.000000    Median :0.000000    Median :0.00000    Median
    :0.000000    Median :0.000000    Median :0.000000
Mean    :0.005479    Mean    :0.008482    Mean    :0.00631    Mean
     :0.004622    Mean    :0.003453    Mean    :0.004631
3rd Qu.:0.000000    3rd Qu.:0.000000    3rd Qu.:0.00000    3rd Qu
   .:0.000000    3rd Qu.:0.000000    3rd Qu.:0.000000
Max.    :1.000000    Max.    :1.000000    Max.    :1.00000    Max.
     :1.000000    Max.    :1.000000    Max.    :1.000000

> #Multi Linear Regression
```

```
> regression =lm(degree~streetlength+speed_25+speed_30+speed_35
  +speed_40+speed_45+speed_50+speed_55+speed_60+speed_65+
  age_25_to_34+age_35_to_44+age_45_to_54+age_55_to_64+
  age_65_to_74+age_75_to_84+age_85_and_over+gender+education_1
  +education_2+morning+afternoon+night+person1+person2+person3
  +person4+person5+person6+person7+person8+person9+person10+
  person11+person12+person13+person14+person15+person16+
  person17+person18+person19+person20+person21+person22+
  person23+person24+person25+person26+person27+person28+
  person29+person30+person31+person32+person33+person34+
  person35+person36+person37+person38+person39+person40+
  person41+person42+person43+person44+person45+person46+
  person47+person48+person49+person50+person51+person52+
  person53+person54+person55+person56+person57+person58+
  person59+person60+person61+person62+person63+person64+
  person65+person66+person67+person68+person69+person70+
  person71+person72+person73+person74+person75+person76+
  person77+person78+person79+person80+person81+person82+
  person83+person84+person85+person86+person87+person88+
  person89+person90+person91+person92+person93+person94+
  person95+person96+person97+person98+person99+person100+
  person101+person102+person103+person104+person105+person106+
  person107+person108+person109+person110+person111+person112+
  person113+person114+person115+person116+person117+person118+
  person119+person120+person121+person122+person123+person124+
  person125+person126+person127+person128+person129+person130+
  person131+person132+person133+person134+person135+person136+
  person137+person138+person139+person140+person141+person142+
  person143+person144+person145+person146+person147+person148+
  person149+person150+person151,res)
> summary(regression)
```

```
Call:
lm(formula = degree ~ streetlength + speed_25 + speed_30 +
    speed_35 +
     speed_40 + speed_45 + speed_50 + speed_55 + speed_60 +
        speed_65 +
     age_25_to_34 + age_35_to_44 + age_45_to_54 + age_55_to_64 +
     age_65_to_74 + age_75_to_84 + age_85_and_over + gender +
     education_1 + education_2 + morning + afternoon + night +
     person1 + person2 + person3 + person4 + person5 + person6 +
     person7 + person8 + person9 + person10 + person11 +
        person12 +
     person13 + person14 + person15 + person16 + person17 +
        person18 +
     person19 + person20 + person21 + person22 + person23 +
        person24 +
     person25 + person26 + person27 + person28 + person29 +
        person30 +
     person31 + person32 + person33 + person34 + person35 +
        person36 +
     person37 + person38 + person39 + person40 + person41 +
        person42 +
     person43 + person44 + person45 + person46 + person47 +
        person48 +
     person49 + person50 + person51 + person52 + person53 +
        person54 +
     person55 + person56 + person57 + person58 + person59 +
        person60 +
     person61 + person62 + person63 + person64 + person65 +
        person66 +
     person67 + person68 + person69 + person70 + person71 +
        person72 +
```

```
    person73 + person74 + person75 + person76 + person77 +
       person78 +
    person79 + person80 + person81 + person82 + person83 +
       person84 +
    person85 + person86 + person87 + person88 + person89 +
       person90 +
    person91 + person92 + person93 + person94 + person95 +
       person96 +
    person97 + person98 + person99 + person100 + person101 +
    person102 + person103 + person104 + person105 + person106 +
    person107 + person108 + person109 + person110 + person111 +
    person112 + person113 + person114 + person115 + person116 +
    person117 + person118 + person119 + person120 + person121 +
    person122 + person123 + person124 + person125 + person126 +
    person127 + person128 + person129 + person130 + person131 +
    person132 + person133 + person134 + person135 + person136 +
    person137 + person138 + person139 + person140 + person141 +
    person142 + person143 + person144 + person145 + person146 +
    person147 + person148 + person149 + person150 + person151,
    data = res)

