



CHALMERS



# Diagnostic Trouble Code Analysis

A statistical approach, surrounding fault codes within Volvo Group Electromobility

Bachelor's Thesis in Electrical Engineering, EENX20

JOHANNES BENGTNER  
ALEX SMILEVSKI

---

Department of Electrical Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2022  
[www.chalmers.se](http://www.chalmers.se)



BACHELOR'S THESIS 2022

# Diagnostic Trouble Code Analysis

A statistical approach, surrounding fault codes within Volvo Group  
Electromobility

Johannes Bengtner  
Alex Smilevski



**CHALMERS**

Department of Electrical Engineering  
*Division of Electric Power Engineering*  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2022

Diagnostic Trouble Code Analysis

A statistical approach, surrounding fault codes within Volvo Group Electromobility

JOHANNES BENGTTNER

ALEX SMILEVSKI

© JOHANNES BENGTTNER, ALEX SMILEVSKI, 2022.

Supervisor: Prashanth Rudramuny Ganada, Volvo Group

Examiner: Torbjörn Thiringer, Department of Electrical Engineering

Bachelor's Thesis 2022

Department of Electrical Engineering

Division of Electric Power Engineering

Chalmers University of Technology

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Cover: An Electric Volvo bus at its charging station.

Typeset in L<sup>A</sup>T<sub>E</sub>X

Gothenburg, Sweden 2022

# Abstract

Since Volvo Group is using data analytic methods to process fault codes, which were obtained by data extraction via the workshop tools and the transmitters, the two thesis workers from the electrical engineering line have decided to explore and make a deeper analysis of why and how often various error codes occur. The method used to do this type of in-depth analysis is via information gathering, statistical formulas and sort-programming via Python. This is done with an external use of Excel to be able to visualize the statistical work that was achieved via graphs and probability theory.

The work focuses on component fault codes that arises in the hybrid/electric vehicles within Volvo buses and medium duty trucks. The main work on how Volvo GTT can apply transmitters in a part of the vehicle fleet has been calculated using statistical formulas against the error codes to reach its entire population.

The work included useful information from many different sources, such as interviews, searches on Google and within Volvo Group's own information databases. The report also include how the triggering conditions in vehicles sensors behave, how the database/data flow works as well as the general information gathering regarding standardization's, such as the ISO standard OBD-II and the ISO/SAE standard 15031-6 for the DTC format.

Keywords: DTC, Diagnostic, Trouble Code, Analysis, Error Code, UDS, OBD, ECU, CAN-bus, Python, Pandas, FTB, Statistics, Volvo, Trucks, Group, GTT

# Sammanfattning

Eftersom Volvokoncernen använder dataanalytiska metoder för att bearbeta felkoderna, som man får genom dataextraktion via OBD-scannern eller de trådlösa sändarna, så har de två examensarbetarna från den elektrotekniska linjen beslutat sig för att utforska och göra en djupare analys om varför samt hur ofta diverse felkoder förekommer. Metoden som användes för att göra den här typen av fördjupningsanalys är bland annat via informationssamlande, statistiska formler och stortingsprogrammering via Python. Detta utförs med en extern användning av Excel för att kunna visualisera det statistikarbetet som uppnåddes via grafer och sannolikhetslära.

Arbetet har fokuserats kring de hybrid/elfordon inom Volvobussar och medelstora lastbilar samt de komponent felkoder som uppstår. Huvudarbetet kring hur Volvo GTT kan applicera sändare i en del av fordonsflottan har kalkylerats fram med hjälp av statistiska formler gentemot felkoderna för att kunna täcka hela sin population.

Arbetet har inkluderat nyttig information från många olika håll, såsom intervjuer, sökningar på Google och inom Volvo Groups egna informationsdatabaser. Rapporten inkluderar också hur utlösningförhållandena i fordonssensorer beter sig, hur databasen/dataflödet fungerar samt den allmänna informationsinsamlingen om standardiseringar, såsom ISO-standarder OBD-II och ISO/SAE-standarder 15031-6 för DTC-formatet.

Nyckelord: DTC, Diagnostic, Trouble Code, Analysis, Error Code, UDS, OBD, ECU, CAN-bus, Python, Pandas, FTB, Statistics, Volvo, Trucks, Group, GTT



# Acknowledgements

Firstly, we want to thank the whole data analytics team at Volvo Group, CampX. An extra thank you to Prashanth Ganada Rudramuny and Josefin Agnas who helped us with the coaching throughout the work. We would also like to thank our examiner Torbjörn Thiringer and Johan Tykesson who supported us with interesting tips, tricks and guidance.

Johannes Bengtner, Alex Smilevski, Gothenburg, June 2022







# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ART	Volvo Trucks Accident Research Team
CAN	Controller Area Network
CARB	California Air Resources Board
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
eMob	Electromobility
EU	European Union
EV	Electric Vehicle
FMI	Failure Mode Identifier
FTB	Failure Type Byte
GDPR	General Data Protection Regulation
GTT	Group Trucks Technology
HW	Hardware
IP	Intellectual property
ISO	International Organization for Standardization
MIL	Malfunction Indicator Lamp
OBD	On-Board Diagnostics
SAE	Society of Automotive Engineering/Engineers
SW	Software
UDS	Unified Diagnostic Service
VGCS	Volvo Group Connected Services



# Contents

<b>List of Acronyms</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	2
1.2 Purpose . . . . .	3
1.3 Limitations . . . . .	3
<b>2 Theoretical &amp; Technical Background</b>	<b>5</b>
2.1 ECU . . . . .	5
2.1.1 UDS . . . . .	5
2.1.2 Sensors . . . . .	5
2.1.3 CAN-Bus . . . . .	6
2.2 OBD . . . . .	7
2.2.1 OBD-II DTCs . . . . .	9
2.2.1.1 Trouble Code Subsystem . . . . .	9
2.2.1.2 Type of Code . . . . .	10
2.2.1.3 Affected Subsystems . . . . .	10
2.2.1.4 Specific Problem Code . . . . .	11
2.2.1.5 FTB/FMI . . . . .	11
2.3 Python . . . . .	11
2.3.1 Short Description of Code Progress . . . . .	11
2.3.2 Pandas . . . . .	12
2.3.3 Sorting with Jupyter Notebook . . . . .	12
2.4 Transmitter & Workshop Tools . . . . .	12
2.4.1 The Data . . . . .	13
2.4.2 Transmitter software . . . . .	13
2.5 General Volvo Group Information & eMob . . . . .	14
2.5.1 Batteries . . . . .	14
2.5.2 Combustion Efficiency & Environment . . . . .	15
2.6 Important relations . . . . .	15
<b>3 Case Set-up</b>	<b>19</b>
3.1 Implementation . . . . .	20
3.1.1 Data Collection . . . . .	20
3.1.2 Python sorting . . . . .	20
3.1.3 Statistical Methods . . . . .	20

3.1.4	Excel . . . . .	20
3.1.5	Ethics . . . . .	20
<b>4</b>	<b>Requirements</b>	<b>21</b>
4.1	IP . . . . .	21
4.2	GDPR . . . . .	21
4.3	Battery Directive EU . . . . .	22
4.4	Road safety . . . . .	22
4.5	Emission regulations . . . . .	23
<b>5</b>	<b>Results &amp; Discussion</b>	<b>25</b>
<b>6</b>	<b>Conclusion</b>	<b>35</b>
	<b>Bibliography</b>	<b>37</b>
<b>A</b>	<b>Appendix 1</b>	<b>I</b>

# 1

## Introduction

In an ever-changing transport industry, with electrification being the new future standard, trucks and buses around the world are increasingly getting developed and becoming more and more intelligent. With the evolution of smart vehicles, facilitation benefits and disadvantages occur. Due to newly implanted, electrified systems in the vehicles, engineers are in need to develop a way to map and manage its systems.

In 1971, vehicles began to use a simpler type of electronic control unit in the engines. Previously, a mechanical system was used for the fuel injection instead. Because of the stricter rules and regulations that emerged, regarding emissions in 1975, the development of electronic ECUs led to better- reliability and precision for combustion in the vehicle [1]. With time, ECUs advanced even further and began to be used even more widely within the vehicles. The electronics around the vehicle was streamlined and its functions even more so, such as, brakes, transmission and steering.

The efficiency did not stop there, it got better with time, and nowadays, ECUs are also used to display even further, in-depth type of faults. By the creation of the international standard for the industry, that fits most companies within DTC maintenance, the customer and diagnostician are directly able to visualize the problems occurring within the vehicles systems.

