FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2016.... Open
Author: Ine Foss	…… (signature author)
Instructor: Tomasz Wiktorski Supervisor(s):	
Title of Master's Thesis: Data quality and transit delay analysis for historical AVL data based on the SIRI protocol	
ECTS:	
Subject headings: Data Analysis	Pages: 46…… + attachments/other: …… Stavanger, 15/06/16…… Date/year

Abstract

Traffic agencies are investing in advanced technologies to improve their services. An automatic vehicle monitoring (AVM) system with vehicle location provides vehicles with a device to record information. In a bus transit system, vehicles transmit information about the trip to an external server each time it arrives at a destined terminal. The instant benefit is to create a real time overview of the system, giving the agency the ability to monitor the system and locate delays and problems faster. However, storing this information creates a valuable dataset of transit information. The collected data may be explored, visualized and analyzed to find patterns, tendencies and other useful information which can contribute to make improvements on the bus system.

In this thesis, historical data from four routes located in Stavanger was used as a case study. The data was collected from an automatic vehicle location (AVL) system using the Service Interface for Real Time Information (SIRI) standard. The thesis focuses on the development of trip delay on the bus stop level, delta delay for locating bottlenecks as well as the quality of the data analyzed.

Acknowledgement

I would like to express my appreciation to my adviser Associate Professor Tomasz Wiktorski of the Department of Electrical and Computer Engineering at the University of Stavanger for providing guidance and support throughout the work on my master thesis.

Table of Contents

List of Figures

List of Tables

1 *Introduction*

1.1 Background

1.1.1 The society's need for public transport

Public transport is a collection of different types of public available vehicles transporting people within and between urban and suburban areas. In this thesis, the focus will be on the bus transit systems centered in and around Stavanger.

Improving the public transport systems gives benefits to both the society and the environment. The road network cannot efficiently handle the increasing number of vehicles, which during the rush hours results in traffic congestion and traffic jams. The size and capacity of a bus is contributing to reduce the heavy load on the road network. Slow moving traffic causes more pollution than normal traffic and is significantly reducing the quality of the air and surroundings.

The size and capacity of a bus is contributing to reduce the heavy load on the road network. While the average size of a bus is about two or three times the size of an average private car, the capacity can hold up to twenty times more passengers assuming both capacities are maximized. However, the average car is rarely holding more than two passengers [1]. Hypothetically, one full bus could replace fifty cars in a queue. While the busses contribute to fewer vehicles on the road, they also reduce the total level of pollution. A

bus uses more fuel than an average private vehicle, but with the capacity of a bus in mind, the pollution per passenger is lower.

Since most private cars are used as a means of transport, they will need to occupy a parking space when not in use. The cities often have a limited number of parking spaces available, and in the rush hours lack of parking may cause a traffic jam, extra pollution and even illegal parking. Parking spaces also takes up a lot of space in the cities.

In addition to the environmental and economical benefits of public transport, there also give social advantages. Many passengers would be unable to travel without the use of public transport. In all societies there are several groups of people without the opportunity to drive. Children, teenagers, elderly, sick and poor people may have public transport as their only means of transport past walking and cycling distances.

However, there are some challenges related to the public transport systems. One of the main problems is when the bus is late on schedule. Delay caused by the traffic is a problem the transport agencies cannot control, only make account for.

1.1.2 Improvements

There are some challenges present when talking about the public transport systems. One of the main problems is the bus not being on time. While it sometimes is ahead of schedule, the largest problem is when it is delayed. Delay caused by the traffic is a problem the traffic agencies cannot control, only make account for. However, there are some improvements made to the system to make it more reliable.

Bus Rapid Transit(BRT) is a concept where the bus is operating in a separate line to avoid traffic. This results in more reliable predictions on the transit time, which will affect the accuracy of the route. "Busway2020" [2] is a BRT project in Rogaland where they will be introducing a busway connecting Tananger, Stavanger and Sandnes. In Hillevåg and Mariero a busway with bus prioritized roundabouts are already build. There the bus is operating in a seperate lane, and when approaching a roundabout, all other traffic stops and the bus is able to drive through it.

The time spent at a bus stop is also affecting the trip time. To avoid unnecessary wait time at the bus stop, buss cards and mobile applications can be used to pay for the ticket. When buying a ticket aboard the bus is more expensive than using a prepaid ticket system, fewer people will pay aboard the bus.

Another improvement which also benefits the passengers are the scheduling applications. Mobile applications and bus stop screens are making it possible for passengers to rapidly changing their planned way of travelling based on the scheduled bus times and the monitored bus times. Sometimes there are several bus lines travelling in the same direction, and giving the passengers the ability to see if the bus is on time will help them chose the fastest route.

1.2 Collection of data

1.2.1 Data Systems

Transit agencies are turning to advanced systems to improve their services. Automatic vehicle monitoring(AVM) is a system for monitoring scheduled and actual transit data. One of the key technologies in this system is the automated vehicle location (AVL) system, which is a computer based real-time tracking system.

The Service Interface for Real Time Information (SIRI) is a CEN standard for exchanging real-time transit data between different computer systems. Devices located in the vehicles transmits information about the trip to an external server each time it arrives at a destined terminal. For excanging informaiton from the vehicle (client) and the server, SIRI uses general communication protocols such as Request/Response and Publish/Subscribe.

The Vehicle Monitoring Service (VM) [3] is one of the main sevices provided by SIRI, which provides information about the location and expected activities for a particular vehicle. This information includes the previous bus stop and a ordered list of the next bus stops in the trip along with the scheduled arrival. There are usually a screen in the vehicle showing this information to the passengers.

SIRI also provides Stop Timetable (ST) and Stop Monitoring (SM). These services includes information about the current and expected vehicles to arrive at a given bus stop. At the bus stations and on some bus stops, a monitor is typically displaying this information to the passengers.

The instantly benefit of SIRI is to create a real time overview of the system, giving the agency the ability to monitor the system and locate delay and problems faster as well as providing real time information to the passengers. However, the collection of these data creates datasets with valuable information. The transit information datasets are growing, day by day. Working

with Big Data may cause some increased running time, but analyzing large datasets gives more accurate patterns and tendencies.

1.3 Previous work

In the field of public transport analysis, methods for analyzing transit data to find patterns for making the schedule and route planning process more accurate have been around for decades. In the recent years, the importance for information retrieval and data collection have been amplified, along with the need for better analyzing methodes.

In 1964, the Federal Transit Administration was formed with the purpose of providing information, support and development assist to improve the public transport systems in the United States. One of their projects, the Transit Cooperative Research Program (TCRP) [4] have regularly performed and posted free research and publications regarding various aspects about public transport. Their report "Data Analysis for Bus Planning and Monitoring" [5] from 2000 describes analyzing methods for small datasets and samples including trip time analysis, running time, route level schedule adherence and headway regularity for APC and AVL systems. TCRP later posted a paper called "Using Archived AVL-APC Data to Improve Transit Performance and Management" [6] where they in addition to the methods above presented tools for analyzing running times and designing scheduled running times.

The tools created were made as extensions to the software TriTAPT [7] created by the Delft University of Technology. The original software uses route description, scheduled arrival and departure times for all stops and all trips in addition to stop time and delay to create aggregated and trip-by-trip-graphs.

1.4 Objectives

The first objective in this thesis will be to evaluate the quality of the data used for the case study. The quality will be evaluated by looking at how well the trips are monitored. Well monitored trips are important for further analysis.

The second objective is to observe and analyse the development of the delay for all the routes. By looking at the delay on a bus stop level, we can look for tendencies and irregularities and compare the routes on trip delay. Delta delay per bus stop will also be analysed.

The third objective is to explore the rush hours. The rush hour will be

defined from the route delay. The mean delta delay will be calculated per bus stop to look for bottlenecks in the system. We will also take a look at a bus stop during the rush hours to see how the mean delay affects the times.

1.5 Dissertation Organization

This thesis consists of four chapters. The first chapter has introduced the background, previous work and objectives for the research. Chapter Two includes an overview of the dataset used and describes the quality of the data. In Chapter Three, trip delay and trip delta delay on bus stop levels are analyzed. The peak hours of the delta delay have been found and used to find bottlenecks in the system. Chapter Four gives a conclusion on the data quality and the trip time analysis, and discusses ideas for future work.

2
Data Quality

In this chapter, the quality of the dataset will be analysed and vizualized. Each bus is equipped with a live monitoring system to record data about the trips. In a perfect system, there would be complete data from each trip, which would be a great starting point for data analysis. Unfortunately, the live monitoring system could fail or the bus driver could turn the monitoring off manually, which makes the data incomplete. Data from trips without live monitoring lacks the actual timestamps, but still includes information about the trip and the expected schedule, which makes is possible to find out the proportion of monitored trips.

2.1 Overview of the Dataset

The dataset used in this thesis was provided by Kolumbus. It contains transit information from busses in Rogaland and it was created by logging trip information at each bus stop using the Siri protocol. The dataset consists of more than 5 000 000 lines divided into 54 .txt files, 1.29 GB of data, collected between August and December 2015. The files are sorted on dates where each file includes all recorded transit information for the given day.

2.1.1 Attributes

The dataset consists of 21 attributes;

Table 1: Overview of the attributes of the dataset

Attribute	Description
AreaExternalCode	ID of the transport type
AreaDescription	Name of the transport type
LineCode	ID of the bus line
LineDesc	Name of the bus line
VehicleShift	Shift of the vehicle
TripID	ID of the trip
VehicleCode	ID of the vehicle used
VehicleDesc	Type of vehicle
NodeCode	Bus stop ID
NodeDesc	Bus stop name
NodeType	Event type
ActualArrival	The actual time of arrival
ScheduledArrival	The scheduled time of arrival
stopTime	The time the bus spent on the bus stop
delayAdvance	The time between scheduled and actual time of arrival including the deviation
actualHeadway	Actual time since last bus at the bus stop
scheduledHeadway	Scheduled time since last bus at the bus stop
scheduledTimeBtwStops	Scheduled time between bus stops
actualTimeBtwStops	Actual time between bus stops
deviation	The deviation between scheduled and actual time between stops

Table 1 shows an overview of the attributes included in the dataset. The description is written by the author of this thesis based on her understanding of the different fields. Table 2 shows an example of how the dataset is structured. While there are a lot of available information stored in each record, not all are relevant to answer the research questions. In the pre-processing subchapter, only the relevant attributes will be retrieved, while a few extra will be calculated from other attributes. The relevant list of attributes will follow in the next subchapter.

Table 2: Example of record from the dataset

Attribute	Example
AreaExternalCode	1
AreaDescription	Boreal Transport Sør
LineCode	1033
LineDesc	X60
VehicleShift	150
TripID	10332029
VehicleCode	233
VehicleDesc	BUSS
NodeCode	11036092
NodeDesc	UiS øst (11031532)
NodeType	Bus stop
ActualArrival	2015-05-05 15:26:30
ScheduledArrival	2015-05-05 15:18:00
stopTime	00:00:45
delayAdvance	-00:07:20
actualHeadway	00:17:43
scheduledHeadway	00:15:00
scheduledTimeBtwStops	00:01:00
actualTimeBtwStops	00:02:10
deviation	-00:01:10

2.1.2 Trips

One of the key attributes is the trip ID. All trips include an ID which can be used to find out which bus line it belongs to, where the start and stop terminals are and what time of day it is active. Trip IDs are daily unique, but for the weekdays or weekends, a trip ID is repeated daily. In other words, trips departing at the same time each day have the same trip ID.