Residuals:
    Min      1Q  Median      3Q     Max
-1.3912 -0.1497  0.0225  0.1530  6.1722

Coefficients: (10 not defined because of singularities)
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     8.386e-01  2.390e-02  35.082  < 2e-16 ***
streetlength    1.280e-04  3.685e-06  34.736  < 2e-16 ***
speed_25        6.587e-01  1.238e-02  53.206  < 2e-16 ***
speed_30       -3.703e-02  1.185e-02  -3.125 0.001781 **
speed_35       -1.177e-01  1.158e-02 -10.162  < 2e-16 ***
```

| | | | | | |
|---|---|---|---|---|---|
| speed_40 | −8.413e−03 | 1.157e−02 | −0.727 | 0.467026 | |
| speed_45 | −3.285e−03 | 1.286e−02 | −0.255 | 0.798432 | |
| speed_50 | 8.432e−03 | 1.326e−02 | 0.636 | 0.524808 | |
| speed_55 | 1.056e−01 | 1.152e−02 | 9.164 | < 2e−16 | *** |
| speed_60 | 9.919e−02 | 1.162e−02 | 8.538 | < 2e−16 | *** |
| speed_65 | 3.506e−02 | 1.242e−02 | 2.823 | 0.004758 | ** |
| age_25_to_34 | 7.488e−02 | 1.509e−02 | 4.963 | 6.94e−07 | *** |
| age_35_to_44 | −1.055e−01 | 2.683e−02 | −3.933 | 8.38e−05 | *** |
| age_45_to_54 | −2.184e−01 | 2.572e−02 | −8.490 | < 2e−16 | *** |
| age_55_to_64 | −3.575e−02 | 2.363e−02 | −1.513 | 0.130242 | |
| age_65_to_74 | −1.572e−01 | 2.140e−02 | −7.345 | 2.06e−13 | *** |
| age_75_to_84 | −5.696e−02 | 1.451e−02 | −3.927 | 8.60e−05 | *** |
| age_85_and_over | −3.847e−02 | 2.611e−02 | −1.474 | 0.140547 | |
| gender | −1.026e−02 | 1.770e−02 | −0.580 | 0.562177 | |
| education_1 | 1.520e−01 | 1.685e−02 | 9.020 | < 2e−16 | *** |
| education_2 | 1.347e−01 | 2.882e−02 | 4.676 | 2.93e−06 | *** |
| morning | −3.225e−02 | 6.823e−03 | −4.726 | 2.29e−06 | *** |
| afternoon | −3.566e−02 | 6.786e−03 | −5.254 | 1.49e−07 | *** |
| night | −2.295e−02 | 6.986e−03 | −3.285 | 0.001019 | ** |
| person1 | 8.224e−02 | 1.385e−02 | 5.936 | 2.93e−09 | *** |
| person2 | −3.585e−01 | 4.104e−02 | −8.735 | < 2e−16 | *** |
| person3 | −2.376e−01 | 3.023e−02 | −7.859 | 3.90e−15 | *** |
| person4 | −1.397e−01 | 2.816e−02 | −4.961 | 7.03e−07 | *** |
| person5 | −8.730e−02 | 2.141e−02 | −4.077 | 4.57e−05 | *** |
| person6 | 8.727e−02 | 2.382e−02 | 3.663 | 0.000249 | *** |
| person7 | −5.351e−02 | 2.767e−02 | −1.934 | 0.053160 | . |
| person8 | 1.706e−02 | 1.635e−02 | 1.043 | 0.296725 | |
| person9 | −2.118e−01 | 2.885e−02 | −7.342 | 2.12e−13 | *** |
| person10 | −1.365e−01 | 2.645e−02 | −5.162 | 2.45e−07 | *** |
| person11 | −3.694e−01 | 5.026e−02 | −7.349 | 2.01e−13 | *** |
| person12 | −1.909e−01 | 1.673e−02 | −11.414 | < 2e−16 | *** |
| person13 | −1.762e−01 | 1.570e−02 | −11.223 | < 2e−16 | *** |

| | | | | | |
|---|---|---|---|---|---|
| person14 | 3.327e−02 | 2.254e−02 | 1.476 | 0.139913 | |