The time consuming work of physically handling identification and locating faults throughout the vehicles therefore changed from being analogue, to be further digitized. In the beginning of the era with electrified trucks, Volvo Group chose to invest in fault code management. This was done in order to build upon the principle of reliable trucks, and thus remedy as many faults as possible within the winding of electrification. By avoiding expensive and unnecessary repairs for the customers, as one had to do in the past [2], it gives the company more time to focus on further system development.

Volvo Group receive these DTCs via the CAN-bus on-demand, transmitter -, or via the workshop tools which has the standardized OBD-II format, - from the ECUs. Volvo GTT has recently begun working with fault tracing analysis on a larger scale, facing these DTCs from different points of view.

The DTCs are stored in the company's database. Volvo Group does this to find out why they are generated, how often they are generated and what it might be depending on. By finding out the root cause of these errors, Volvo GTT can provide and ensure better reliability for the costumers which gives the end users a better experience.

The goal of this thesis is to find out how many high frequency transmitters is needed to be installed to cover the whole population of Volvo buses and medium duty trucks (hybrid/electric). The thesis work-around, will go on a deeper basis into why some of these DTCs occur, and how to go about counteracting these using statistics, probability theory and Python to sort out the essential data needed. The work will be specifically, concentrated on component fault codes, which includes important parts such as the ECUs, sensors and batteries. Volvo trucks/buses around the world will be compared to the different seasons to see if countries' mediums have an impact on the component's quality aspect within the vehicles.

### 1.1 Background

As previously mentioned in the introduction, Volvo GTT is working intensively with DTCs from different points of view and how faults might stack with one and each other. Volvo Group strives to map all the fault codes occurring in their vehicles and to understand the root cause behind these faults.

The work around DTCs is done to refine the quality of products and the improvement of their predictive maintenances. Key reasons like, why certain parts of the vehicle trigger problematic faults are also considered for future – repair and development of Volvo GTT's technology.

Volvo Group's goal is to constantly improve its quality of products, ideally, in the most flexible way possible. With the collection of data during vehicle use, Volvo GTT can compile excellent future predictions.

The total knowledge that Volvo Group's database receives, comes through the workshop tools at the workshops and the transmitter in the vehicles. The transmitter is the most effective way of gathering quality data from the transport vehicles. By the usage of the transmitter, data can be sent faster to the database, and the data sent is more reliable in relation to the workshop tools, since it does not include any HW plug-in connection or human factors/interactions collecting it.



## 1.2 Purpose

The purpose of this thesis project is to give Volvo Group a better insight and understanding of the probability cause of the DTCs, this with the help of Normal distribution, Central limit theorem, Poisson distribution and Confidence interval. The area of faults that will be specifically examined is within the field of component, received from the vehicle's transmitter. The two thesis workers will analyse the occurrences of these DTCs to see how often, try to find out why they happened and triggering conditions regarding these.

Furthermore, a target is to provide workable solutions to these faults, by investigating whether it's the differences in medium, SW versions or something else that might link to the disturbance. Along with the continuation of the work, this will to be identified during the stay at Volvo GTT.

## 1.3 Limitations

The work will be limited to only hybrid/electric Volvo- buses/trucks in medium duty. The two thesis workers did not take any position on the economic views of the work, for instance how much Volvo Group gains from doing this type of work. They will also limit themselves to how many DTCs they chose to examine.

Regarding the temperature differences between the countries' component fault codes, the thesis workers could not cover the area with reliable, statistical result since there was not enough data.

The statistical values will be based on the transmitter in the vehicle. The workshop tools data was not included because it is not as reliable as the transmitter's.



# 2

## Theoretical & Technical Background

### 2.1 ECU

The ECU (Electronic Control Unit) is used to monitor all the electrified systems and subsystems within the vehicle, which means it also affects every aspect of the vehicle's electronics. It prioritizes the safety systems onboard, but it is also available for the entertainment within, and even the lighting, when you for example open the doors [3]. The ECUs comes in various sizes, it can be everything from a small micro processing unit up to a more substantial sized embedded system.

These ECUs or nodes as they also are called, are processing units, which can send and receive messages. Depending on the unit, it can also store information about the health of the subunits controlled by the nodes. These nodes communicate with each other mainly through the CAN-bus [4].

#### 2.1.1 UDS

UDS (Unified Diagnostic Service) is an ISO standardized, communication protocol used for diagnostics within the ECUs of the vehicle. Using this protocol, you can read and clear fault codes, produce parameters such as temperature and modify the settings of the ECUs. Since its standardized, the protocol is used in the ECUs produced by the manufacturers [5].

By being an ISO protocol, it facilitates for the diagnosticians, it also makes it easier for all parties involved. This includes manufacturers, developments, testing, services and vehicle inspections. This also opens the possibility of easy access when it comes to renewing the ECU with firmware and software (Software) updates [6].

#### 2.1.2 Sensors

The sensors around the vehicle, sends data to the ECUs by their triggering conditions. Nearly all sensors experience errors after usage, due to the installation of the sensor. The errors that show up can differ, depending on what sort of sensor it is, but the most common type of errors that comes because of installation, is the following:

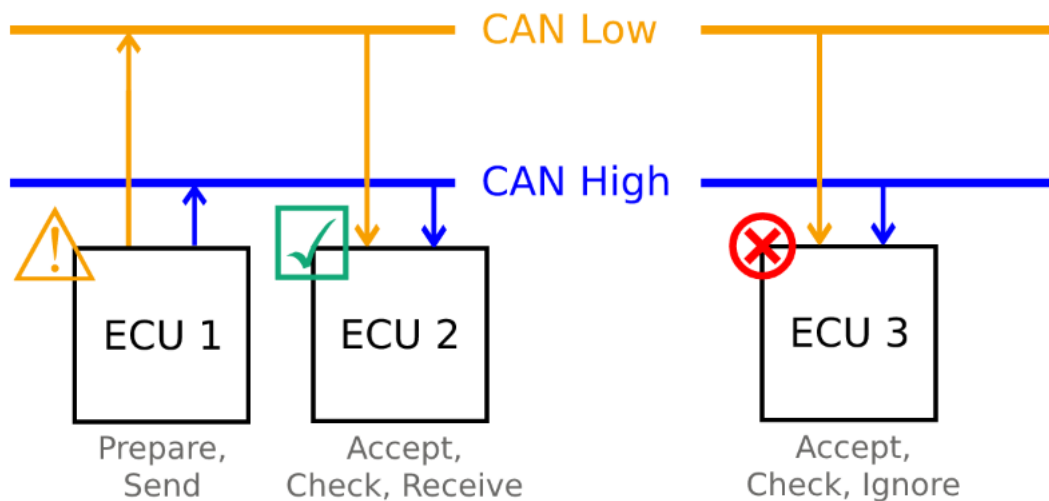
- Loading effect – The object’s characteristics may have changed when being measured.
- Supply of the sensor – When the temperature of the medium changes or if the voltage is generally affected over time.
- Grounding – Differential or common potentials can cause effects through improper grounding.
- Cable shielding – Depending on the method, one can minimize the electromagnetic field, which also has an impact on this type of error.

The sensors are used for example, to detect temperature, Speed, Fuel Quality, Oxygen (O<sub>2</sub>), Ammonia (NH<sub>3</sub>), Hydrocarbon or Pressure.

### 2.1.3 CAN-Bus

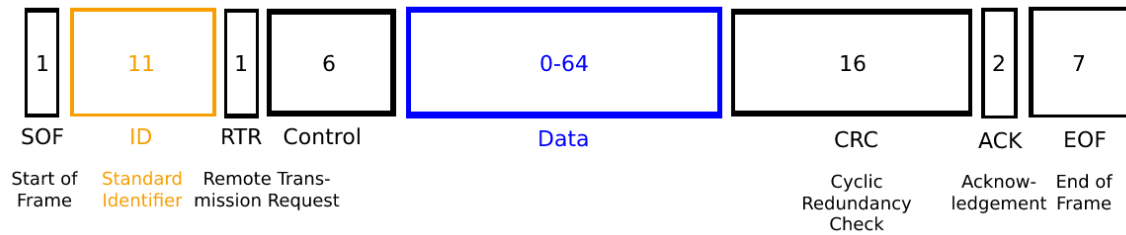
In the modern vehicle nowadays, every aspect of electronics is sent through a network via the onboard computer. The network standard used for this, is CAN (Controller Area Network) and was created by Bosch in 1986. The CAN-bus is a network connected by the nodes (ECUs). These nodes can be in control of the airbags, brakes or other connections, controlled by either the driver or the pre-programmed systems.

