This document explains the process of importing data into the TaxiVis system. The current version uses a fixed schema represented by the following struct (found in the KdTrip.hpp file):

```
struct Trip {
        uint32_t   pickup_time;
        uint32_t   dropoff_time;
        float      pickup_long;
        float      pickup_lat;
        float      dropoff_long;
        float      dropoff_lat;
        uint32_t   field1;         // extra field to import data
        uint32_t   field2;         // extra field to import data
        uint32_t   field3;         // extra field to import data
        uint32_t   field4;         // extra field to import data
        uint16_t   id_taxi;
        uint16_t   distance;
        uint16_t   fare_amount;
        uint16_t   surcharge;
        uint16_t   mta_tax;
        uint16_t   tip_amount;
        uint16_t   tolls_amount;
        uint8_t    payment_type;
        uint8_t    passengers;
    };
```

This schema includes the attributes present in the data released by the New York City Taxi & Limousine Commission (TLC). We have also included additional attributes (field1, field2, field3 and field4) to allow for additional data that may be required by different applications.

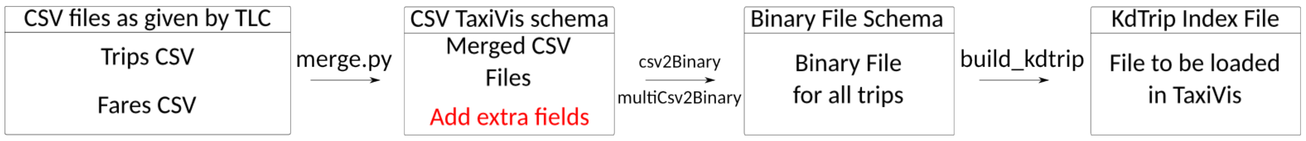


Figure 1: Summary of the data import process.

## 1 Importing the Data and Building the Index

The source code for building the index is in the src/preprocess folder. The data import process is illustrated in Fig. **??**. The inputs are two separate sets of csv files (provided by the TLC), namely trip and fare files. Two sample data files (sample_trip_data_1.csv and sample_trip_fare_1.csv, one of each type) are included in the *data/rawData* folder. The *merge.py* script is used to merge this files into a single csv file. This script also adds the placeholder columns *field1*,*field2*,*field3,* and *field4* (the default values are all zeros, but additional data might be derived and imported here). It is invoked as follows:

```
> python merge.py sample_trip_data_1.csv sample_trip_fare_1.csv merged_test_da
```

The merged filed is then serialized as a binary file. This is done so that the index construction can scale to large volumes of data. There are two programs that can accomplish this job: *csv2Binary* and *multiCsv2Binary*. csv2Binary processes **one** csv file and outputs a corresponding binary file. It is used as follows:

```
./csv2Binary merged_test_data.csv bin_test_data.bin
```

multiCsv2Binary has as input a file containing a **list** of csv files to be processed and outputs a single binary file containing all the trips in the input files. An example of input to multiCsv2Binary is the content of the *fileLocations.txt* file shown in the following:

```
../../data/rawData/data_1.csv
../../data/rawData/data_2.csv
```

The usage multiCsv2Binary is illustrated in the following:

```
./multiCsv2Binary fileLocations.txt bin_test_data.bin
```

Finally, the *build_kdtrip* script that takes as input the binary file containing the trip records and outputs the kdtrip index that is loaded by TaxiVis. The usage is:

```
./build_kdtrip bin_test_data.bin test_data.kdtrip
```

## 2 Parsing files from TLC website

TLC now releases the data at `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`. To parse this data into TaxiVis format, we can use a process similar to the one described in the previous section. In this new format, fare and trip files are already integrated, so there is no need to use the merge script. Instead we use the parser *newFormatCsv2Binary.cpp* as follows

```
././newFormatCsv2Binary yellow_tripdata_2015-01.csv bin_test_data.bin
```

Once this is done, we can proceed as before and use the *build_kdtrip* to build the index to be used in TaxiVis.