| person15 | 4.261e−02 | 1.668e−02 | 2.554 | 0.010644 | * |
| person16 | −1.367e−01 | 4.077e−02 | −3.354 | 0.000797 | *** |
| person17 | 5.759e−02 | 2.231e−02 | 2.581 | 0.009850 | ** |
| person18 | −1.470e−01 | 3.445e−02 | −4.267 | 1.98e−05 | *** |
| person19 | −2.957e−01 | 2.274e−02 | −13.006 | < 2e−16 | *** |
| person20 | −6.261e−02 | 2.196e−02 | −2.851 | 0.004361 | ** |
| person21 | −6.608e−02 | 2.861e−02 | −2.310 | 0.020911 | * |
| person22 | −1.003e−01 | 1.636e−02 | −6.134 | 8.59e−10 | *** |
| person23 | −1.187e−01 | 1.716e−02 | −6.919 | 4.56e−12 | *** |
| person24 | −4.141e−02 | 2.727e−02 | −1.518 | 0.128919 | |
| person25 | −1.422e−01 | 2.766e−02 | −5.142 | 2.73e−07 | *** |
| person26 | −1.782e−01 | 4.047e−02 | −4.405 | 1.06e−05 | *** |
| person27 | 9.587e−02 | 2.346e−02 | 4.086 | 4.39e−05 | *** |
| person28 | −5.970e−02 | 1.395e−02 | −4.280 | 1.87e−05 | *** |
| person29 | −3.586e−02 | 1.350e−02 | −2.656 | 0.007915 | ** |
| person30 | 6.698e−02 | 2.057e−02 | 3.256 | 0.001129 | ** |
| person31 | 1.175e−01 | 2.247e−02 | 5.232 | 1.68e−07 | *** |
| person32 | −1.328e−01 | 4.219e−02 | −3.147 | 0.001649 | ** |
| person33 | −1.934e−01 | 2.543e−02 | −7.604 | 2.88e−14 | *** |
| person34 | −1.760e−01 | 1.603e−02 | −10.983 | < 2e−16 | *** |
| person35 | −4.660e−02 | 1.302e−02 | −3.580 | 0.000343 | *** |
| person36 | −1.517e−02 | 1.858e−02 | −0.817 | 0.414151 | |
| person37 | 6.025e−02 | 2.669e−02 | 2.258 | 0.023970 | * |
| person38 | −1.463e−01 | 3.257e−02 | −4.491 | 7.11e−06 | *** |
| person39 | −1.074e−01 | 2.747e−02 | −3.911 | 9.21e−05 | *** |
| person40 | −2.098e−01 | 1.309e−02 | −16.033 | < 2e−16 | *** |
| person41 | 4.365e−02 | 1.208e−02 | 3.612 | 0.000304 | *** |
| person42 | 5.804e−02 | 2.217e−02 | 2.618 | 0.008833 | ** |
| person43 | NA | NA | NA | NA | |
| person44 | −8.501e−02 | 1.572e−02 | −5.409 | 6.33e−08 | *** |
| person45 | −6.978e−02 | 2.156e−02 | −3.237 | 0.001209 | ** |

| | | | | | |
|---|---|---|---|---|---|
| person46 | 2.890e−01 | 3.176e−02 | 9.099 | < 2e−16 | *** |
| person47 | 6.476e−02 | 2.697e−02 | 2.401 | 0.016347 | * |
| person48 | 1.977e−01 | 2.470e−02 | 8.003 | 1.22e−15 | *** |
| person49 | −1.747e−02 | 2.336e−02 | −0.748 | 0.454528 | |
| person50 | −4.392e−02 | 1.430e−02 | −3.071 | 0.002133 | ** |
| person51 | 3.185e−02 | 1.533e−02 | 2.078 | 0.037705 | * |
| person52 | −1.023e−01 | 2.728e−02 | −3.750 | 0.000177 | *** |
| person53 | 7.841e−03 | 4.337e−02 | 0.181 | 0.856517 | |
| person54 | 1.065e−01 | 1.244e−02 | 8.561 | < 2e−16 | *** |
| person55 | 8.024e−02 | 2.169e−02 | 3.699 | 0.000217 | *** |
| person56 | −1.941e−01 | 2.274e−02 | −8.537 | < 2e−16 | *** |