Every node consists of either a small “computer” (a micro processing unit) or a micro controller unit. The nodes communicate via the bus through two wires, a CAN-High and CAN-Low (Figure 2.1). By eliminating the need for excessive wiring, using a point-to-point communication, makes the CAN-bus superior to its predecessors.



**Figure 2.1:** The CAN-bus with nodes. An example where ECU 1 is asking the bus CAN Low to send a message, ECU 3 decline the message, ECU 2 accepts and receives the message via the CAN High node.

With this type of, one point of entry, makes the bus quite simple and reduces the cost. Messages which are sent via the bus are called frames (Figure 2.2), these frames consist of 11 bits (CAN 2.0A) or the extended 29 bit (CAN 2.0B) [7].



**Figure 2.2:** Standard CAN Frame.

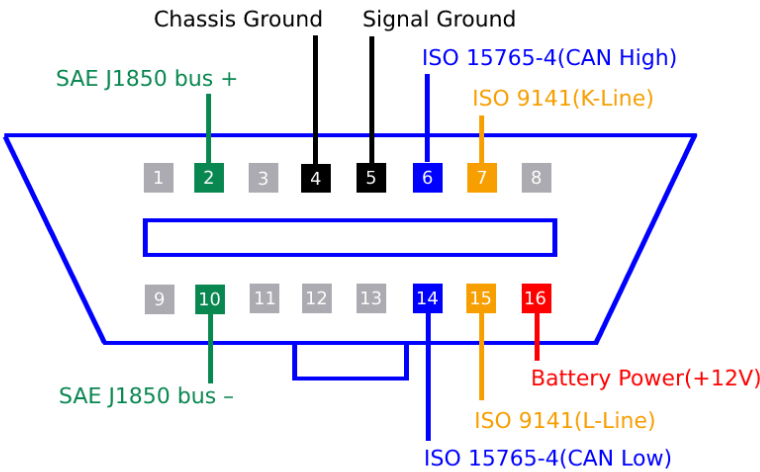
## 2.2 OBD

The OBD (On-Board-Diagnostic) port, is located close to the steering wheel. This connection is used to set up the communication between the vehicle and the OBD scanner. When the communication between the OBD scanner and the vehicle is maintained, the extraction of the vehicle's aggregated data can begin.

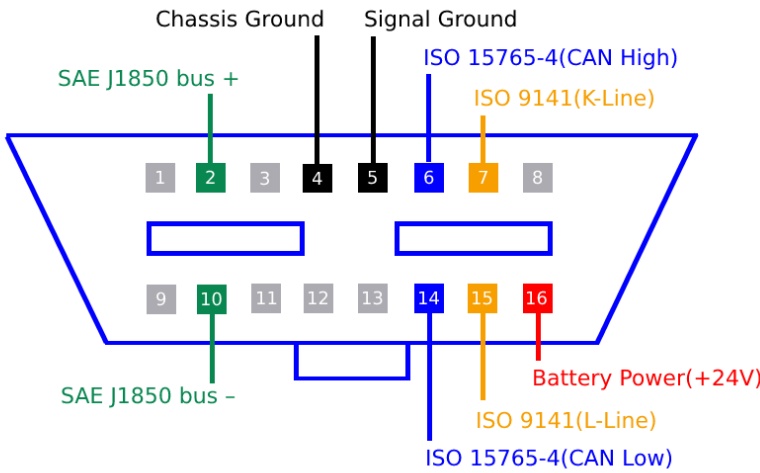
If the vehicle's MIL (Malfunction Indicator Lamp) is toggled on and the driver attends a workshop. The technicians at the workshop, uses an OBD scanner to pinpoint what is wrong vehicle and why the MIL is active [8].

Nowadays, the OBD system has been developed even further and almost everyone uses OBD-II today. The differences in the upgraded system are within the area of connectivity, features, reliability and standardization [9].

The OBD-II port is standardized and only differs in voltage and in the middle, which is made of plastic. The visual difference can be seen between cars (OBD-II A) and transport vehicles (OBD-II B)(Figure 2.3, Figure 2.4).



**Figure 2.3:** OBD-II Connector "Type A" (Cars & Light Duty).



**Figure 2.4:** OBD-II Connector "Type B" (Medium & Heavy Duty).

The OBD-II frame is packaged as a CAN frame (Figure 2.5), with all the necessary information about the errors that have occurred, which is stored in the vehicle's memory [8].

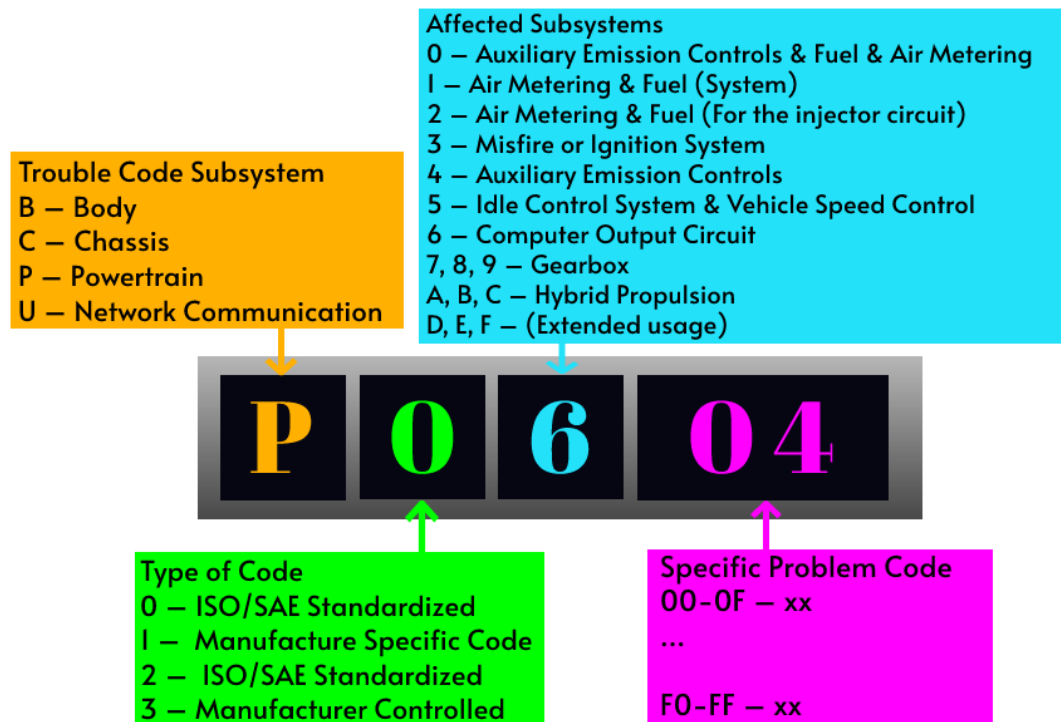


**Figure 2.5:** Shows the OBD-II frame packaging.

### 2.2.1 OBD-II DTCs

The project focuses on a statistical perspective within DTCs (Diagnostic Trouble Codes). DTCs overall, are codes that give derivation to the vehicle's various diagnostic errors. It is easy to confuse DTCs with a MIL (Malfunction Indicator Lamp) from the driver's panel, but this is not the case. DTC is a way to orient around what is wrong with the vehicle and where the fault is located [10].

If a fault is detected in the vehicle, a hexadecimal DTC from the OBD will be generated. When the fault has been detected by the OBD, it will perform two tasks. The first task is diagnosing the fault by code, the second one is highlighting the fault to the technicians by, for example, lighting a control lamp displaying the problem [11].



**Figure 2.6:** DTC Frame Package.

#### 2.2.1.1 Trouble Code Subsystem

The first character of the DTC consists of four different letters B, C, P and U [12].

- **Body** – Displays the parts you can find in the passenger compartment area of the vehicle.
- **Chassis** – Displays parts that you often can find outside the passenger compartment such as braking, steering and suspension.

- Powertrain – Displays parts that you can find associated with the vehicle's fuel system, transmission, and engine, also displays these three main objects.
- Network – Shows everything in vehicle within the network integration and related systems. These are functions are handled by the OBD-II.

### 2.2.1.2 Type of Code

In the industrial world, within the ISO/SAE-controlled DTC standard, they have succeeded in developing a norm that fits most of the companies. This by looking at the similarities that arise from manufacturers' applications perspective and how they adapt themselves to their own DTCs. By doing this kind of research, a common ground has been found that suits most of the companies' area within fault codes.