For a bus line, not all trips have the same route. Looking at the ID itself, one can only see what bus line it belongs to, not which of the different types of routes it is from. A bus line have typically two different trips in their route; trips travelling from one part of the city to another, and the returning trips.

2.1.3 Delay and Delta Delay

The dataset contains a column named delayAdvance. It is defined as the time between the scheduled and the actual time of arrival plus the deviation between the scheduled and actual time between stops. While this field may be useful for some statistical analysis itself, it will not be included in this thesis. However, the delay itself is one of the key attributes used in this thesis, and will be in the center of attention for the analysis. The delay is calculated based on the actual and scheduled arrival.

$$Actualarrival - scheduledarrival = delay \tag{1}$$

While the delay is an useful way to look at the load on the roadwork, the delta delay will help us find out how much the delay has increased or decreased between the bus stops. Regardless of the bus being on schedule or delayed, this will tell us more about the time it takes the bus to move from one bus stop to another. The delta delay will be measured in minutes, and the value is calculated by the delay of the given row minus the delay of the previous row.

$$delay_r - delay_{r-1} = \Delta delay \tag{2}$$

2.1.4 The Bus Lines Used

In order to reduce the amount of data, the following bus lines will be used in this case study:

- X60: Hundvåg - Stavanger - Sandnes
- X76: Randaberg - Kvadrat
- 6: Stavanger - Grannes
- 7: Stavanger - Sola

These four lines are all passing the University of Stavanger, and this thesis will therefore focus on the traffic situation for the buses passing this specific area.

Figure 1 shows an overview map of the bus lines

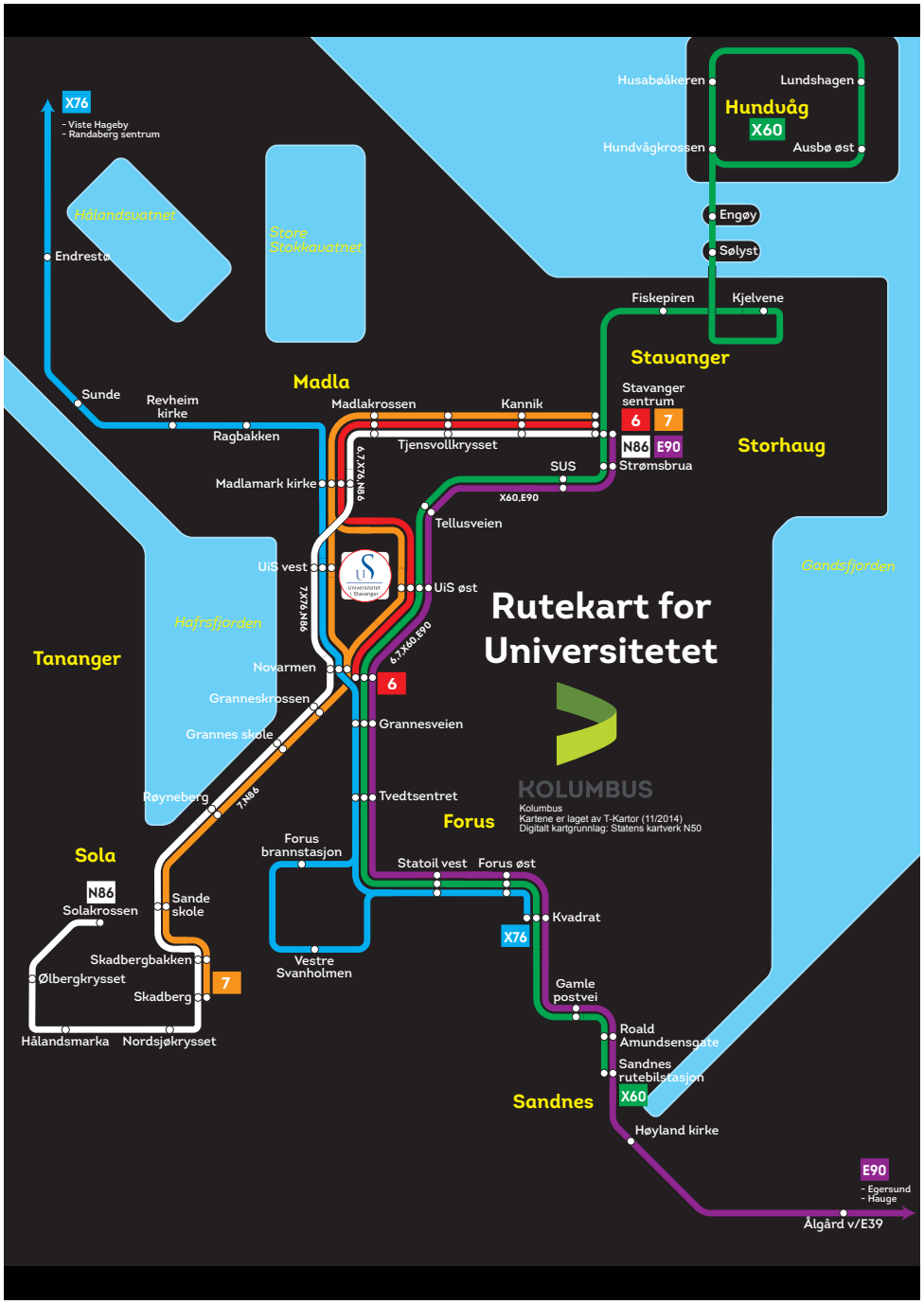


Figure 1: A map over the routes for the bus lines used

2.2 Pre-prosessing

The pre-processing of the dataset are done in the `import_data` notebook. This notebook imports the dataset from txt-files, pre-processes the data, extracts the relevant parts and saves it to a Pandas-file. This way, this time consuming process only have be ran once.

In Listing 1, the DataFrame `all_data` is created from the .txt files located in a subfolder named 'data'. We need to use both comma and semicolon as separators since both of them are present in the files. The DataFrame created contains the raw dataset where missing values are replaced with 'na'.

```
allcsvs = [f for f in listdir('./data')]
csvlist = [pd.read_csv('./data/'+fcsv, sep= ',|;', engine='python',
    header=0, encoding='utf-8')
          for fcsv in allcsvs]
all_data = pd.concat(csvlist).fillna(0)
```

Listing 1: Create a DataFrame from all files in the subfolder 'data'

The pre-processing process is shown in Listing 2 and is one of the most time consuming tasks in this project. Unprintable characters in the dataset are removed by rebuilding it character by character of printable characters. Unfortunately, "ÆØÅ" are not considered printable characters in pandas. Since this doesn't have an impact on the programming, the extra run time it would take to remove unprintable characters without the build in printable function does not make up for the extra characters in the description attributes.

After removing unprintable characters, the values in the time fields are converted to the timedelta format. Timedelta is a subclass of datetime representing a duration expressing the difference between two dates.

Looking at the column `NodeDesc` it is clear that it needs some extra processing. The values in this attribute should only include the name of the bus stop. However, in addition to the name it includes a lot of white spaces and the bus stop ID. Since the bus stop ID is present in another column, it doesn't have to be in this. By stripping the `NodeDesc` column for white spaces and removing the bus stop ID's, the values become shorter and more manageable.

Some of the rows have letters as part of the trip ID. When a bus is driving from the bus station to the bus stop where the trip starts, two letters are added to the trip ID. This results in several similar rows in the dataset, only seperated by a differend node description. They will not be needed in the

further analysis, and are removed by only including records where the trip ID is a digit.

```
all_data_p = all_data.applymap(lambda x: ''.join(char for char in x
    if char in string.printable))
delta_fields = ['stopTime', 'delayAdvance', 'actualHeadway', '
    scheduledHeadway', 'scheduledTimeBtwStops', 'actualTimeBtwStops'
    , 'deviation']
all_data_p[delta_fields] = all_data_p[delta_fields].applymap(
    timedelta_convert)
datetime_fields = ['ActualArrival','ScheduledArrival']
all_data_p[datetime_fields] = all_data_p[datetime_fields].applymap(
    lambda x: dt.datetime.strptime(x, "%Y-%m-%d %H:%M:%S") if len(x)
    >18 else np.datetime64('nat'))
all_data_p['NodeDesc'] = all_data['NodeDesc'].str.replace(r"\(.\)"
    ,"").str.strip()
```

Listing 2: Pre-processing by removing nonprintable characters and convert the time to timedelta

In addition to the existing fields, a field calculating the actual delay is added by subtracting the scheduled arrival from the actual arrival. This is shown in Listing 3. Some of the fields are also renamed before saving the DataFrame to a HDFS store.

```
all_data_p['La'] = all_data_p['ActualArrival'] - \
                   all_data_p['ScheduledArrival']

all_data_p.rename(columns={'ScheduledArrival': 'As',
                           'TripID': 'IDt',
                           'NodeCode': 'Sid',
                           'NodeDesc': 'Sn',
                           'NodeType': 'St',
                           'LineDesc': 'Bn',
                           'LineCode': 'Bid',
                           'VehicleCode': 'Vid'}, inplace=True)

all_data_p = all_data_p.drop('index', 1)
```

Listing 3: Renaming columns in the preprocessed dataset

This store contains the full dataset fully pre-processed, but since we will focus on only four of the bus lines in this thesis and not the whole dataset, another store is made containing only the relevant part of the dataset. Before storing, two functions are applied and two extra columns are made.

When an extra bus have been deployed into the system, there are two trips with the same trip ID at the same date. The only way to distinguish

them is the vehicle code, but in a multi-indexed DataFrame based on trip date and the trip ID, the trip will look duplicated with two inputs per bus stop. When they are present in the dataset, the integrated statistical functions of Pandas will give a unreliable result, and since the amount of extra busses are to low to do a proper analysis on in this setting, and the integrated pandas functions are valuable, they will be removed from the dataset. The function `add_column_dup` adds a new column to the DataFrame where trips including two vehicle codes at the same day are set to `True`. The dataset only includes rows where the `dup` column are set to `False`.

```
def add_column_dup(x):
    try:
        if len(x.VehicleCode.unique()) > 1:
            x['dup'] =  True
        else:
            x['dup'] = False
    except:
        x['dup'] = False
    return x

relevant_data['Dt'] = relevant_data.As.apply(add_column_day)
```

Listing 4: Adding a column with a more descriptive date

A dilemma with using the date to separate the different trips are the late night trips. Trips in transit at and after midnight will be separated by the date creating overlaps or the illusion of half monitored trips. Instead of checking the hour of the day on each trip at all executions, a new column is added. The function `add_column_date_trip` creates a new column based on the date, where after midnight trips gets the date from before midnight.

```
def add_column_date_trip(x):

    if  (x.hour < 5):
        x = x.date() - dt.timedelta(days = 1)
    else:
        x = x.date()

    return x

group = relevant_data.groupby([relevant_data.As.dt.date,
    relevant_data.IDt]).apply(add_column_dup)
relevant_data = group[group.dup == False]
relevant_data = relevant_data.drop('dup', 1)
```

Listing 5: Removing duplicates

Another column is also added. The `La_d` is the delta delay which is calculated by the delay of the row minus the delay of the previous row. By finding the delta delay between two bus stops, we will be able to find out how much the delay has increased or decreased between the stops. The `Departure terminal` bus stops, which is the start node of each trip, are set to 0.

```
relevant_data['La_d'] = relevant_data.La - relevant_data.La.shift
    (1)
relevant_data['La_d'] = relevant_data.La_d.apply(lambda x: x /np.
    timedelta64(1, 's')) / 60
q0.loc[q0.St == 'Departure terminal', 'La_d'] = 0
q0 = q0.dropna()
```

Listing 6: Adding a column with the delta delay

The renamed fields now becomes:

Table 3: Renamed attributes

Old Attribute	New Attribute	Description
LineCode	Bid	ID of the bus line
LineDesc	Bn	Name of the bus line
TripID	IDt	ID of the trip
NodeCode	Sid	Bus stop ID
NodeDesc	Sn	Bus stop name
NodeType	St	Event type
ScheduledArrival	As	The scheduled time of arrival
	La	Actual delay
	La_d	Delta delay
	Dt	Main trip date

2.3 Monitoring

Since the data with live monitoring is the relevant part of the dataset, looking at how much of the dataset is useful is a good way to start. From the preprocessed data, there are created four new DataFrames for each of the four bus lanes. The new DataFrames includes all records from that bus lane where the node is considered a bus stop. Listing 7 shows how all the relevant data for bus lanes are extracted from the preprocessed data `q0`.