| person57 | −1.035e−01 | 1.441e−02 | −7.183 | 6.86e−13 | *** |
| person58 | −1.395e−01 | 1.300e−02 | −10.733 | < 2e−16 | *** |
| person59 | −3.652e−02 | 1.562e−02 | −2.339 | 0.019353 | * |
| person60 | 5.077e−02 | 2.000e−02 | 2.539 | 0.011115 | * |
| person61 | −2.185e−01 | 2.817e−02 | −7.756 | 8.84e−15 | *** |
| person62 | 4.141e−02 | 1.510e−02 | 2.742 | 0.006112 | ** |
| person63 | −6.742e−02 | 1.491e−02 | −4.522 | 6.14e−06 | *** |
| person64 | 8.739e−02 | 3.194e−02 | 2.736 | 0.006213 | ** |
| person65 | 9.984e−02 | 1.691e−02 | 5.906 | 3.52e−09 | *** |
| person66 | −6.332e−02 | 2.641e−02 | −2.398 | 0.016508 | * |
| person67 | −3.954e−02 | 2.115e−02 | −1.869 | 0.061576 | . |
| person68 | −1.039e−02 | 2.121e−02 | −0.490 | 0.624268 | |
| person69 | −2.376e−01 | 2.676e−02 | −8.878 | < 2e−16 | *** |
| person70 | 1.313e−01 | 2.141e−02 | 6.131 | 8.76e−10 | *** |
| person71 | 4.016e−02 | 1.182e−02 | 3.398 | 0.000678 | *** |
| person72 | −3.664e−01 | 3.123e−02 | −11.732 | < 2e−16 | *** |
| person73 | −4.029e−01 | 2.367e−02 | −17.022 | < 2e−16 | *** |
| person74 | −9.615e−02 | 3.697e−02 | −2.601 | 0.009303 | ** |
| person75 | −2.107e−01 | 3.367e−02 | −6.258 | 3.92e−10 | *** |
| person76 | −3.576e−02 | 1.497e−02 | −2.390 | 0.016869 | * |
| person77 | −1.457e−01 | 2.462e−02 | −5.917 | 3.29e−09 | *** |

| | | | | | |
|---|---|---|---|---|---|
| person78 | 4.996e−02 | 1.193e−02 | 4.188 | 2.82e−05 | *** |
| person79 | 7.623e−03 | 2.296e−02 | 0.332 | 0.739919 | |
| person80 | −1.377e−01 | 3.726e−02 | −3.697 | 0.000218 | *** |
| person81 | −1.418e−01 | 2.766e−02 | −5.127 | 2.94e−07 | *** |
| person82 | −7.100e−02 | 2.211e−02 | −3.211 | 0.001323 | ** |
| person83 | −6.847e−02 | 1.427e−02 | −4.799 | 1.60e−06 | *** |
| person84 | −3.327e−02 | 4.062e−02 | −0.819 | 0.412738 | |
| person85 | −1.575e−02 | 2.743e−02 | −0.574 | 0.565958 | |
| person86 | 1.221e−02 | 1.418e−02 | 0.861 | 0.389079 | |
| person87 | −2.053e−02 | 2.479e−02 | −0.828 | 0.407570 | |
| person88 | 1.361e−01 | 2.304e−02 | 5.907 | 3.50e−09 | *** |
| person89 | 4.039e−02 | 2.234e−02 | 1.807 | 0.070698 | . |
| person90 | −5.762e−02 | 1.312e−02 | −4.393 | 1.12e−05 | *** |
| person91 | 2.052e−02 | 2.415e−02 | 0.850 | 0.395438 | |
| person92 | 4.655e−02 | 1.641e−02 | 2.836 | 0.004574 | ** |
| person93 | NA | NA | NA | NA | |
| person94 | −9.194e−02 | 2.607e−02 | −3.527 | 0.000420 | *** |
| person95 | −3.116e−02 | 2.258e−02 | −1.380 | 0.167594 | |
| person96 | −5.631e−03 | 1.763e−02 | −0.319 | 0.749409 | |
| person97 | 3.557e−02 | 2.388e−02 | 1.490 | 0.136314 | |
| person98 | 6.141e−03 | 2.398e−02 | 0.256 | 0.797859 | |
| person99 | −6.040e−02 | 2.735e−02 | −2.209 | 0.027197 | * |