Due to differences within different companies' basic systems, another option has also been made, which gives more freedom to the manufacturer. This option is called Manufacture Specific Code [13].

The second character is a number between 0 to 3 which displays the following:

- 0 – ISO/SAE Standardized. (Global or generic codes)
- 1 – Manufacture Specific Code. (Specific or enhanced codes)
- 2 – ISO/SAE Standardized. (Generic codes)
- 3 – Manufacturer Controlled.

### 2.2.1.3 Affected Subsystems

The third character uses numbers between 0 to F.

- 0 – Auxiliary Emission Controls & Fuel & Air Metering.
- 1 – Air Metering & Fuel (System).
- 2 – Air metering & Fuel (For the injector circuit).
- 3 – Misfire or Ignition System.
- 4 – Auxiliary Emission Controls.
- 5 – Idle Control System & Vehicle Speed Control.



- 6 – Computer Output Circuit.
- 7, 8, 9 – Gearbox (Transmission).
- A, B, C – Hybrid Propulsion.
- D, E, F – (Extended usage)

#### **2.2.1.4 Specific Problem Code**

The 4th and 5th characters are merged as one and have a value between 00 to FF.

The mission of these two characters is to refer to the problematic area of the sub-system. This displays a thoroughly level which shows the exact problem within the area of the vehicle [12].

#### **2.2.1.5 FTB/FMI**

The FTB/FMI (Failure Type Byte/Failure Mode Identifier) is not included in the DTC format but expressed when someone talks about the ID (Identification) of the DTC.

Its task is to show the triggering link-fault of the components or electronic circuit in the vehicle, this by providing more information about the detection area. In other words, the FTB is not used as a root cause finder, but instead, a good indicator for the beginning of the troubleshooting in the area, since there can be an array of different root causes [14].

## **2.3 Python**

Python is a high-level easy to use programming language built on an open source, it was created by Guido Van Rossum in 1991 and is named after “The Monty Python’s Flying Circus”. The purpose is to have a programming language that is easy to understand, write and just as powerful as the leading competitors and suitable for everyday tasks [15].

### **2.3.1 Short Description of Code Progress**

The programming part of the project started off by creating a minor program, with the only task of removing all fault codes but one. The program was extended and was soon able to remove columns and end with printing out the result down into an Excel file.

The sorting continued, and the first iteration grew to sort out all vehicles with a registered fault code over the period 2019-2022. This was exported to an Excel file

and later became the first normal distributed graph.

After this, sorting continued by season, year and countries. Removal of data that wasn't necessary for the analysis, also export of data to an Excel file, for visualisation and for the statistical analysis. The code grew larger, more information was gathered and implemented with time.

All source code developed for this project could be found in the appendix (A) section.

### 2.3.2 Pandas

Pandas is a software library that is built on subroutines that has usage within data analysis and also used as a manipulation tool. It's built-on top of the programming language Python [16]. Pandas' goal is to become the world's best software library in data analysis and manipulation tool in the world.

Pandas was developed in 2008 and became an open source in 2009. Since then, it has enlarged with time and even included editions of Python for Data Analysis. Pandas have also become sponsored by NumFOCUS [17], which is a company that helps solve challenging problems that is occurring.

### 2.3.3 Sorting with Jupyter Notebook

The two thesis workers did their database sorting with Jupyter Notebook. It is a web-based programming interface which connects to many different kernels, by doing this it can handle over 40 programming languages [18]. Pandas is as mentioned a library that connects to Jupyter and facilitates the database manipulation.

## 2.4 Transmitter & Workshop Tools

The vehicles' error codes are stored in Volvo Group's database in two major ways, by cable or via direct connectivity. The most common in the past and the widely most used way in the company's history, is via the workshop tools, but with the workshop tools comes problems. The largest issue with this tool is that it gives incorrect timestamps.

By using the physical interaction method with the vehicle, the performed troubleshooting with the collection of data can be affected. Possible reasons could be that the quantity of error codes may be due to the transmission tool having hung up, the data bus has become too full or technicians pulling out the workshop tool at the wrong time.

Because of this dilemma, the two thesis workers chose to work with the transmitter data only. The transmitted information about the vehicles is more reliable, since it is auto generated and directly transmitted - the data is more solid. Nowadays, Volvo

Group uses the transmitted collection of data more frequently than the workshop tools.

### **2.4.1 The Data**

The data taken from the vehicles, goes through various process modules from the time it is being detected, until it is received into the database.

It starts with the sensors being triggered, then the disturbed data is sent through the vehicle system and saved until its arrival at the workshop, alternatively directly sent via the transmitter to Volvo GTT's database. Whether the data is sent via the transmitter or the workshop tools depends entirely on what the customer has agreed to in the agreement, when purchasing the vehicle.

Volvo Group get readouts every day. The data, that enters Volvo GTT's database, gets analysed and sorted in various ways. This is done to facilitate diagnostics, which makes the data more readable and the visualization clearer.

By organising the data and putting it in its right structure, gives Volvo Group a better insight which saves both time and money. This type of data organization, in turn, provides the company with more resources for other things like product improvements.

When the vehicle is requested a readout, the request needs to be done correctly for the readout to work. The global organization VGCS (Volvo Group Connected Services) must have the right pattern, which suits both data and vehicle type. The patterns are then loaded into the Volvo Group's communication gateway and then into the vehicle. Depending on how the requester want to see a specific data area, there are different types of points of view when doing so. This refers to how patterns are formatted differently from each other, depending on what the purpose is when the information is taken out of the vehicle.

### **2.4.2 Transmitter software**

All newly produced vehicles have a transmitter installed. The transmitter communicates with all nodes (ECUs) in the vehicle. These ECUs contains a software which includes special functions and process information and the logging data from the vehicle.

For the CAN signal database to work, the transmission between the nodes and transmitter needs to be confirmed by transmission and receiving – messages. This is important since the transmitter software requires this in order to work. The transmitter's software must also be developed and updated according to the type of vehicle, and electronic topology since it differences too. For this to work properly, without errors, Volvo GTT has a team that handles and maintains the transmitter software so that the runtime is compatible with the various Volvo Group brands.

For every transmitter software version, there is two things that must be included. Firstly, an updated node template that includes all the vehicle's ECUs and secondly, a communication matrix for the transmitter to know which ECU to communicate with when being active.

### 2.5 General Volvo Group Information & eMob

Since the environmental issues has such a large impact on our future, the market is also increasingly heading towards more and more environmentally friendly vehicles, which is one of the reasons why Volvo Group has started its own team called eMob. This team works with everything from hybrids to electric vehicles and is a team that is getting bigger and bigger. There are currently only medium duty trucks and buses within the hybrid/electrical department.

Volvo Group's vision is to find effective ways to develop their electrification through AI and statistics gathered from the data they receive from the vehicles. By using the data, Volvo GTT can thus also understand what the customer wants and needs, regardless of whether they are aware of what they want or not. The data that is downloaded is the guide to an improved product, this by, among other things, seeing how the customer uses the product and how the life cycle is affected on various components, depending on where the sold vehicle in the world is located at.

Things that Volvo Group strives for, includes that the products are 100%- safe, fossil-free and productive. The products that are released are also carefully inspected for quality defects via advanced analysis systems and therefore predictive maintenance of the product is obtained as well as prescriptive maintenance that can be found in Volvo GTT's production mode. In this way, earlier detection of potential errors can take place and therefore quality issues can be handled more efficiently and faster. Through algorithms, Volvo Group can predict when the different parts of the vehicle need to be maintained.

#### 2.5.1 Batteries

The batteries used in the medium duty Volvo truck and buses are supplied by a major European battery manufacturer. The batteries have a long cycle of life, which makes them a perfect fit for vehicles and machinery. The life cycle is divided into three parts, first life, second life and recycling or re manufacturing. Using this approach the battery manufacturer gives the batteries a total life cycle of approximately 20 years. All this is in accordance with the European battery directive Article 2 (Directive 2006/66/EC) stating that:

*“1. This Directive shall apply to all types of batteries and accumulators, regardless of their shape, volume, weight, material composition or use. It shall apply without prejudice to Directives 2000/53/EC and 2002/96/EC.” [19]*

### 2.5.2 Combustion Efficiency & Environment

Due to the environmental pressures of vehicles around the world, the EU is investing in becoming fossil-free by 2050 and cutting emissions by half by 2030 [20].