2.3.1 Monitored Trips

```
q0 = bd[['As', 'IDt', 'Sid', 'La', 'Sn', 'Bn', 'NodeType', 'Bid']]

q0_x60 = q0[q0.Bid.isin({'1033'}) & q0.NodeType.isin({'Bus stop'})]

q0_x76 = q0[q0.Bid.isin({'1039'}) & q0.NodeType.isin({'Bus stop'})]

q0_6 = q0[q0.Bid.isin({'1007'}) & q0.NodeType.isin({'Bus stop'})]

q0_7 = q0[q0.Bid.isin({'1008'}) & q0.NodeType.isin({'Bus stop'})]
```

Listing 7: Creating a new dataset containing data from x60

Some of the bus lines have departures more frequently than others and the trips have a different number of bus stops, making the total number of both trips and records different. This is something that have to be considered when making conclusions based on the findings later in this thesis. Table 4 gives an overview of the size of each of the bus lines subset.

Table 4: Overview of the size of each subset

Monitored trips	X60	X76	6	7
Total number of records	153 094	26 434	44 822	152 056
Total number of trips	2 986	468	2 181	3 975

Records including the delay are considered live. However, even if a trip contains live data, there are no guarantee that all records from that trip does. The `make_live_trip_count()` method is called on each of the bus line subsets, in this function called `bd`, to make a new DataFrame showing how many of the records from each trip contains live data.

```
def make_live_trip_count(bd):
    tot = bd.groupby([bd.As.dt.date, 'IDt'])['La'].size().
    reset_index()
    tot.columns = ['As', 'IDt', 'tot']

    live = bd.groupby([bd.As.dt.date, 'IDt'])['La'].count().
    reset_index()
    live.columns = ['As', 'IDt', 'live']

    result = pd.merge(tot, live, on=['As', 'IDt'])
```

```
9     result['missing'] = result.tot - result.live
10
11    return result
```

Listing 8: Creating the DataFrame with counts for each trip

Since the trip IDs only are unique for each day, the data is grouped by both date and trip ID. The columns contains a count of how many records each trip each day contains and how many of them are live and missing. The `tot` column contains how many records there are for that trip, and the `live` column contains how many of the records includes live data. The `missing` column contains a count on how many of the records which are missing the delay. When all the records for a trip includes the delay, the value in the `missing` column would be 0. Likewise if none of the records from a trip contains live data, the value in the `missing` column would be equal to the total number of records.

	As	IDt	tot	live	missing
0	2015-08-03	10331000	43	39	4
1	2015-08-03	10331002	43	43	0
2	2015-08-03	10331004	43	40	3
3	2015-08-03	10331006	43	40	3
4	2015-08-03	10331008	43	43	0
5	2015-08-03	10331010	43	0	43
6	2015-08-03	10331012	43	43	0
7	2015-08-03	10331014	43	41	2
8	2015-08-03	10331016	43	39	4
9	2015-08-03	10331018	43	41	2

Figure 2: Preview of the trip-count DataFrame for x60

The method `calculate_trip_count()` takes in the DataFrame above and calculates how many percent of the trips which are live. The `live` and the `missing` columns are extracted as different lists. The lists are normalized and sorted by index.

```
1 missing_percentage_list = bd.missing.value_counts(normalize=True).
      sort_index() 100
2 live_percentage_list = bd.live.value_counts(normalize=True).
      sort_index() 100
```

```
live_all  = missing_percentage_list[0]
live_any  = live_percentage_list[0]
```

Listing 9: Calculating live trips

The index is the first element in the `missing_percentage_list`, `live_all`, is the percentage of trips where all the records includes live data. The first element in the `live_percentage_list`, `live_any`, is the percentage of trips where none of the records includes live data. This gives us enough data to create a table of the percentage of the monitoring types.

Table 5: The degree of each trip monitoring type

Monitored trips	X60	X76	6	7
Fully monitoring	77.33	51.07	84.14	82.54
Any monitoring	88.38	82.48	89.78	89.96
Partially monitoring	11.05	31.41	5.64	7.42
Without monitoring	11.62	17.52	10.22	10.04

At first glance, the data seems rather consistent. However, the data from X76 does stand out a bit with a lower number of full monitoring, and a higher number of partially and no monitoring. Plotting the data to bar plots gives an easier overview.

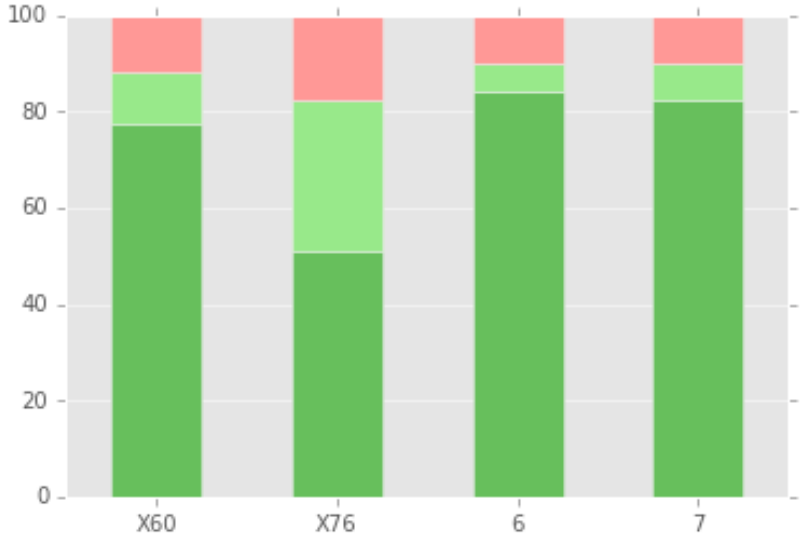
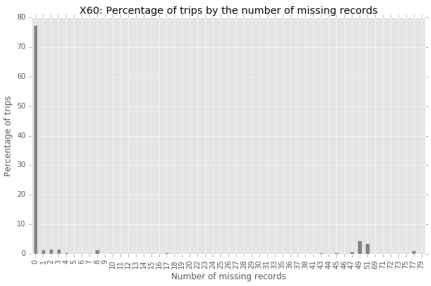


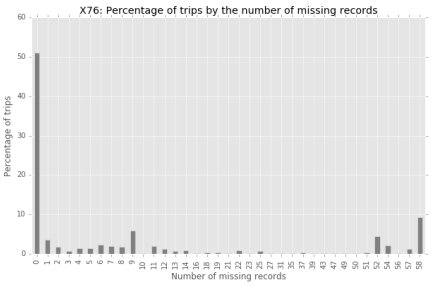
Figure 3: Monitored trips compared

In Figure 3, the data from Table 5 are presented with bar for each bus line. Dark green represent trips where all the records include live data. On these trips, the monitoring system was turned on and fully working and recorded the delay at each of the bus stops. The red part of the pie chart represent the trips where all records are without live data, which means that the monitoring system was turned off during the whole trip. The light green represent partially monitored trips where the monitoring system failed or got turned off during the trip. The two green parts combined represent all trips with live monitoring on at least one record. Figure 3 is a bar representation of Table 5 with the same distribution.

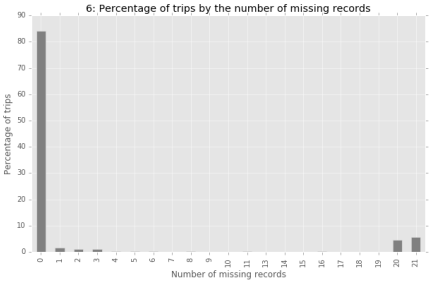
Looking at the comparison in Figure 3, it is clear that bus lines X60, 6 and 7 have a higher percentage of fully monitored trips than X76. However, looking at the percentage of trips with any monitoring, X76 is not far from the rest of the bus lines when including the partially monitored trips. The discrepancy may be due to the live monitoring system aboard the X76 bus lines were more frequently disrupted either by system failure or manually handling of the system. Looking at how many records are missing per trip for each of the bus lines gives a better view of the relevance of the partially monitored trips.



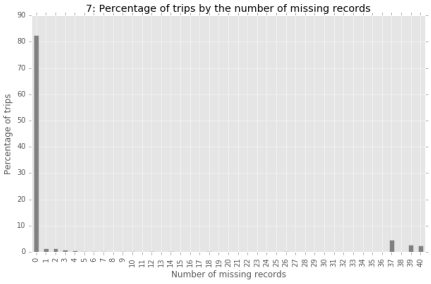
(a) X60-trips



(b) X76-trips



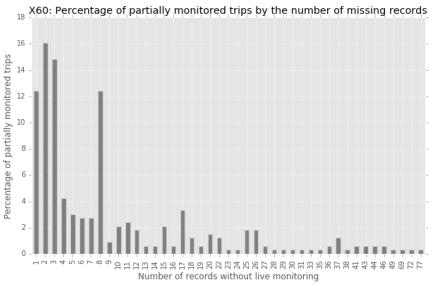
(c) 6-trips



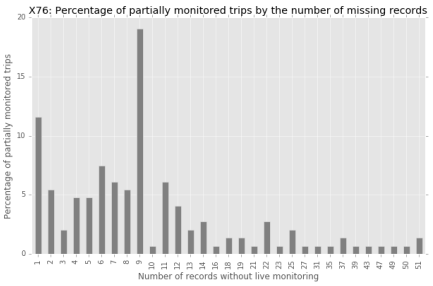
(d) 7-trips

Figure 4: Proportion of missing live data for monitored trips

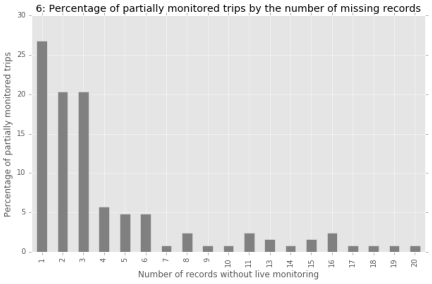
Figure 4 shows how many percent of the trips are missing a given number of live data. The x-axis is the number of records in each trip without live data. Looking at the graph, it's clear that most of the trips are fully monitored. It is also possible to see the trips without monitoring, but the rest of the bars are less easy to see. By removing the fully monitored trips and the trips without any monitoring, it gets easier to see how the partially monitored trips look like.