| person100 | 3.813e−03 | 2.387e−02 | 0.160 | 0.873055 | |
| person101 | −7.372e−02 | 4.181e−02 | −1.763 | 0.077881 | . |
| person102 | −5.340e−02 | 2.790e−02 | −1.914 | 0.055633 | . |
| person103 | 7.016e−02 | 1.522e−02 | 4.611 | 4.02e−06 | *** |
| person104 | 4.638e−02 | 2.352e−02 | 1.972 | 0.048571 | * |
| person105 | 9.160e−02 | 2.641e−02 | 3.468 | 0.000524 | *** |
| person106 | −3.594e−02 | 3.483e−02 | −1.032 | 0.302070 | |
| person107 | −4.507e−01 | 3.144e−02 | −14.335 | < 2e−16 | *** |
| person108 | 1.246e−01 | 2.107e−02 | 5.915 | 3.33e−09 | *** |
| person109 | −8.831e−02 | 3.765e−02 | −2.346 | 0.018985 | * |

| | | | | | |
|---|---|---|---|---|---|
| person110 | 9.840e−02 | 2.772e−02 | 3.550 | 0.000386 | *** |
| person111 | −2.075e−01 | 2.470e−02 | −8.400 | < 2e−16 | *** |
| person112 | −1.366e−01 | 3.017e−02 | −4.529 | 5.95e−06 | *** |
| person113 | −6.209e−02 | 4.119e−02 | −1.507 | 0.131701 | |
| person114 | −1.552e−02 | 3.097e−02 | −0.501 | 0.616191 | |
| person115 | −1.417e−01 | 2.805e−02 | −5.053 | 4.35e−07 | *** |
| person116 | 7.728e−02 | 2.270e−02 | 3.404 | 0.000663 | *** |
| person117 | 5.878e−02 | 2.729e−02 | 2.154 | 0.031259 | * |
| person118 | 6.930e−02 | 1.485e−02 | 4.668 | 3.05e−06 | *** |
| person119 | −1.189e−01 | 2.671e−02 | −4.451 | 8.54e−06 | *** |
| person120 | 2.940e−02 | 2.257e−02 | 1.303 | 0.192656 | |
| person121 | −1.247e−01 | 1.649e−02 | −7.560 | 4.07e−14 | *** |
| person122 | −3.562e−01 | 2.247e−02 | −15.848 | < 2e−16 | *** |
| person123 | 1.281e−01 | 3.155e−02 | 4.058 | 4.95e−05 | *** |
| person124 | 9.131e−02 | 1.822e−02 | 5.012 | 5.40e−07 | *** |
| person125 | −9.923e−03 | 2.820e−02 | −0.352 | 0.724897 | |
| person126 | 3.512e−02 | 2.456e−02 | 1.430 | 0.152618 | |
| person127 | 2.766e−01 | 3.146e−02 | 8.792 | < 2e−16 | *** |
| person128 | −7.630e−01 | 2.680e−01 | −2.847 | 0.004419 | ** |
| person129 | NA | NA | NA | NA | |
| person130 | 2.102e−02 | 3.722e−02 | 0.565 | 0.572222 | |
| person131 | 7.813e−02 | 2.336e−02 | 3.344 | 0.000825 | *** |
| person132 | NA | NA | NA | NA | |
| person133 | −1.086e−01 | 1.352e−02 | −8.032 | 9.71e−16 | *** |
| person134 | 3.639e−03 | 1.447e−02 | 0.251 | 0.801465 | |
| person135 | −9.218e−02 | 4.189e−02 | −2.200 | 0.027783 | * |
| person136 | −1.304e−01 | 2.717e−02 | −4.799 | 1.60e−06 | *** |
| person137 | −2.016e−01 | 1.516e−02 | −13.297 | < 2e−16 | *** |
| person138 | −8.496e−02 | 3.014e−02 | −2.819 | 0.004818 | ** |
| person139 | −1.771e−01 | 2.668e−02 | −6.638 | 3.19e−11 | *** |
| person140 | 9.608e−02 | 2.292e−02 | 4.192 | 2.77e−05 | *** |
| person141 | −2.037e−01 | 2.711e−02 | −7.512 | 5.85e−14 | *** |