Even though electric vehicles are green, there is an underlying dilemma with the idea. Calculations about the emissions when manufacturing batteries and the electric operation of producing electricity for them are often not taken as a consideration by the ordinary person around the world.

In 2017, it was measured that fossil fuels make up as much as 64.5% of today's electricity [21].

Coal plants have mostly been running the electricity supply through out our history [22] and now days have an efficiency between 32-42%. The efficiency of natural gas is around 45% and oil-fired power 38% [23], while nuclear power plants have an efficiency of 35% [24].

When it comes to petrol-powered vehicles, they have an efficiency of 30%, while diesel engines have an efficiency of 40-45% [25].

In conclusion, it is important to think about how we turn to emissions as we are still far from our global goal. While some countries can afford to develop more environmentally friendly technologies, there are countries that cannot because of their limited economic resources. Therefore, it is important for us who have the resources to supply other countries with renewable technology ideas, to save future generations.

## 2.6 Important relations

The Central limit theorem states: Assume that  $\xi_1, \dots, \xi_n$  is  $n$  pieces of independent and equally distributed stochastic variables with the expected value  $\mu$  and the standard deviation  $\sigma$  then it applies as

$$\lim_{n \rightarrow \infty} P \left( \frac{\sum_{i=1}^n \xi_i - n\mu}{\sigma\sqrt{n}} \leq x \right) = \phi(x) \quad (2.1)$$

where  $\xi$  in this case is the total amount of faults,  $n$  is the total amount of vehicles  $\mu$  is the mean and  $\sigma$  is the standard deviation. This equating to  $\phi(x)$  being the sought after probability.

For our use, let's call the left fraction  $q$  and that corresponds with the sought probability value for  $\phi(x)$ . This is determined by the certainty you wish to get from your confidence interval. To get  $n$  vehicles with a ninety-five percent confidence one has to look at the normal distribution table seen in the appendix A, for 0.9505 this

will give  $q$  the value of 1.65

$$\phi\left(\underbrace{\frac{\xi - \mu n}{\sigma \sqrt{n}}}_q\right) = \phi(x) \quad (2.2)$$

with  $q$  being the t-distribution table A value for our confidence interval, it differs with what kind of confidence level you choose to use.

This further developed and rewritten for our purpose, we look for the amount of vehicles that produce  $\xi$  amount of faults based on  $\mu$ ,  $\sigma$  and  $q$

$$n_{1,2} = \frac{2\xi\mu + \sigma^2 q^2 \pm \sigma q \sqrt{4\xi\mu + \sigma^2 q^2}}{2\mu^2} \quad (2.3)$$

To be able to use the relations previously mentioned we need to use the standard deviation which states

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n - 1}} \quad (2.4)$$

where  $s$  is an estimation of  $\sigma$  which is the standard deviation,  $\sum$  is the sum of the calculation inside the parenthesis,  $x_i$  is the sample value,  $\bar{x}$  being the mean value of all samples and  $n$  is the sample size.

Gaussian normal distribution is used to plot our distribution based on the data we collected from our python program and states as follows

$$P(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.5)$$

Poisson Distribution is characterised by studying the probability of an event happening in a given interval, the events are occurring with a known mean and also independently from the last occurrence. The distribution is defined as

$$P(\xi = k) = \frac{e^{-\lambda} \lambda^k}{k!} \quad (2.6)$$

where  $\lambda$  is the mean in the interval previously mentioned as  $\bar{x}$  and  $k$  is the number of events.

Interval estimation is useful when you want to compare if there are any differences between two populations of different size, with a certain trait. The interval estimation states

$$\mu_1 - \mu_2 \pm Z_{\alpha/2} \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}, \quad \begin{matrix} n_1 \geq 100 \\ n_2 \geq 100 \end{matrix} \quad (2.7)$$

where  $\mu_1$  and  $\mu_2$  is the average fault in both populations,  $Z_{\alpha/2}$  is the degree of confidence you like to apply,  $s_1^2$  and  $s_2^2$  is the estimated standard deviation for the

populations and lastly  $n_1$  and  $n_2$  is the individual size of the population.

The Hypergeometrical distribution is a discrete probability distribution where you have a set amount of favourable outcomes from a limited pool of possibilities

$$P(X = k) = \frac{\binom{m}{k} \binom{N-m}{n-k}}{\binom{N}{n}} \quad (2.8)$$

where  $P$  is the probability for  $X$  amount of successful draws,  $N$  is the population size,  $m$  is the success state in the population,  $n$  is the amount of draws from the pool and  $k$  is the number of observed successes.





# 3

## Case Set-up

This chapter will describe the task given from Volvo GTT. The starting point was that the thesis workers should do a fault trace analysis looking into a certain fault code, another viable way forward was the question of how many logging devices should you fit to a selection of vehicles to cover the entire population. The tasks given was quite open and was subject to change and evolve over time.

Since the aim was to investigate the different fault codes within the electrical and hybrid trucks and busses, the focus was towards the internal system that composes the handling of fault codes within a vehicle. This provided valuable knowledge on how the ECU registers error codes and how the CAN sends them

With large databases an efficient way to handle them needed to be applied, with over hundreds of thousand entries' it needed to be handled with computing and coding. Volvo GTT uses the programming language Python with Pandas extension library; Pandas is great for data manipulation.

Using Python and the Pandas extension library to manipulate data. Step by step code was written, learning how to manipulate the database and removing obsolete data. To see the result of the code a trimmed down data frame was exported to an Excel spreadsheet, this gives an opportunity to see what data has been removed and what was left.

The decision was made to investigate component failure codes to see if there was any correlation between component failure codes and overheating. Singling out a set of twenty-five vehicles with component failures paired with overheating issues in different fault ranges. No obvious connection could be found to confirm this hypothesis. The very few cases that might suggest any correlation between components and overheating was circumstantial at best.

Not getting any workable results that could be presented, this prompted a change in direction. The data could be sorted to be more refined, and then used to make statistical assumptions. Using Normal distribution, Central limit theorem, Poisson distribution and confidence interval this could be achieved.

## 3.1 Implementation

In this section, the implementation of the case setup will be described.

### 3.1.1 Data Collection

The data used in this project, was collected from buses and the medium duty truck pool, spanning from late 2019 to spring 2022. This pool included both hybrid vehicles and full electric. The first iteration was presented in Excel format, later this changed to the more compressed Parquet format. The file was transferred from a CSV-file into a Parquet from the database, which was stored for analysis.

### 3.1.2 Python sorting

Volvo Group uses Python programming language because the library Pandas makes it easier to sort the database. This is an excellent combination and to make the data manipulation more straight forward. The cleaned data was then exported to an Excel sheet based on what data could be useful to analyse.

### 3.1.3 Statistical Methods

The statistical methods used were, Normal distribution, Poisson distribution, Central limit theorem and confidence interval. The key to some of these was to first calculate the normal distribution to get a curve of how the errors were distributed, to get this curve you first had to calculate the mean and the standard deviation of the distributed errors. These could later also be applied to, for example, the Poisson distribution and the central limit theorem.

By using the central limit theorem, we can estimate the amount of fault codes received based on the mean and standard deviation within the population. We need to select an amount of vehicles for this equation to apply. This works when we know the amount of vehicles, but if we want to reverse it and estimate the amount of vehicles which produce a desirable amount of fault codes. You then need to rewrite the equation and make the amount of faults known and the vehicles unknown.

### 3.1.4 Excel

The refined data generated from the code was exported to an Excel file. Excel is a useful tool when visualising the data and handling large data tables, it conveniently is used to create graphs and calculations of the data in the tables.

### 3.1.5 Ethics

The thesis workers addressed the ethic aspect of the data handling and company conduct in their work. This was done by following Volvo Group's Code of conduct, which include Volvo Group's ethics regarding the confidential information, data privacy and personal conduct.

# 4

## Requirements

There are several laws and regulations that is forced upon the vehicle industry, either by their own country or the EU. In this chapter we will address the key laws that Volvo GTT must comply with, along ethics and sustainability considerations.

If you look at the private data usage based on ethics, there are both positive and negative parts. The use of data has led to many benefits for Volvo Group, like predictive maintenance and future improvements based on the number of diagnostic trouble codes transmitted from the fleet. However, it could lead to some negative consequences if the data were to be compromised. To prevent any form of data compromise to occur, Volvo Group enforces the current legislation's regarding data management.

### 4.1 IP

When talking about IP, one refers to the creator of the idea. By patenting a work, where the IP is protected by law, a security is granted for the creators. This can be found in within names, images, designs and symbols used in trade [26].