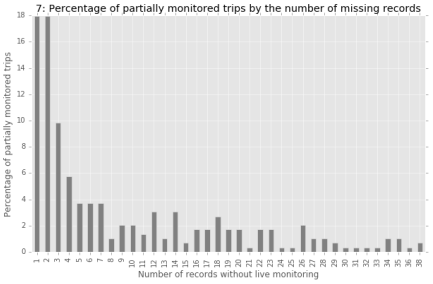
(a) X60-trips



(b) X76-trips



(c) 6-trips



(d) 7-trips

Figure 5: Proportion of missing live data for partially monitored trips

Figure 5 shows that the majority of the partially monitored trips are missing few records. This means that on the trips where not all records include live data, most of the records do.

2.3.2 Monitored Bus Stops

Now that we know the proportion of monitored trips, it's time to look at the monitoring at the bus stop level. In the previous subchapter, it is shown that many of the trips are partially monitored, and in this subchapter we will look at the bus stops to see where in the trip there are missing live data.

The same DataFrames shown in Listing 7 will be used except for the subset for bus line number 7. Some of the bus lines have different versions of the trips during a day, where the bus drives two different versions of the trip. There are two different versions of trips on bus line number 7, a day trip where the the bus drives through the University of Stavanger and a night trip where is drives around it. There are only a few bus stop IDs separating the two versions, so they can't be separated by the departure and arriving bus stop types. Since all trips for a given day have unique trip IDs and no

distinguishing on the different versions of the trips other than the different bus stop IDs included, making two different DataFrames for bus line number 7 based on the inclusion of the different bus stops gives the least time consuming solution. In Listing 10, two lists of trip IDs are created based on the inclusion of the IDs of the bus stops which is used to extract the relevant data to two new DataFrames.

```
q0_7 = q0[q0.Bid.isin({'1008'}) & q0.NodeType.isin({'Bus stop'})]

q0_7_trips_d = q0_7[q0_7.Sid.isin({'11031638'} | {'11031637'})]['
    IDt']].squeeze().unique()
q0_7_trips_n = q0_7[q0_7.Sid.isin({'11031539'} | {'11031156'})]['
    IDt']].squeeze().unique()

q0_7_d = q0_7[q0_7.IDt.isin(q0_7_trips_d)]

q0_7_n = q0_7[q0_7.IDt.isin(q0_7_trips_n)]
```

Listing 10: Creating two subset for bus number 7 based on version of trip

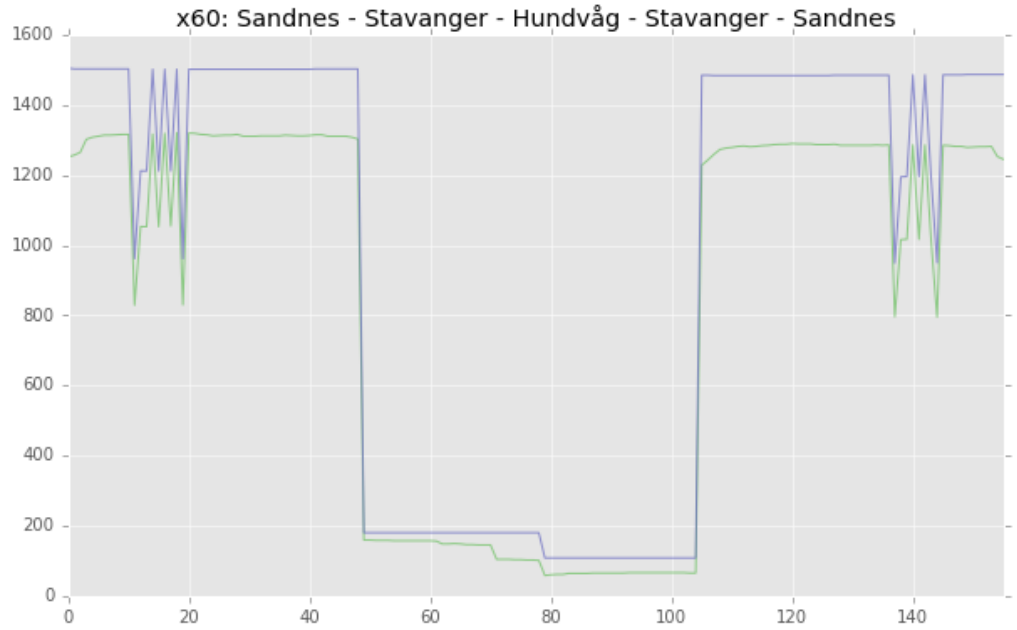
To create a graph from each of the subsets, we need to find the bus stop IDs for all trips. Since a trip is a ordered composition of bus stops, sorting by bus stop ID is a bad idea to create a informative graph later. The easiest way to extract the different bus stop ID's would be to get the unique values from the bus stop ID column in each bus line's subset. This would result in a list of stops sorted by the order of appearance in the subset. However, if roadwork have caused bus stops to physically move, the temporary bus stops are given a new ID and will be added last to the bus stop ID list. To avoid this, the `get_route()` function returns a list where temporary stops are merged in. The function finds all the different trips for a given bus line, sorts them by number of trips, concats them and lists the bus stops in order of appearance. The idea is that the trip with the most bus stops includes temporary stops that are only present on some trips.

With the list of the bus stops in the right order, the function `make_stop_count()` creates a new DataFrame where each bus stop is listed with the number of live records and the total number of records connected to the given bus stop.

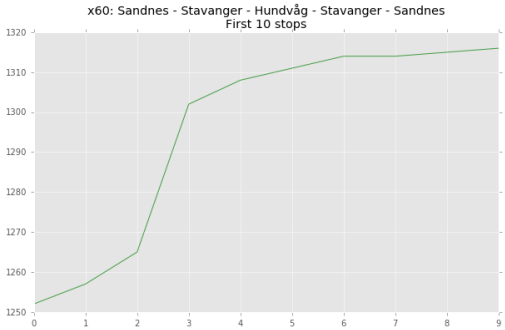
	Sid	live	tot
0	11035806	980	1177
1	11031234	994	1177
2	11031232	1013	1177
3	11031227	1031	1177
4	11031226	1037	1177
5	11031224	1039	1177
6	11031216	1043	1177
7	11031218	1043	1177
8	11031472	1044	1177
9	11031183	1045	1177

Figure 6: Preview of the first ten stops for bus number 6 and the number of live and total records

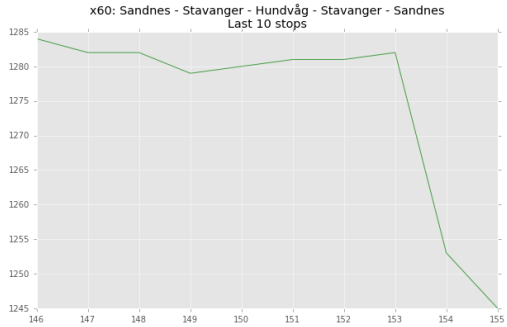
Turning this stop-count DataFrames into graphs results in Figure 7 to Figure 11. The x-axis are the bus stops and the y-axis are the number of records. The blue line is the number of maximal records on the particular bus stop and the red line is the number of records with live data.



(a) Overview



(b) First 10 stops



(c) Last 10 stops

Figure 7: X60: Bus stop monitoring

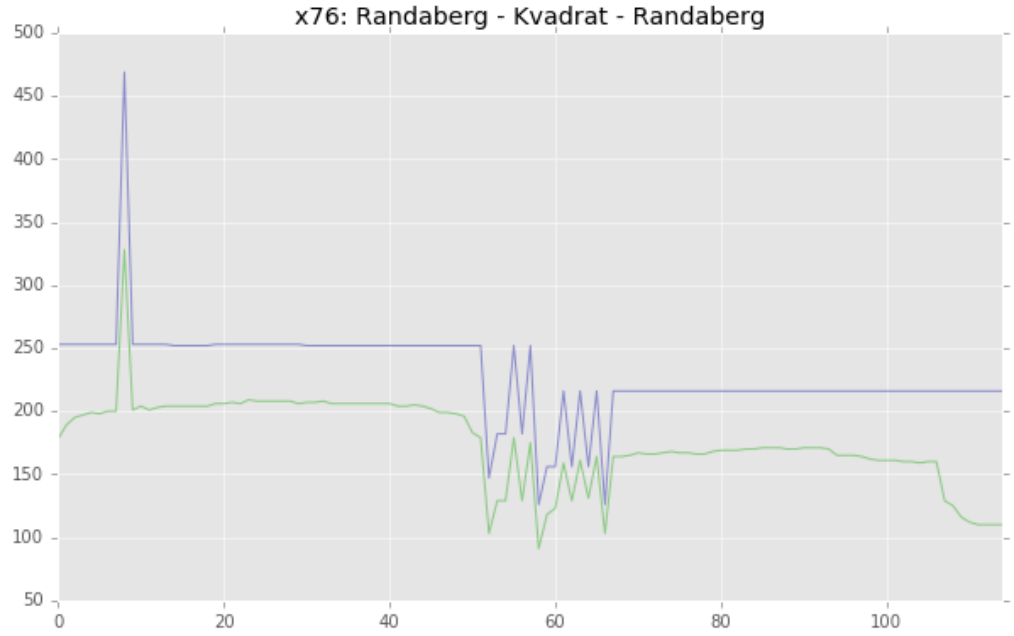
The first 48 bus stops are from the trips from Sandnes to Stavanger. Due to roadwork on Forus, some of the trips have a few temporary stops extra. This affects the graph where you can see them with both less live and total number of records. However, the difference between the lines remains more or less the same.

The bus stops from 49 to 78 are from the trips from Stavanger to Hundvåg. These trips are only available as an extended trip from Sandnes during the

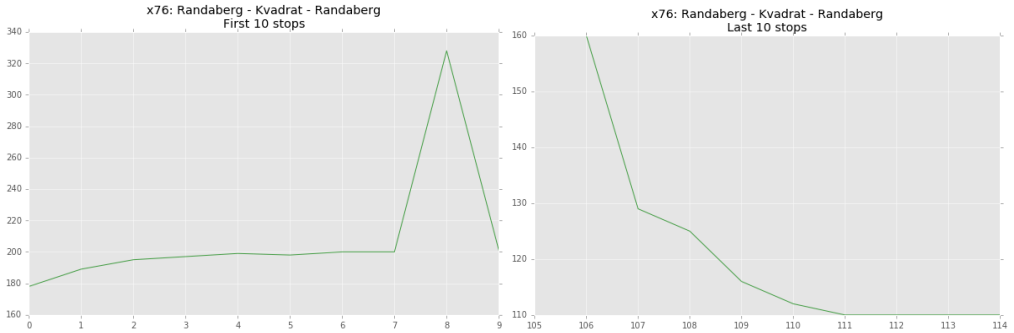
rush hour at afternoon.