```
person142       -9.676e-02  1.584e-02  -6.108 1.01e-09 ***
person143       -1.203e-01  4.158e-02  -2.894 0.003801 **
person144               NA         NA      NA       NA
person145        1.800e-01  2.807e-02   6.411 1.45e-10 ***
person146               NA         NA      NA       NA
person147       -4.584e-02  2.363e-02  -1.940 0.052383 .
person148               NA         NA      NA       NA
person149               NA         NA      NA       NA
person150               NA         NA      NA       NA
person151               NA         NA      NA       NA
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1         1

Residual standard error: 0.2671 on 115371 degrees of freedom
Multiple R-squared:  0.2851,    Adjusted R-squared:  0.2841
F-statistic: 280.5 on 164 and 115371 DF,  p-value: < 2.2e-16

> write.csv (summary(regression)$coef, "regression.csv")
>
> #Boxplot
> #Speedlimit
> boxplot(degree ~ Speedlimit, data=res, ylim=c(0, 8), yaxp=c
    (0, 8, 8), xlab="Speed limit(mph)", ylab="driving speed/
    speed limit")
> #Time
> boxplot(degree ~ Hour, data=res, ylim=c(0, 8), yaxp=c(0, 8,
    8), xlab="Time", ylab="driving speed/speed limit")
> #Education
> boxplot(degree ~ o_edu, data=res, ylim=c(0, 8), yaxp=c(0, 8,
    8), xlab="Education", ylab="driving speed/speed limit")
> #Age
```

```
> boxplot(degree ~ o_age, data=res, ylim=c(0, 8), yaxp=c(0, 8,
    8), xlab="Age", ylab="driving speed/speed limit")
> #Gender
> boxplot(degree ~ o_gender, data=res, ylim=c(0, 8), yaxp=c(0,
    8, 8), xlab="Gender", ylab="driving speed/speed limit")
>
> #Plot between street length and speeding
> street_length <- res$streetlength
> speeding <- res$degree
> plot(street_length, speeding,pch=".", ylim=c(0, 8), yaxp=c(0,
     8, 8), xlab="Link length", ylab="driving speed/speed limit
    ")
> lm.obj<-lm(speeding~street_length)
> abline(lm.obj,col=2)
> summary(lm.obj)

Call:
lm(formula = speeding ~ street_length)

Residuals:
    Min      1Q  Median      3Q     Max
-1.3322 -0.1889  0.0193  0.1696  6.8196

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.246e-01  1.249e-03   580.0   <2e-16 ***
street_length 3.043e-04  3.412e-06    89.2   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3053 on 115534 degrees of freedom
```

```
Multiple R-squared:  0.06443,    Adjusted R-squared:  0.06442
F-statistic:  7957 on 1 and 115534 DF,  p-value: < 2.2e-16

> mtext("N=115,536, R^2=0.0644, y=0.7246+0.0003x", side=3)
>
> #Link length is less than 1,000m
> length1000 <- subset(res, res$streetlength <= 1000)
> street_length_1 <- length1000$streetlength
> speeding_1 <- length1000$degree
> lm1000.obj<-lm(speeding_1~street_length_1)
> summary(lm1000.obj)

Call:
lm(formula = speeding_1 ~ street_length_1)

Residuals:
     Min       1Q   Median       3Q      Max
-1.0305  -0.1855   0.0137   0.1677   6.8350

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     6.883e-01  1.425e-03  482.86   <2e-16 ***
street_length_1 4.787e-04  4.812e-06   99.47   <2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05
      .    0.1          1

Residual standard error: 0.3033 on 113124 degrees of freedom
Multiple R-squared:  0.08043,    Adjusted R-squared:  0.08042
F-statistic:  9894 on 1 and 113124 DF,  p-value: < 2.2e-16
```

```
> plot(street_length_1, speeding_1,pch=".", ylim=c(0, 8), yaxp=
  c(0, 8, 8), xlab="Link length", ylab="driving speed/speed
  limit")
> abline(lm1000.obj,col=2)
> mtext("N=115,536, R^2=0.0804, y=0.6883+0.0005x", side=3)
>
```