Direct layer Estimation

Flow : 10007 -> 10004 -> 10002 | Tagret Link: 10004-10002

Aim: To find the TTStatistics for the Target Link

Input: TTstatistics_Veh2Veh of 10004 -> 10002

```
Step 1: Drop the last 3 columns of the dataframe, add headers, a column for BMS id and form
at the Date to (dd/mm/yyyy).
Step 2: Get the data of only the working day, ( DayType = 1).
Step 3: Trim the time between 7-9 am ( Morning Peak ).
Step 4: Calculate the standard deviation of each time interval for different working days.
```

Output: TT statistics of the target link for working days during the Morning peak hours.

Key variables:

```
Path1 : Contains the file location of the Target Link TTstatistics_Veh2Veh.
Tl_df_DE : Targest Link data frame for direct layer estimation.
ID : BMS ID of the sensnors.
Day_filter_DE : Filters the dataframe into a dataframe containg only working days ( DayType
1).
Time_filter1_DE,Time_filter2_DE : Trim the time between 7-9 am.
SD_df : Contains the dat after caculating standard deviation of each time interval for diff
erent working days.
```

In [1]:

```
# Add libraries
import csv
import os
import glob
import re
from pandas import read_csv
import pandas as pd
import numpy as np
```

Step 1: Drop the last 3 columns of the dataframe, add headers, a column for BMS id and format the Date to (dd/mm/yyyy).

In [2]:

```
# Path of all the TTstatistics_Veh2Veh datasets of link 10004 -> 10002
path1=r'C:\Users\shesa\Dropbox\Saagar\Sara\Code\DataSet\Input\Direct Layer
Estimation\All\2019_*.csv'
# Count for naming output files
count=0
for filename in glob.glob(path1):
    # Read the TTstatistics_Veh2Veh dataset
    Temp_df= pd.read_csv(filename)
    # Drop the last 3 columns
    Temp_df.drop(Temp_df.columns[len(Temp_df.columns)-1], axis=1, inplace=True)
    Temp_df.drop(Temp_df.columns[len(Temp_df.columns)-1], axis=1, inplace=True)
    Temp_df.drop(Temp_df.columns[len(Temp_df.columns)-1], axis=1, inplace=True)
    # Assign headers to the dataset
    Temp_df.columns = ['year','month (in digit)','DAY','DayType','Bin (min)','Shift (min)','Time (h
r)','SampleSize','MeanTT(sec)','Median','Mode','Max','Min','Variance','StandardDev of TT
(v2v)','LowerQUartile','UpperQuartile','95th_Percentile of TT','Buffer time','BufferTimeIndex']
    # BMS ID
    id=1000410002
```

```
    # Insert the BMS ID column to the dataset
    Temp_df.insert(20,"Btlinkid", id)
    # Format the date columns to a single column of format (dd/mm/yyyy)
    cols = ['year','month (in digit)','DAY']
    newcol = ['/'.join(i) for i in Temp_df[cols].astype(str).values]
    Temp_df = Temp_df.assign(Date=newcol).drop(cols, 1)
    # Save the modelled data for combination
    Temp_df.to_csv(r'C:\Users\shesa\Dropbox\Saagar\Sara\Code\DataSet\Input\Direct Layer
Estimation\Combined\C%s.csv' %count,index=False)
    # Increament file naming variable
    count=count+1

# Add the working directory that contains the csv files from the prevuious step to combine into on
e csv.
os.chdir(r'C:\Users\shesa\Dropbox\Saagar\Sara\Code\DataSet\Input\Direct Layer Estimation\Combined'
)
# Searching for all file names in the directory that has "csv" as its extention
extension = 'csv'
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
# Combine all files in to a list
combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames ])
# Export the list to csv
combined_csv.to_csv(r"C:\Users\shesa\Dropbox\Saagar\Sara\Code\DataSet\Input\Direct Layer
Estimation\Input\2019_1000410002.csv", index=False, encoding='utf-8-sig')
```

Step 2: Get the data of only the working day, (DayType = 1).

In [3]:

```
# Path of the resultant dataset after step 1
path1=r'C:\Users\shesa\Dropbox\Saagar\Sara\Code\DataSet\Input\Direct Layer
Estimation\Input\2019_1000410002.csv'
# Read the combined dataset
tl_df_de= pd.read_csv(path1)
# Filter the dataset only for working days of the week ( DayType 1)
Day_filter_de = tl_df_de[tl_df_de['DayType']==1]
```

Step 3: Trim the time between 7-9 am (Morning Peak).

In [4]:

```
# Filter the resultant dataset further into morning peakhours TTstatistics_Veh2Veh dataset
Time_filter1_de = Day_filter_de[Day_filter_de['Time (hr)']>=7]
Time_filter2_de = Time_filter1_de[Time_filter1_de['Time (hr)']<=9]
```

Step 4: Calculate the standard deviation of each time interval for different working days.

In [5]:

```
# Add a new colum to the dataset for Square of V2V standard deviation
Time_filter2_de['Sq.of StandardDev of TT (v2v)']=Time_filter2_de['StandardDev of TT (v2v)']**2
# Square root(sum(square(V2V standard deviation))) at diffeent time intervals
y=np.sqrt((Time_filter2_de.groupby('Time (hr)')['Sq.of StandardDev of TT (v2v)'].sum())) # Square
root(sum(square(V2V standard deviation))) at diffeent time intervals
# Get time unique time intervals from V2V standard deviation (5 min intervals)
i=Time_filter2_de['Time (hr)'].unique()
# Convert to a list
i.tolist()
# name index of variable y to the value in variable i
y.index=i
# Conver the variable to a dictinoary containg a key value pair {'key':'value'}, In this case {'Ti
me':'Standarddev of link'}
#eg {'7.0000':'137.1937615','7.016666667':'130.5633541',..... }
y=y.to_dict()
# Map the key value pair to the time intervals in the TTstatistics_Veh2Veh dataset
Time_filter2_de['StandardDev of Link at Time(hr)'] = Time_filter2_de['Time (hr)']
Time_filter2_de['StandardDev of Link at Time(hr)']=Time_filter2_de['StandardDev of Link at
Time(hr)'].map(y)
# Filter needed columns to a new dataframe and save the result
SD_df = Time_filter2_de.filter(['Btlinkid','Date','Time
(hr)','MeanTT(sec)','SampleSize','StandardDev of Link at Time(hr)'], axis=1)
```

```
SD_df.to_csv(r'C:\Users\shesa\Dropbox\Saagar\Sara\Code\DataSet\Input\Direct Layer Estimation\Output\2019_sep_10004_to_10002_TTStatistics_Veh2Veh_5min_(Direct_Estimation).csv',index=False)
```

```
C:\Users\shesa\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\shesa\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  import sys
C:\Users\shesa\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```