The first patent that Volvo Group created was in 1949. Since then, more and more patents have been issued, and in 2016 this was measured up to 9,000 patents and 3,000 design rights. About 200 new patents are patented every year and about 45 employees work with IP for Volvo Group, around the world [27].

### 4.2 GDPR

Volvo GTT works to ensure that all its customers can feel safe and secure when handling of private data is used within Volvo Group. Volvo GTT has implemented a policy, where every employee is encouraged to learn about how to use private data through courses, according to GDPR and IP, which was legislated in the EU in 2016 [28].

The private data collected from the vehicles will not go through any pursuit of handling, after various development tasks have been accomplished by Volvo Group [29].

### 4.3 Battery Directive EU

Since most batteries contain environmentally hazardous substances such as mercury lead and cadmium, it is important for the general public to make sure to sort at source [30].

This is not just about ethics for the common people, the EU has in fact laid down legislation and directives for how producers should proceed the technology of batteries. This to reduce the fossil footprint from different areas of the EU [31]. Some important names of the most critical document can be found via the names:

- Batteries Directive: DIRECTIVE 2006/66/EC
- Reach Regulation: REGULATION (EC) No 1907/2006
- Ecodesign Directive: DIRECTIVE 2009/125/EC
- The list of Critical Raw Materials (2017): COM(2017) 490 final
- Strategic Action Plan on Batteries: COM(2018) 293 final – Annex 2
- New Batteries Regulation: EC Proposal for a Regulation on batteries and waste batteries (Dec 2020)

Proposal to have an electrical register and carbon dioxide declaration for the manufacture of batteries, where information on, among other things, where the battery was created, the producer, and the total carbon dioxide emissions for the producer company, can be found in the directive “Batteries Directive: DIRECTIVE 2006/66/EC“ [32].

### 4.4 Road safety

One of the Volvo Group’s most important values is safety and has been so for over 90 years [33]. Volvo GTT’s vision is that there will be zero accidents and has gone so far as to even name the vision "Zero". Although within a long distance to cover all accidents, Volvo Group has been able to significantly reduce accidents through innovative and high-quality products [34].

In a safety report released in 2017, from Volvo Trucks, predicts that they have analysed as many as 1,700 accidents that stretch all the way back to 1969. The work around minimizing accidents is investigated by a Volvo Group called ART, and the effort off reducing accidents is constantly being improved, this by absorbing statistics presented from the EU, and by the knowledge of truck accidents from all the way back in 1969. EU itself aims to reduce the number of traffic deaths by 50% between 2010-2020.

Since the human factor accounts for about 90% of traffic accidents, and because Volvo Group works in all aspects of safety [35], the company therefore started a child safety campaign called "STOP. Look. Wave.". Through a playful education, children are taught how to approach a truck. This by stopping, looking around for trucks that indicates danger, and confirm by waving to the truck driver so you know that he has seen you [36].

## 4.5 Emission regulations

In order to overcome the worldwide greenhouse effect, International Climate Negotiations (UNFCCC) were initiated through the UN in 1992. The aim was to stabilize the relationship with the greenhouse effect, which related to the quote "*dangerous human interference with the climate system*" [37].

Over time, more international agreements surfaced from different countries such as the Kyoto Protocol which was launched in the city of Kyoto in 1997, Japan and is an international agreement which most countries signed. The goal of the agreement was to jointly reduce emissions by at least 5.2%, between 1990 and 2012 [38].

In 2009, United Nations Climate Change Conference was held in Copenhagen, Denmark, where 192 countries participated. The aim of the conference was for the participants to together put up a plan, that could produce measurements that would possibly slow down the global warming [39].

Finally, the Paris Agreement emerged in 2015. The goal of this agreement was to work together, internationally, to strive to stay below 2 degrees Celsius until 2100. Furthermore, it has made even more efforts to counteract global warming by instead targeting 1.5 degrees in order to limit the environmental impact [40]. More about how Volvo Group adheres to this can be found in section 2.5.

Since lorries and buses account for about 6% of Europe's carbon dioxide emissions and about a quarter of vehicle transport emissions, given that, combustion engines have become more efficient, emissions still increase. This is happening because more and more transport vehicles are being produced and put into duty.

Due to the large influx from these large transport vehicles, the first ordinance on heavy-duty came into force in 14 August 2019, called (EU) 2019/1242. The aim of this regulation is for manufacturers to reduce emissions from their transport vehicles by 15% from 2025, and by 30% from 2030[41].



# 5

## Results & Discussion

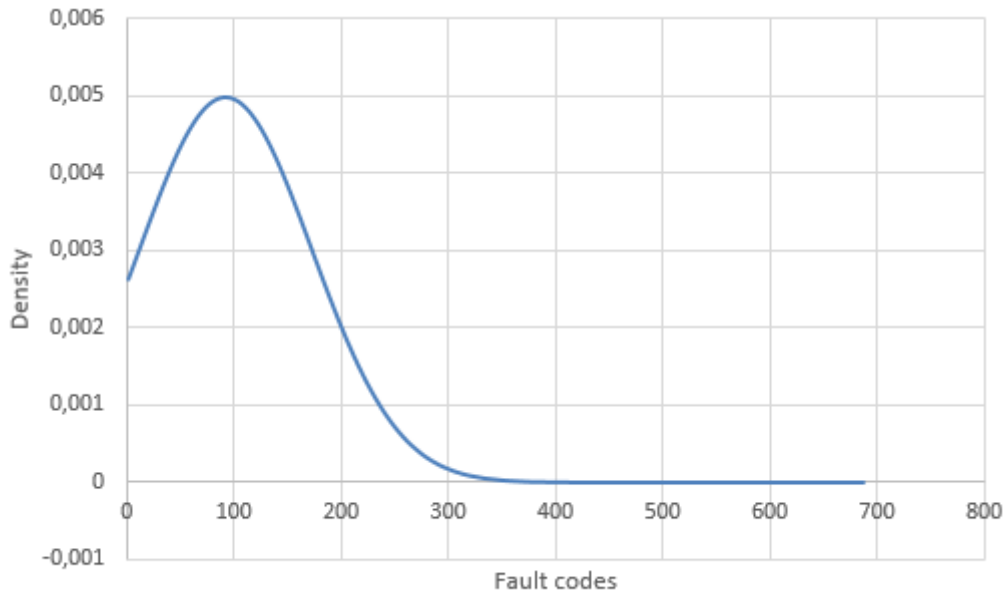
In this chapter the results of the sorted data and the statistics calculated will be presented and discussed. During the project a large amount of time was spent trying to widen the scope and deciding what the scope should be.

The purpose of the thesis project was to see if there is a possibility to mount high frequency transmitter on a selection of vehicles to cover the entire population, also to take a closer look at general component faults connected to the batteries.

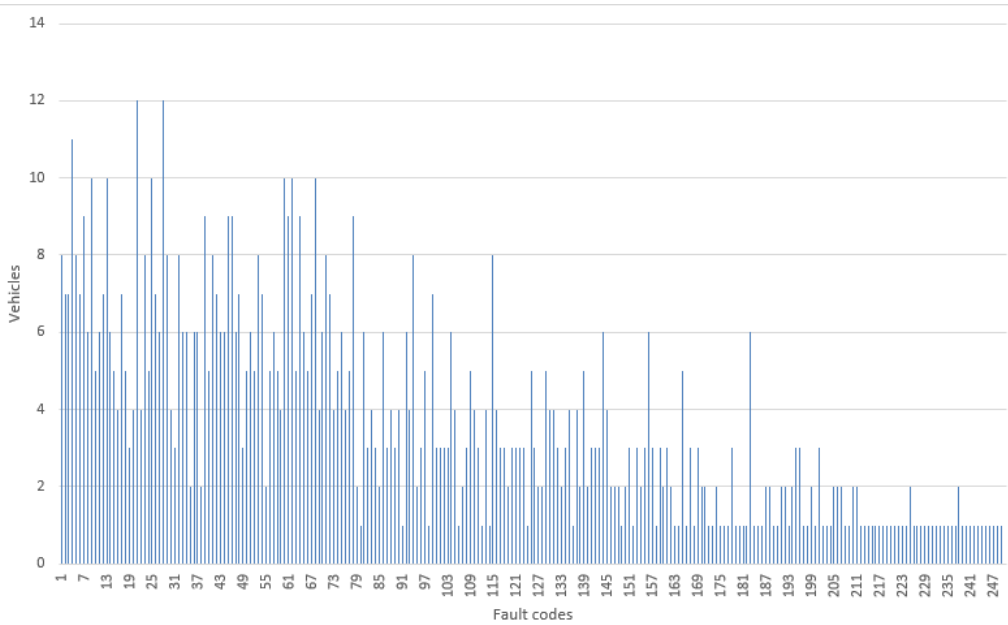
The first objective is easily answered in retrospect, when you know the fault distribution and which vehicle having certain faults or the lack of these. The vehicles you want to isolate are easily located, if you want to do this in advance, without knowing the distribution it will fall under Hypergeometric distribution, described in section 2.8. Where you have your pool of vehicles and randomly draw one out of the mass hoping for a certain vehicle with desired feature, this can be seen as more of a lottery when wanting or needing that special feature to analyze.