The next 25 stops on the graph are from the trips from Hundvåg to Stavanger. These trips are only available during the rush hours on the mornings, and have fewer departures than the afternoon trips. Here, the number of records with live data is consistent during the whole route. The last 50 bus stops are from the trips from Stavanger to Sandnes.

From the research questions, one of the hypothesis was that the live monitoring had a tendency to be turned on after the trip was started. We can see that the number of records with live data increases after the first few stops of each of the trips. However, it is also visible on the graph that there are no increase in the number of records with live data when the bus is driving from Stavanger to Hundvåg. This may be due to the fact that there are only continuing trips from Sandnes to Hundvåg with a pit stop in Stavanger. This suggests that the monitoring system are not turned off during pit stops. At the end of the trips we can see that the live data are decreasing a lot.



(a) ...



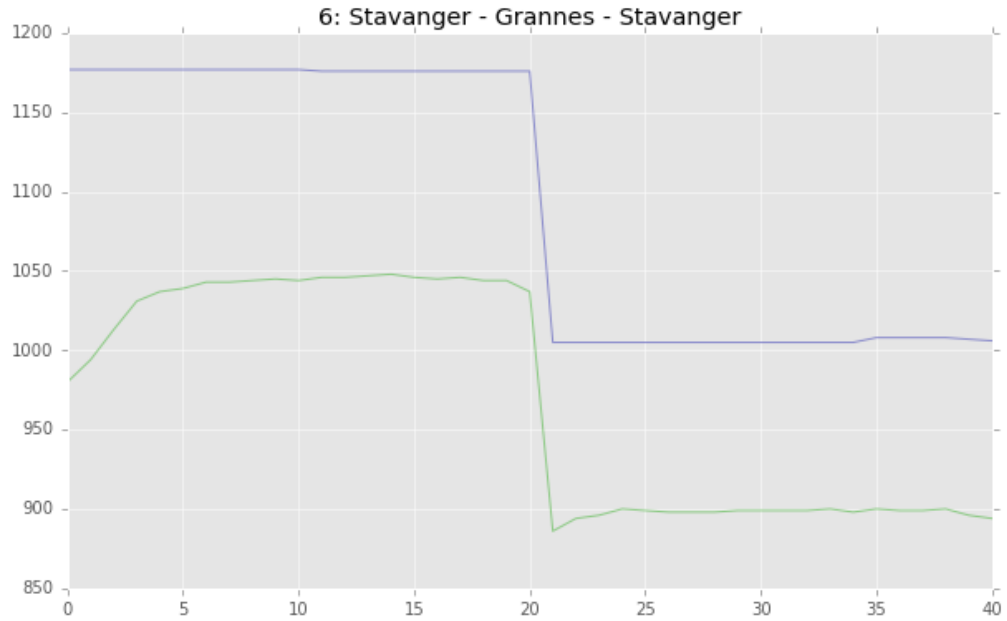
(b) First 10 stops

(c) Last 10 stops

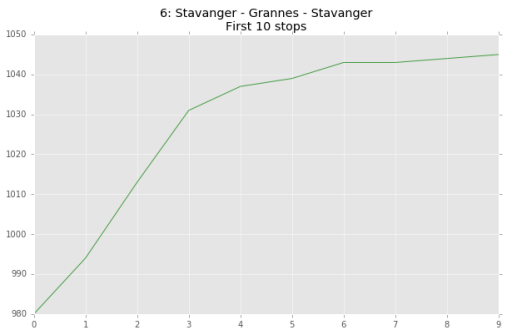
Figure 8: X76: Bus stop monitoring

The first 56 bus stops are the trip between Randaberg and Kvadrat, and the next 54 are the stops between Kvadrat and Randaberg. The peak on stop number 8 is because there are two stops with the same bus stop number, so it appears only once with the double amount of records instead of twice, one in each direction. Due to roadwork on Forus, some of the trips have a few temporary stops extra. This affects the graph where you can see them with both less live and total number of records.

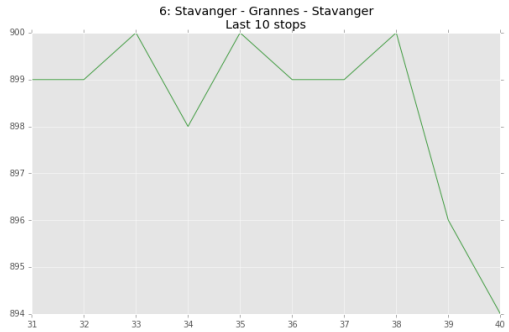
At the end of each of the trips, there are a fall in the number of records with live monitoring. This is especially visible on the trips from Kvadrat to Randaberg where the number of live monitored stops almost halved during the last ten stops.



(a) ...



(b) First 10 stops

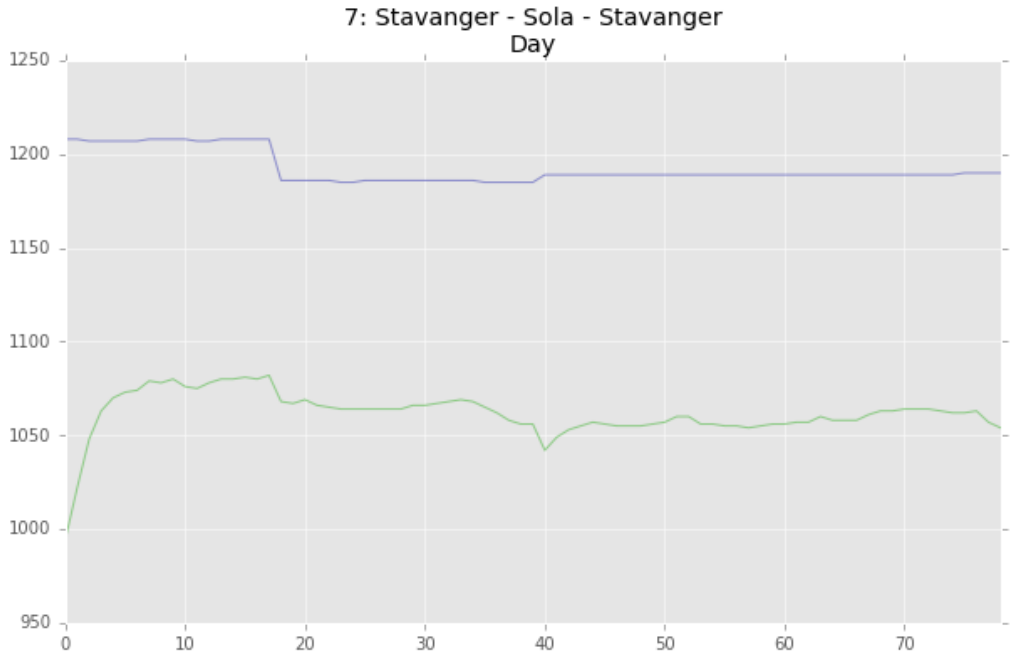


(c) Last 10 stops

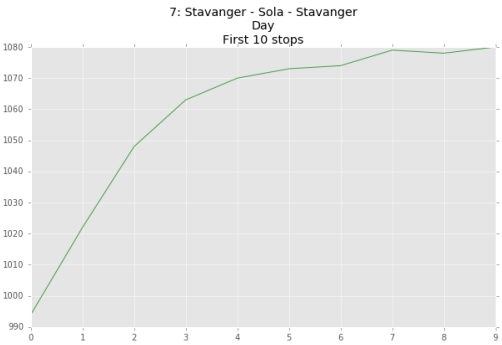
Figure 9: 6: Bus stop monitoring

The first 20 bus stops are from the trip between Stavanger and Grannes. The last 20 stops are the returning trip from Grannes to Stavanger, which have fewer departures.

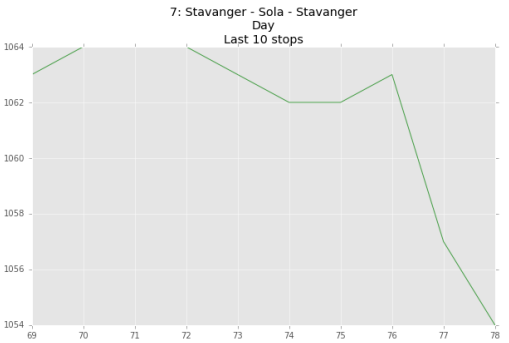
We can see that the live monitoring is less frequent at the beginning of the trips, and increases to max after a few stops. There is a slightly decreasing in the total number of monitored records at the ten last bus stops.



(a) ...



(b) First 10 stops



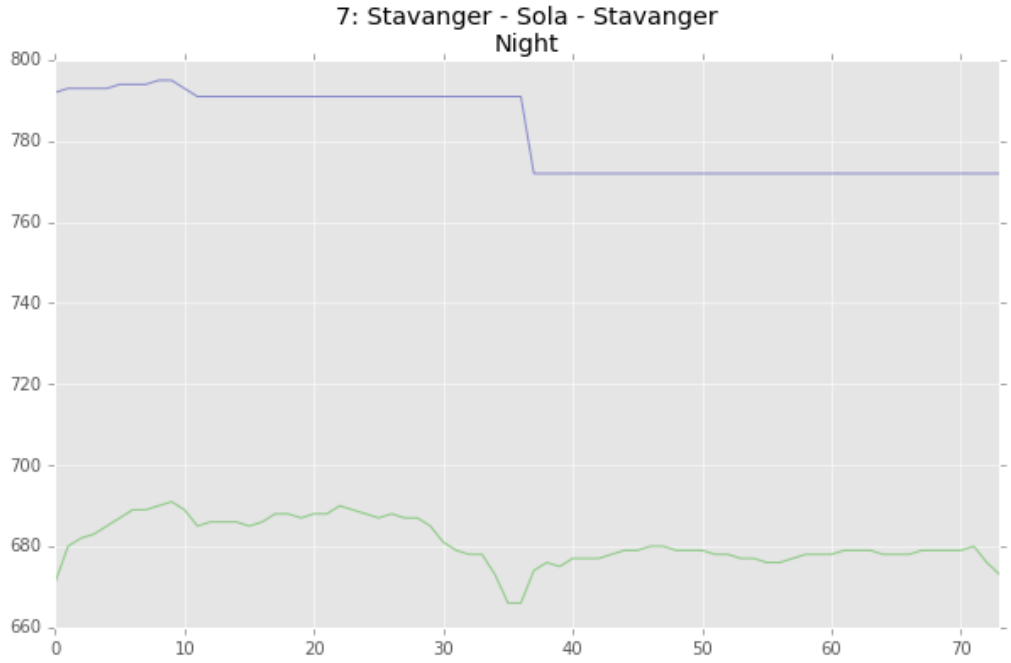
(c) Last 10 stops

Figure 10: 7(day): Bus stop monitoring

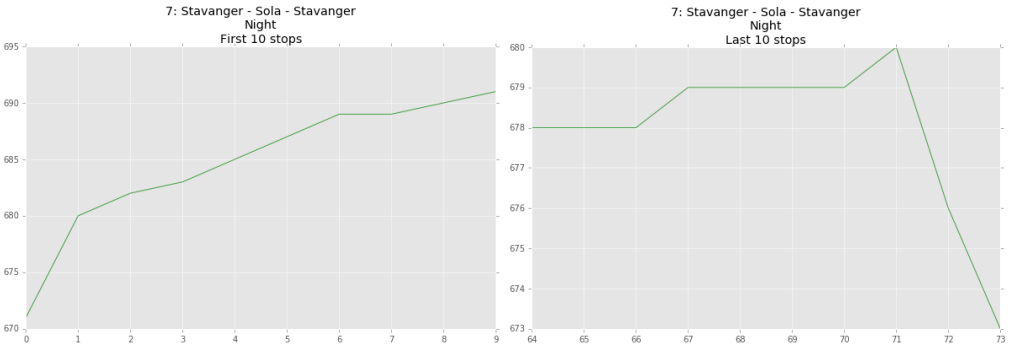
Figure 10 shows the graph from the day trips from bus line number 7. The first 40 bus stops are from the trips from Stavanger to Sola. The next 40 stops are the returning trips. In november and december 2015, there were

added an extra morning trip each week day from Stavanger to the University of Stavanger, resulting in the first 17 bus stops having a few more records.