## 3 Loading File in TaxiVis

The kdtrip file is loaded by TaxiVis using the constructor of the *QueryManager* class (*querymanager.cpp* file):

```
std::string fname = "/path/to/file/test_data.kdtrip";
```

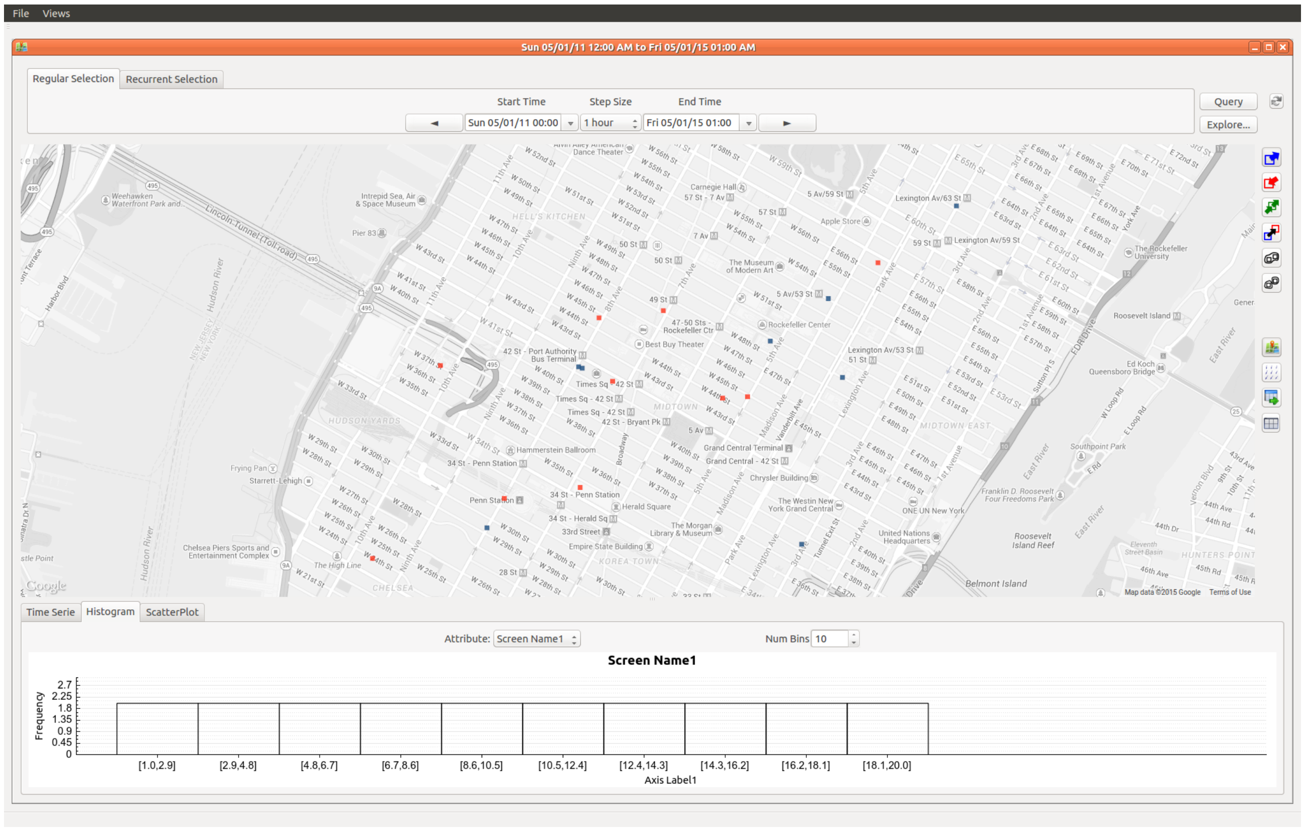
Once this change is made and TaxiVis is run, the file will be loaded.

## 4 Configuring how place holder fields are displayed in TaxiVis

To configure how the placeholder fields are displayed on TaxiVis, we included a configuration file called *extra_fields.txt* (located in the data directory). This is a csv file whose contents have the following format:

```
field1 , Screen  Name1 , Axis  Label1 , DisplayValue
field2 , Screen  Name2 , Axis  Label2 , DisplayValue
field3 , Screen  Name3 , Axis  Label3 , DisplayValue
field4 , Screen  Name4 , Axis  Label4 , DisplayValue
```

The first field represents the internal name used for the attribute in the code, while the second represents the label shown in the interface. The third field contains the text used to label the axis in the plots that used this field. The last field is a flag that indicates if this field should be displayed in the interface or not. It can have two values: 1 (active), and 0 (inactive).



Start Time: Sun 05/01/11 00:00
Step Size: 1 hour
End Time: Fri 05/01/15 01:00

Attribute: Screen Name1
Num Bins: 10

Figure 2: TaxiVis showing the new data file.