Depending on how many fault codes you want to receive for analysis, the question of how many vehicles in the pool you need to mount a high frequency transmitter on differs. There is not a simple answer to this question, but the central limit theorem will help to answer this depending on what kind of data is needed. Either it could be determined on a general basis including all fault codes, or be determined based on a specific fault code that being sought for analysis.

The task was started by cleaning and sorting the database extracting what was needed to create a normal distribution and an empirical distribution graph, this was exported to an Excel sheet where it was visualized.



**Figure 5.1:** Normal distribution for all vehicles from 2019 to 2022.



**Figure 5.2:** Empirical distribution of fault codes for the period 2019 to 2022.

Now that the data necessary was available, this enabled with the help of the central limit theorem to calculate how many errors would be received with  $n$  number of vehicles. We wanted to change this calculation so that we would input the desired number of fault codes instead, and the output would be how many vehicles were to render the desired amount of faults.

The central limit theorem formula was rewritten to suit our needs and yielded the following formula mentioned earlier in section 2.3. This calculation since being a



quadratic equation were to produce a false root. Conclusions were made that this said root were the one subtracting. To give an example how things could look, by using rough estimated numbers from the data on which the figure 5.1 is based, setting a value for  $\xi$ ,  $\mu$ ,  $\sigma$  and  $q$ , this in return gives a value for  $n$  when put in the equation. With  $\mu$  roughly 90, and  $\sigma$  lands on approximately 80. By using a 95% confidence interval this sets  $q$  to 1.65 according to the normal distribution  $\phi(z)$  table in the appendix (A). And let's say we wanted to receive 25 000 and 50 000 fault codes, this sets  $\xi$  to that amount, this gives us two calculations

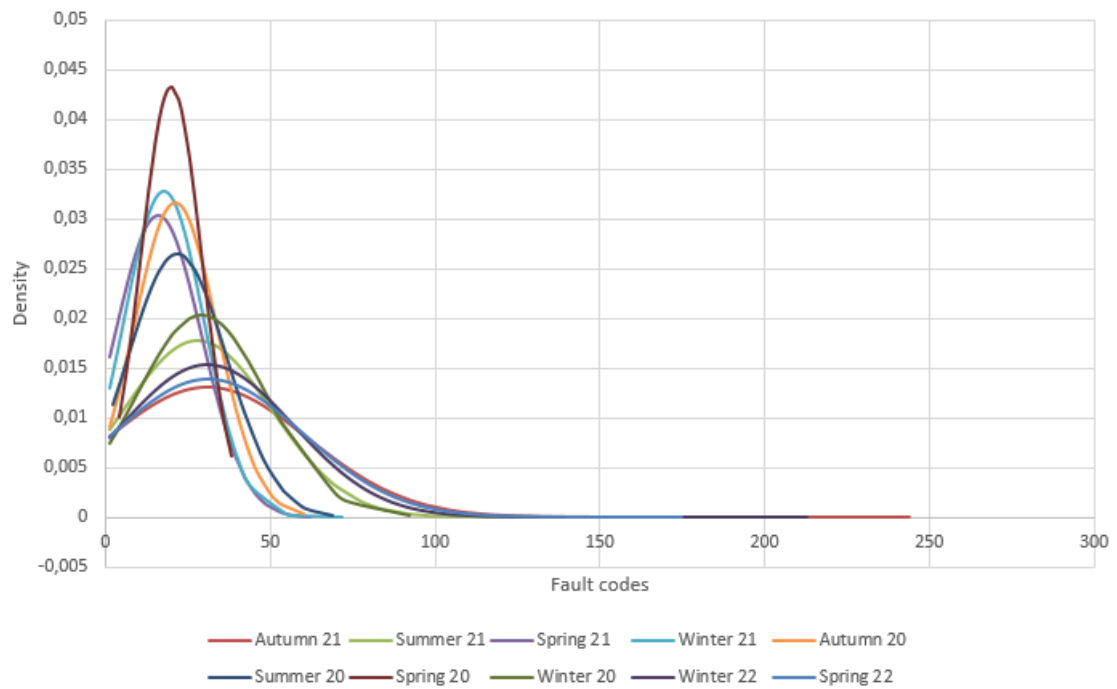
$$n = \frac{2\xi\mu + \sigma^2q^2 + \sigma q\sqrt{4\xi\mu + \sigma^2q^2}}{2\mu^2}$$

$$303 = \frac{2 \times 25000 \times 90 + 80^2 \times 1.65^2 + 80 \times 1.65\sqrt{4 \times 25000 \times 90 + 80^2 \times 1.65^2}}{2 \times 90^2}$$

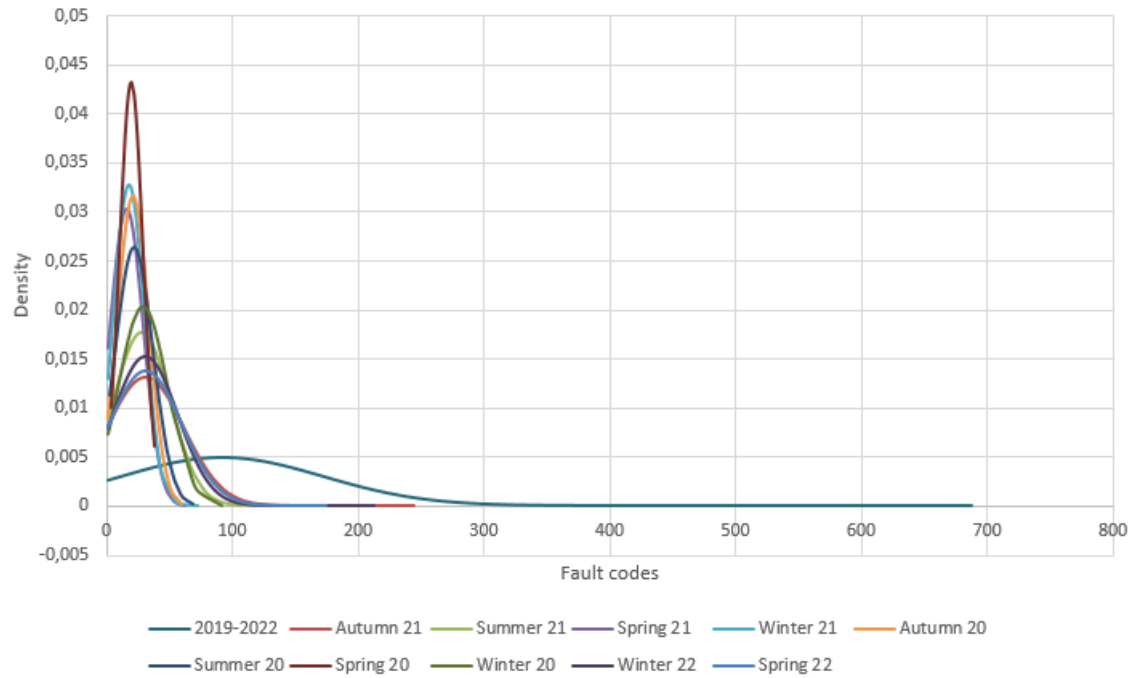
$$591 = \frac{2 \times 50000 \times 90 + 80^2 \times 1.65^2 + 80 \times 1.65\sqrt{4 \times 50000 \times 90 + 80^2 \times 1.65^2}}{2 \times 90^2}$$

in these example calculations, we want to receive faults codes with an approximate probability of 0.95. If we want to receive 25 000 fault codes then we need to collect fault codes from 303 vehicles, and if instead wanting to receive 50 000 fault codes then, the amount of vehicles needs to be 591.