Both the beginning and the end of the trips have a fewer number of monitored bus stops. This is especially seen in the beginning of the trips from Stavanger to Sola.



(a) ...



(b) First 10 stops **(c)** Last 10 stops

Figure 11: 7(night): Bus stop monitoring

Figure 11 shows the graph from the evening and night trips from bus line number 7. The first 40 bus stops are from the trips from Stavanger to Sola. The next 40 stops are the returning trips.

The beginning and the end of the trips have the same pattern as the day trips with a fewer number of monitored trips in both the end and the beginning of the trips.

2.3.3 Conclusion

This chapter have focused on the quality of the dataset. For further analysis it is important to find out how much of the dataset are relevant. In Section 2.3.1 we found out that the percentage of fully monitored trips lies around 80% for the bus lines X60, 6 and 7. For bus line number X76, the percentage is 50%. Looking at the partially monitored trips we find that the bus lines X60, 6 and 7 have 11%, 6% and 7% respectively while bus line number X76 have 31%. In Section 2.3.1 it turnes out that the partially monitored trips are mostly monitored, but lacking monitoring at the start or at the end of the trips. It would seem that there are a tendency to turn on the monitoring system a few stops after starting the trip and to turn it off a few stops before it ends. This results in a lower number of full monitored trips than the number of live records would imply. Even though there are trips with live data from almost all bus stops, they doesn't count as fully monitored which again decreases the quality of the overall dataset.

2.4 Other Quality Observations

Upon spending a lot of time looking at the dataset, a few observations have been made about discrepancies in the dataset. These observations are worth to mention before making a conclusion about the quality of the dataset.

2.4.1 Temporary Bus Stops

When roadwork have caused bus stops to physically move, the temporary bus stops are given a new ID. This creates problems when using Pandas integrated analysis functions, or when attempting to define the route of bus stops.

2.4.2 Decreasing Timestamp

A few times there have been encounters with trips where the timestamps at each of the record have been decreasing. This only lasted for a few records before the timestamps were back to normal again.

2.4.3 Stop Time and Time Between Stops

Looking at the highest values on the stop time and the time between stops fields, it is clear that some of the values must be wrong. Lots of records have a values up to 24 hours, which is the maximum value.

3 *Analysis*

3.1 Trip Delay Analysis

In this subchapter, we will take a look at the mean delay of the different trips for each of the four bus lines, looking for tendencies and patterns. For each trip ID, the median delay will be calculated at each of the bus stops. Visualizing this data will show the development of the delay throughout the trip.

We will also take a look at another way to work with the delay. By examining the delta delay between two bus stops, we will be able to find out how much the delay has increased or decreased between the stops. Regardless of the bus being on schedule or delayed, this will tell us more about the time it takes the bus to move from one bus stop to another. A high delta delay may imply a traffic jam or a miscalculation in the transit time between two stops. A negative delta delay implies that the bus have moved faster than scheduled, and therefore was able to decrease the overall delay.

Neither delay or delta delay were a part of the original dataset, but were added as columns at the pre-processing stage. The values are measured in seconds.

Since each of the bus lines have departures from at least two different terminals, managing a DataFrame for each of them manually would require a lot of duplicate code. Therfore, the creation of the DataFrames is accomplished in the function `create_trip_dict(df, cols, col)`

However, the splitting of the dataset based on the departure terminal will also split extended trips. Bus line X60 is mainly operating back and forth between Stavanger and Sandnes, although the morning trips departure from Hundvåg before they continue from Stavanger to Sandnes The afternoon trips from Sandnes to Stavanger continues to Hundvåg, but since the departure terminal is the same, they will be included in the same DataFrame. While a workaround would be to also include the arriving terminals as a condition, this would create a lot more DataFrames because of the different temporary stops in the end of some of the bus lines routes.

```
def create_trip_dict(df, line):
    dep_dict = {k: list(v.unique()) for k,v in bd[bd.St.isin(
                ['Departure terminal'])].groupby('Bid')['Sid']}

    routes = {i:bd[bd.IDt.isin(bd[(bd.Bid == line) &
              (bd.St == 'Departure terminal')&
              (bd.Sid == i)][['IDt']].squeeze().unique())]
              for i in dep_dict[line] }
    return routes

bus_lines = ['1033', '1039', '1007', '1008']
all_trips = {line:create_trip_dict(bd, line) for line in bus_lines}
```

Listing 11: Creating a dictionary of dictionary of DataFrames

In Listing 11, the function `create_trip_dict(df, line)` creates a dictionary of dictionaries of DataFrames. The first dictionary have the bus line IDs as key and the departure terminal ID as the key. In the second dictionary, the departure terminal ID is the key and the DataFrame based on the line number and the departure terminal is the value.

The function `plot_trips(df, cols, col)` takes in a DataFrame, a list of columns where the values in the first will become the new index and the values in the second will become the new columns. The values in last column of the DataFrame will become the values.

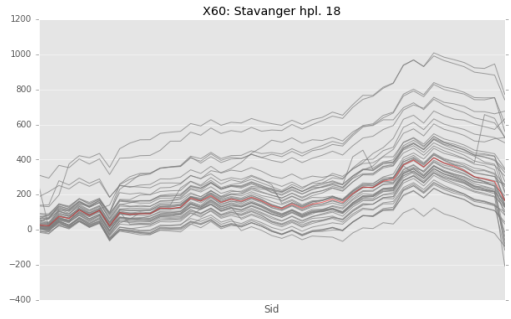
```
def plot_trips(df, cols, col):

    line = ''.join(df.head().Bn.unique())
    dep = ''.join([ch for ch in df.head().Sn.squeeze().unique()[0]
    if ord(ch)<= 128])
    tt = line + ": " + dep
    a = 0.7

    result = df.groupby(cols, sort=False)[[col]].mean().unstack()
    relevant_stops = [s for s in result.index if len(s) >= 4]
```

```
10      result = result[result.index.isin(relevant_stops)]
11
12      fig = plt.figure(figsize=(10,6))
13      ax = fig.add_subplot(111)
14      result.plot(kind='line', ax=ax, alpha=a, legend=False, color='
        grey', title=tt)
15      result.T.mean().plot(kind='line', ax=ax, alpha=0.5, legend=
        False, color='red')
16      ax.axes.get_xaxis().set_ticks([])
17      fig.savefig('pictures/tripdelay/'+ col + ''.join(e for e in tt
        if e.isalnum()), bbox_inches='tight')
18
19      print(tt,'\n' , result.mean().describe())
20
21  [plot_trips(all_trips[x][y],['Sid','IDt'], 'La') for x in all_trips
        for y in all_trips[x]]
22
23  [plot_trips(allofthem[x][y], ['Sid','IDt'], 'La_d') for x in
        allofthem for y in allofthem[x]]
```

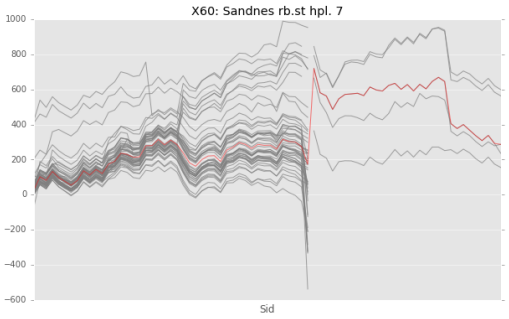
Listing 12: Transform and plot the dataset

The function is called with the list of bus stop IDs, trip ID and the delay, creating a new DataFrame called `result`. This DataFrame have bus stop ID as the index, trip ID as the columns and the mean delay as the values. Plotting the DataFrames results in the graphs below. Positive values indicate the delay when arriving at a bus stop, while negative values indicate that the bus arrived before the scheduled arrival. The graphs show the changes in the delay for all the unique trips where each gray line represent the mean of the trip while the red represent the mean of the mean. The mean of the mean is the mean changes in the delay for the given bus line.

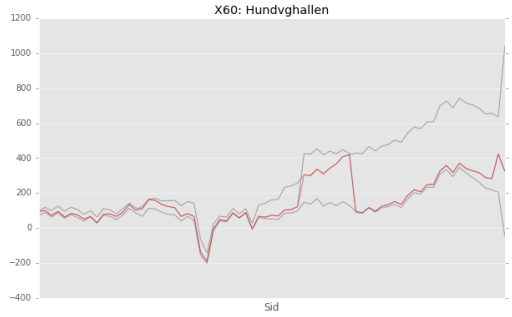
Using the integrated groupby-function in Pandas, any temporary bus stop would be shown in the end of the x-axis which may cause confusion about the end of the trip. Therefore, while temporary stops are still a part of the DataFrame, they will not be included in the visualization.



(a) Stavanger - Sandnes



(b) Sandnes - Stavanger - Hundvåg



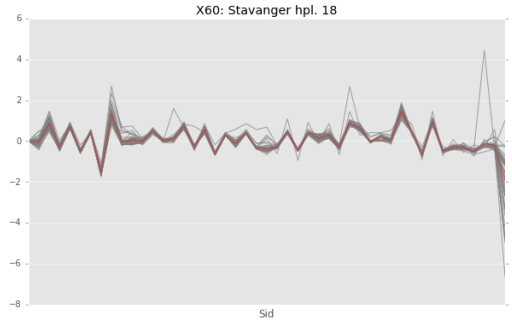
(c) Hundvåg - Stavanger - Sandnes

Figure 12: X60: Mean trip delta delay

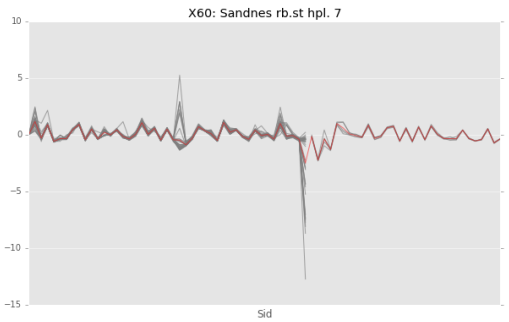
There are created three graphs for bus line number X60 since there are three different places the trips departure from. In Figure 12a, the delay at each bus stop increases slowly throughout the trip. At Forus, at the end of the trips, the delay increase moderately before decreasing at the end. We can see that some of the trips already are delayed when departing the terminal.

Figure 12b shows the trips departing from Hundvåg. These trips are only operating in the morning hours, resulting in few different trips. The low point is reached when arriving at Stavanger centre, indicating extra stop time to prevent delay before departure from Stavanger.

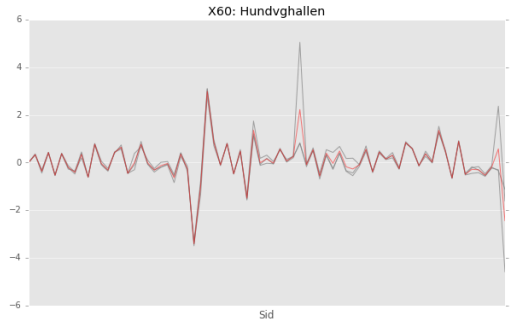
In Figure 12c, the delay increases from start and have a peak at Forus before falling slightly and increasing until arriving at Stavanger. This figure also includes trips who proceed to Hundvåg, which is the afternoon trips. The mean trip delay shows a steep peak, which is caused by the delay now being calculated by fewer trips. It is clear that the bus line experiences high delay time in the afternoon trips, but the delay reduces halfway around Hundvåg.



(a) Stavanger - Sandnes



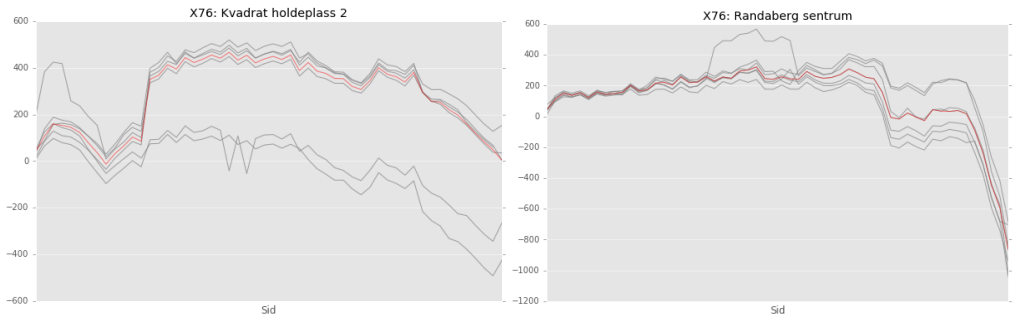
(b) Sandnes - Stavanger - Hundvåg



(c) Hundvåg - Stavanger - Sandnes

Figure 13: X60: Mean trip delta delay

In Figure 13 the delta delay per trip is shown. Here we can clearly see that the trips follow the same pattern regardless of the delay. The peaks tells us at which bus stops there are a significant change in the delay. The deviation lines indicates the correlations between a trip and a given bus stop.



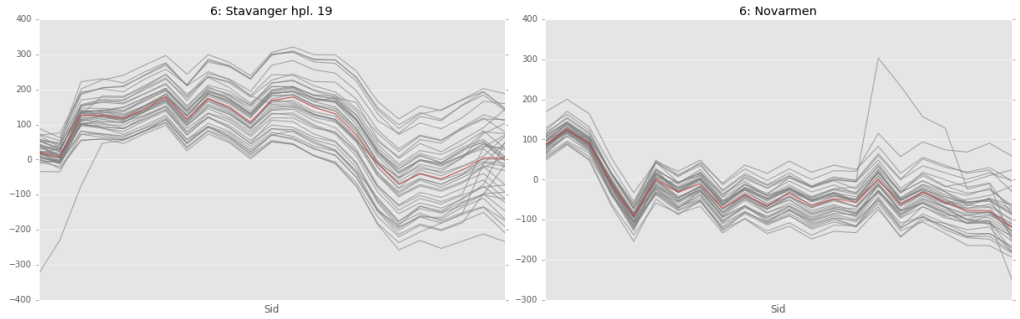
(a) Kvadrat - Randaberg **(b)** Randaberg - Kvadrat

Figure 14: X76: Mean trip delta delay

Bus line number X76 is aimed at people living between Randaberg and Ullandhaug and working in Forus, connecting the places without having to go through Stavanger centre. The bus uses the highway to make the trips faster. In the morning hours, trips are departing from Randaberg and in the afternoon hours, they are departing from Forus.

The trips for bus line number X76 from Kvadrat to Randaberg, shown in Figure 14a, only include afternoon trips. The trips are mostly starting without delay, but are experiencing some delay at Forus. The trips departing at 15:25 are already late, resulting in higher delay compared to the other trips at Forus. The delay increased significantly between the last bus stop at Forus and the first at Ullandhaug. There are almost ten minutes scheduled time between the two stops. In the middle of the route, the median delay is relatively even with a slow decreasing before increasing at Kvernevik. In the end of the route, the delay is decreasing past zero for some of the trips, resulting in early arrivals at the bus stops. If the bus does not stop to take on more passengers, it would explain why the delay keeps falling.

The median delay of the morning trips from Randaberg to Forus, shown in Figure 14b, is slowly increasing. At the trip departing at 07:30, the delay is doubled at the bus stops between Sunde and Madla. The delay is falling significantly at the first stops at Forus. There are a small bump before the median delay is falling considerably. If the bus does not stop to take on more passengers, it would explain why the bus continuously arrive earlier and earlier.

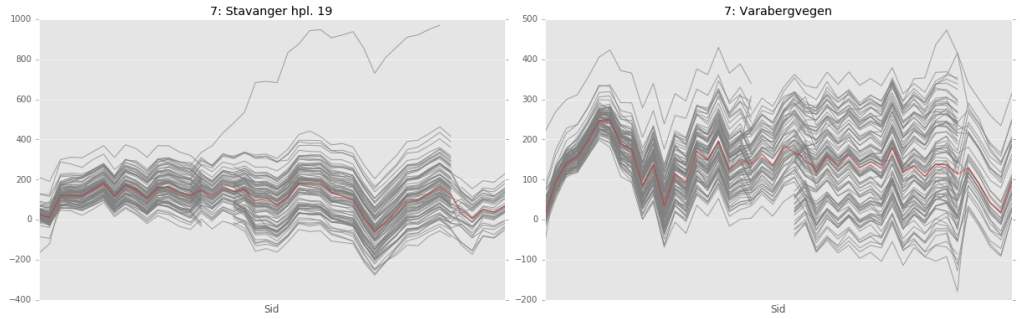


(a) Stavanger - Grannes **(b)** Grannes - Stavanger

Figure 15: 6: Mean trip delta delay

Bus line number 6 contains a relatively short route between Stavanger and Grannes, which is located close to the University of Stavanger. The median delay on the trips from Stavanger to Grannes, shown in Figure 15a, have a steep increase after the trip has started. The first trip at the weekday, with a scheduled departure at 06:30, have a tendency to arrive early at the first bus stops. The two first stops are different terminals around the bus station, but with a negative median delay on the third stop, the bus must have left the second bus stop before schedule. At the bus stops around the University, the delay is decreasing considerably, resulting i negative delay for more than half of the trips.

The trips from Grannes to Stavanger, shown in Figure 15b, are already delayed at the first bus stop, but after arriving at the University a few stops later, the delay becomes negative. The trips are relatively consistent up to two minutes delayed or ahead of schedule at each bus stop, while the delay decreases in the end. One trip is sticking out with a peak on the bus stops around Tjensvollkrysset.Tjensvollkrysset is one of the most trafficked roundabouts in the city.



(a) Stavanger - Sola **(b)** Sola - Stavanger

Figure 16: 7: Mean trip delta delay

Since bus line number 7 are using a slightly different route at night, some of the bus stops are not presented in the right order. At the end of the route, five extra bus stops from the night trips have been added. However, this does only affect the visualization.

Figure 16a shows the median delay on the trips from Stavanger to Sola. The delay is slightly increasing. However, the bus stops around the University of Stavanger, the median delay is at a low point. The first trip at the weekday, with a scheduled departure at 07:10, have a tendency of arriving early, but after a few stops it is back on schedule. The one trip in this graph sticking out the most is the 07:45-trip, with a median delay up to fifteen minutes. This is the most commonly delayed trips by far.

The median delay on the trips from Sola to Stavanger, shown in Figure 16b, have the first peak at Sande and the next peak near the University of Stavanger.The 15:25-trips has an average delay of three minutes when the trip starts. This causes the whole route to be delayed.

Table 6: Route overview

Name	Count	Mean	STD
X60 (Stavanger)	46	218.40	139.27
X60 (Sandnes)	50	253.48	142.53
X60 (Hundvåg)	3	191.79	91.47
X76 (Randaberg)	7	126.61	61.54
X76 (Kvadrat)	6	193.42	173.53
6 (Stavanger)	43	75.75	68.51
6 (Grannes)	39	-27.42	33.73
7 (Stavanger)	94	116.70	86.16
7 (Sola)	93	142.67	62.67

Table 6 is created from the DataFrames plotted above. It contains information about how many trips were used in the calculation, the mean delay of all the trips and the standard deviation.

3.2 Bus Stop Delta Delay Analysis Based on the Rush Hour

During the rush hours at the morning and afternoon, there are more busses than usual. Even so, the busses are barely on time, and when they do arrive, there are often many similar busses in line. The traffic in the rush hours are hard to predict, but by watching the delay develop bus stop by bus stop, possible bottlenecks might become visible. Today's bus schedules have the same estimated time between stops at each trip, including the trips during the rush hours. It is not plausible that the bus uses the same amount of time on the same trip either it is in the rush hour or not, so finding a better schedule during the rush hour would both help the bus holding the schedule and the passengers catching the bus. In this subsection, we will check if there are any bus stops acting as bottle necks in the system during the rush hours. To achieve this, we will have to find out when the rush hours are to see if there are any bus stops with significantly high delta delay.

3.2.1 Finding the Rush Hours

When talking about the rush hours, there are typically two times a day when the load on the road network is increased; in the morning and in the afternoon. On weekdays, there are lots of people travelling from one location to

another at these times a day, resulting in a lot of vehicles on the road. By looking at the median delay for all relevant bus lines with a time span of five minutes, we will be able to see what times a day the delay is increased.

```
rush = bd[(bd.As.dt.dayofweek <5 )]
rush = rush.set_index('As')
rush = rush.resample('5T', how = 'median').dropna().reset_index()
times = pd.DatetimeIndex(rush.As)
result = rush[['As', 'La']].groupby([times.hour, times.minute])
```

Listing 13: Finding the rush hours

Listing 13 shows how the delay is presented by time of day. The data recorded at weekdays are re-sampled in a five minutes time spans. By creating a DatetimeIndex of the scheduled arrival time, we are able to group the dataset by the hour and minute to combine trips from all the days in the dataset. Plotting the data gives the the figure below.

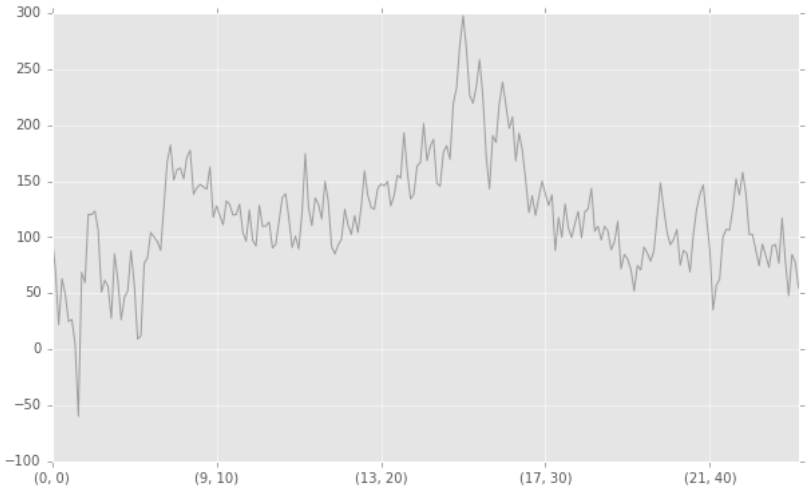


Figure 17: The mean delay by the time of day

As we can see from Figure 17, the highest peak is located from 15:10 to 17:00. The delay have a smaller peak in the morning between 07:30 to 09:05.

3.2.2 Bus Stop Delta Delay Analysis

In this subsection, we will check if there are any bus stops acting as bottle necks in the system. To achieve this, we will use data from the previous achieved rush hours to see if there are any bus stops with significantly high

delta delay. By comparing the bus stops with highest delta delay from the rush hour with other times at the day, we will find out if the bottlenecks are based on time of the day.

```
bm = bd[(bd.As.dt.dayofweek <5 ) &
        (bd.As.dt.time>=dt.time(hour=7, minute=30)) &
        (bd.As.dt.time<=dt.time(hour=9, minute=5))] \
        [['IDt']].squeeze().unique()

ba = bd[(bd.As.dt.dayofweek <5 ) &
        (bd.As.dt.time>=dt.time(hour=15, minute=10)) &
        (bd.As.dt.time<=dt.time(hour=17, minute=0))] \
        [['IDt']].squeeze().unique()

br = bd[(bd.As.dt.dayofweek <5 ) &
        ~bd.IDt.isin(bm) & ~bd.IDt.isin(ba)] \
        [['IDt']].squeeze().unique()
```

Listing 14: Finding trips

In Listing 15, three different subsets are created from the trips found in Listing 14. `bm`, `ba` and `br` are the morning, afternoon and the rest of the non rush hour trips from the weekdays. Some of the bus lines have overlapping bus stops, so the subsets contains trips from all four bus lines to increase the data we have on each bus stop.

```
q0 = bd[['As', 'Dt', 'IDt', 'Sid', 'La', 'Sn', 'Bn', 'St', 'Bid', '
    Vid', 'La_d']]
q0 = q0.dropna()

q0_m = q0[q0.IDt.isin(bm)]

q0_a = q0[q0.IDt.isin(ba)]

q0_r = q0[q0.IDt.isin(br)]
```

Listing 15: Creating new datasets

By sorting the datasets by the highest median of delta delay, shown in Listing 16, a list of the bus stops with the highest mean delta delay is made. For each of the different time windows, there are different dominated bus stops.

```
q0_m.groupby(['Sn']).median().sort_values('La_d', ascending=False)
q0_a.groupby(['Sn']).median().sort_values('La_d', ascending=False)
q0_r.groupby(['Sn']).median().sort_values('La_d', ascending=False)
```

Listing 16: Get highest delta delay

Table 7: Mean delta delay for morning trips

Bus Stop	Delta Delay
Kannik	2.108311
Eikebakken	1.549132
Mosvangen camping	1.371247
Ytraberget	1.305530
UiS ved Kjølv Egelands hus	1.268095

Table 8: Mean delta delay for afternoon trips

Bus Stop	Delta Delay (min)
Grannesveien sør	3.094309
Kannik	2.229691
Mosvangen camping	1.735535
Seljeveien	1.665542
Strømsbrua	1.367019

Table 9: Mean delta delay for regular trips

Bus Stop	Delta Delay (min)
Kannik	1.814289
Seljeveien	1.481366
Eikebakken	1.421621
Ytraberget	1.281248
UiS ved Kjølv Egelands hus	1.262509

From Table 7, we can see that the bus stop "Kannik" is the bus stop with the highest mean delta delay in the morning hours. It is also present in all of the tables, which indicates that this bus stop is causing delay regardless of the time of day. During the afternoon hours shown in Table 9, "Grannesveien sør" is contributing to an increase of the delay of three minutes on average.

3.2.3 Delay at a given bus stop

Looking at the delay at a single bus stop will tell us more about how the passenger experiences the delay. We will take a look at bus stops at "Uis Øst" (stop ID '11031532'), located at the University of Stavanger. The bus stop is used by bus lines number 6, 7 and X60 and recieves trips from Stavanger and connects the passengers to Kvadrat, Sola and Grannes. The bus stop experiences a lot of passenger traffic in the weekdays. We will take a look at the scheduled arrival time for each trip and find the median delay. This way, one can see how long the average wait time is for each bus.

```
def create_stop_overview(bd, stop):
    bd = bd[bd.Sid.isin([stop]) &
            (bd.As.dt.dayofweek <5)][['As', 'La', 'IDt', 'Bn']]
    times = pd.DatetimeIndex(bd.As)
    result = bd.groupby([times.hour, times.minute, 'Bn']).median()
    return result

create_stop_overview(bd, '11031532')
```

Listing 17: Finding average wait time per trip for a given bus stop

In the function `create_stop_overview(bd, stop)` showed in Listing 17, weekday data from the given bus stop is extracted and the trips are grouped by the time of day and the line number. The median delay is calculated for each trip. The results have a multiindex of the hour and the minute for the given trip.

8	2	7	106.0
	3	X60	187.0
	7	7	75.0
	12	7	62.0
	17	6	30.0
	18	X60	232.0
	32	7	69.0
	33	X60	113.0
	47	6	-26.0
	48	X60	164.0

Figure 18: Scheduled arrival times and median delay for trips at bus stop "Uis Øst" ('11031533')

Figure 18 shows the trips scheduled to arrive between 08:02 and 09:48. All trips departed from Stavanger centre, however, they do not operate on

the same route. Bus lines 6 and 7 uses a route past Madla, and X60 uses a faster route past Hillevåg.

We can clearly see that bus line number X60 have the greatest median delay at the given time frame, up to four minutes. From the previous sections we know that the trips for that bus line have a close to linear increase in the delay over time, so a delay was expected. Bus line number 7 have a median delay of approximately 1-2 minutes while number 6 ranges from +-0.5 minutes.

Given the fact that this bus stop is located at the University, we will assume, without having any passenger count available, that there is a great number of passengers before lecture start at 8:15. We can see that the median delay does not cause the relevant trips to arrive after lecture start. After the lectures have started however, bus line X60 with a scheduled arrival time at 08:18 have a tendency to arrive almost four minutes late.

15	2	7	62.0
	3	X60	139.5
	17	6	16.5
	18	X60	349.0
	32	7	77.0
	33	X60	316.5
	47	6	99.5
	48	X60	224.0
16	2	7	86.5
	3	X60	413.0
	17	6	-25.5
	18	X60	164.0
	32	7	50.0
	33	X60	174.0
	47	6	-43.5
	48	X60	117.0

Figure 19: Median delay at bus stop "Uis Øst" (’11031532’)

Figure 19 shows the same bus stop between 15:02 and 16:48. Bus line X60 stands out with the greatest median delay. While the other lines have up to 2 minutes delay, X60 have up to almost 7 minutes. It is worth mentioning that line 6 started the trips only a few bus stop prior to this one.

4 *Conclusion*

4.1 Conclusion

The first objective in this thesis was to evaluate the quality of the data used for the case study. The quality was evaluated by how well the trips were monitored. Based upon the evaluations in Chapter 2, it is shown that the dataset have reduced data quality, but despite of the reduced number of fully monitored trips, the dataset is assessed to be sufficient for establishing a reference case for further studies. In addition, my findings suggest that actions must be taken in order to improve future data quality. This may be achieved both by improving the system, and by strengthen the driversunderstanding of the importance of continuous logging.

The second objective was to observe and analyse the development of the delay for all trips. The first part of Chapter 3 visualizes the different tendencies. The graphs indicates that some trips experienced delay caused by external factors. One of the most important factors was identified as heavy traffic in rush hours, and this made a basis for the next objective. Delta delay per bus stop was also examined in order to visualize the difference between the bus stops regardless of the delay.

The third objective was to explore the rush hours. The rush hour was defined from the route delay. The mean delta delay was calculated per bus stop to look for bottlenecks in the system. It has been presented bus stops

which significantly increases the delay, and these stops should be evaluated more closely for possible adjustment of the time schedule.

4.2 Further Work

The bus schedule is the same each week, and it is divided into weekdays, Saturdays and Sundays. The trips each weekday are the same, which makes it possible to compare them to see if there are any correlation between them and the delay. However, there might be some days with lower delay than other so it might be interesting to check if there are any changes in the delay between the days of the week. It could also be an idea to compare data from one time of the year with another.

The estimated time between stops is calculated by a rough estimate on how long it takes to travel from the previous bus stop. There are no estimate on the stop time for the bus at the bus stop. Some bus stops have few to no passengers during the day while others might lay close to schools or crossing bus routes, which leads to a greater flow of people. It might be interesting to see if some bottlenecks are occurring due to lack of estimated stop time at the bus stops. Now that Kolumbus have increased the ticket price for paying on board the bus, the amount of passengers who are pre purchasing the ticket online is likely to increase. This would result in faster boarding time at the bus stop, so if we get access to newer data from Kolumbus, it might be possible to compare stop times for bus stops with a great passenger flow to see if it affected the delay of the stop.

Some bus stops are so close together that the estimated time between stops are zero. By clustering similar bus stops together, the clusters might give a better picture of the evolving of the delay for the whole route. There might be tendencies and patterns in the delay for each day or for each week that can be used to predict delay.

References

[1] European Environment Agency (EEA). Occupancy rates of passenger vehicles. Online resource. http://www.eea.europa.eu/data-and-maps/indicators/occupancy-rates-of-passenger-vehicles/occupancy-rates-of-passenger-vehicles-1.

[2] Thomas Bore Olsen. Bussvei 2020. Online resource. http://www.stavanger.kommune.no/no/Tilbud-tjenester-og-skjema/Vei-og-trafikk/Trafikk/Bussvei-2020/.

[3] Winfried Bruns. Siri (service interface for real-time information). Online resource. http://www.datex2.eu/user-forum/$2_B runs_S IRI.pdf$.

[4] Transit Cooperative Research Program (TCRP). About tcrp. Online resource, . http://www.tcrponline.org/SitePages/aboutTCRP.aspx.

[5] Transit Cooperative Research Program (TCRP). Data analysis for bus planning and monitoring. Online resource, . http://onlinepubs.trb.org/onlinepubs/tcrp/tsyn34.pdf.

[6] Transit Cooperative Research Program (TCRP). Using archived avl-apc data to improve transit performance and management. Online resource, . http://onlinepubs.trb.org/onlinepubs/tcrp/tcrp$_r pt_1$13.pdf.

[7] Theo H.J. Muller and Peter Knoppers. Tritapt - trip time analysis in public transport. Online resource. http://tritapt.nl.