To widen our understanding of how the distribution was we narrowed the time period to seasons, spanning from November 2019 to April 2022. As seen on the graph the spikes are higher but the width is also more compressed. This helped us seeing tendencies in fault distributions over the different seasons, how the fault density is distributed and also a rough mean for each period.



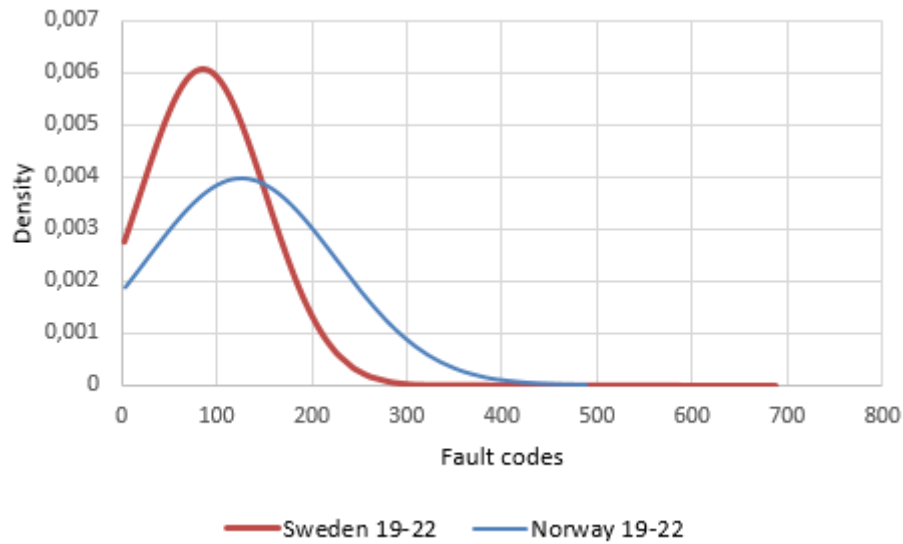
**Figure 5.3:** Normal distribution all seasons, and all seasons compared to the full period.



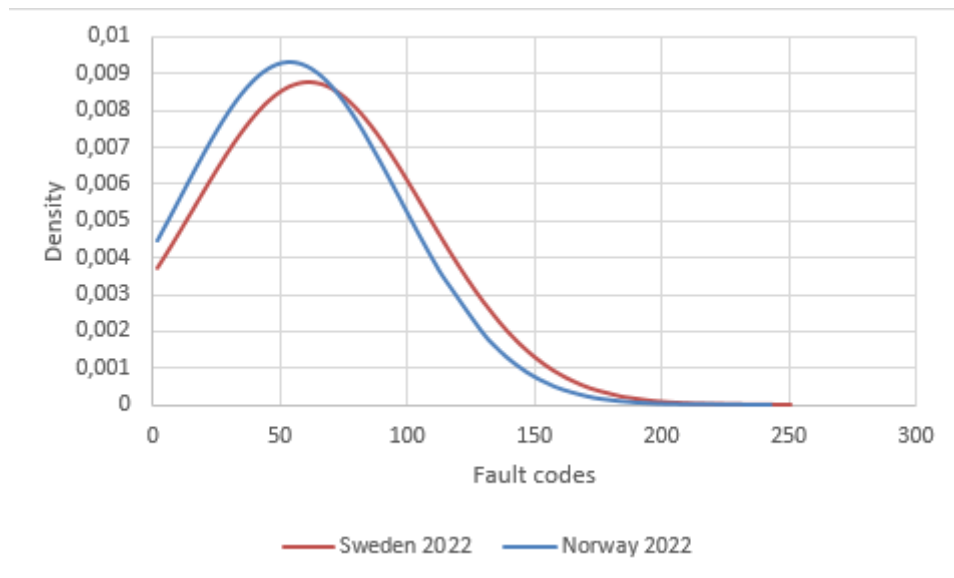
**Figure 5.4:** Normal distribution graphs for all seasons and the period 2019 to 2020.

This prompted us wanting to investigate this further, if there was any kind of difference between hemispheres. With this in mind, we altered the code to get the data required to succeed with this task. The aim was to investigate if there were

any apparent difference between countries on different continents and hemispheres. This unfortunately could not be done in the way we intended, being in the start of electrification for medium duty trucks, the pool of vehicles was too small. The focus shifted to Sweden and Norway instead, both countries had a sufficiently large vehicle fleet covering late 2019 to springtime 2022, to do a statistical analysis of.



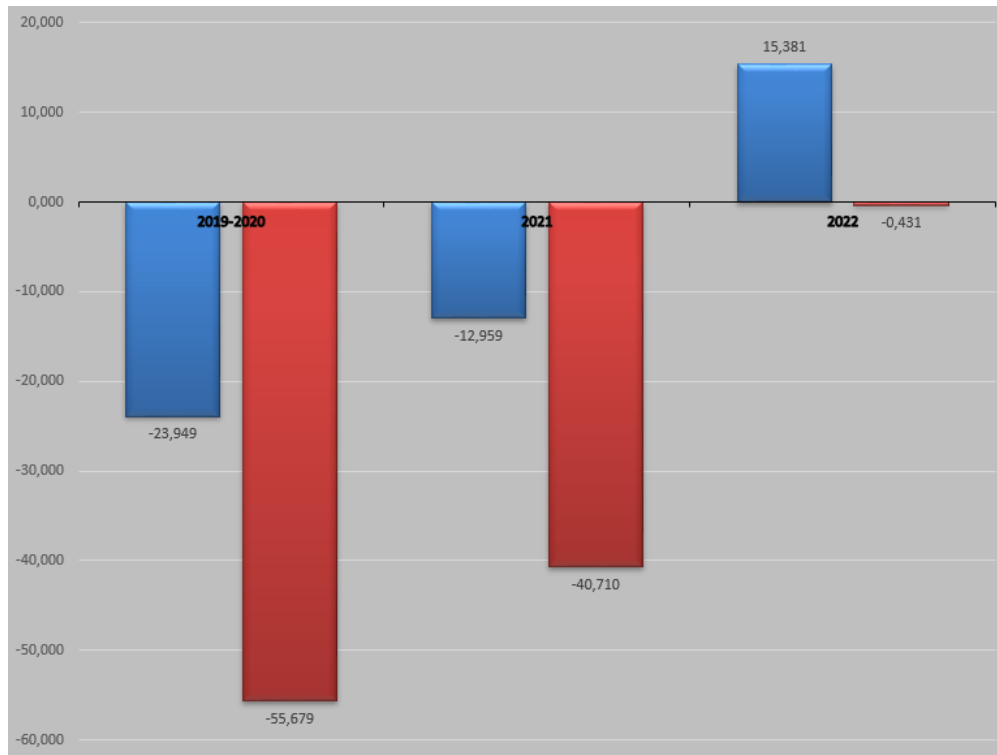
**Figure 5.5:** Shows the normal distribution for Swedish and Norwegian trucks during the period 2019 to 2022.



**Figure 5.6:** Showing the normal distribution graphs for Sweden and Norway the year 2022.

With interval estimation, we use another statistical method to see if there is any difference between Sweden and Norway. With the help of the standard deviation,

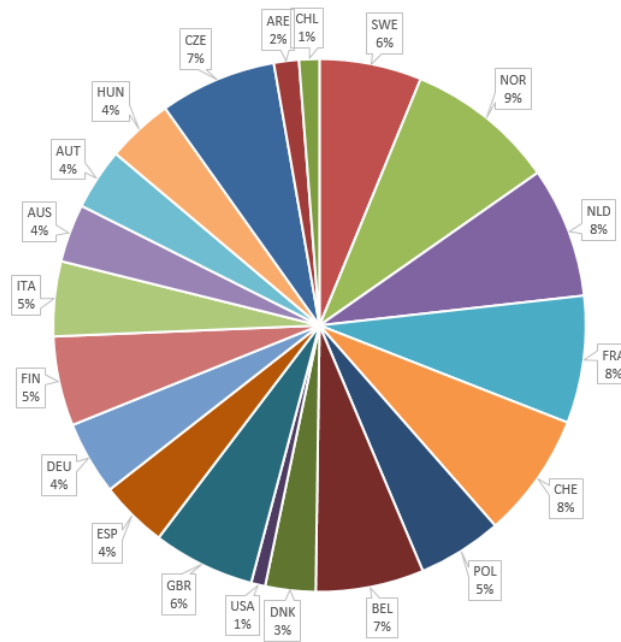
we can calculate if the both countries are within the interval or exceeded it in any way.



**Figure 5.7:** Confidence interval showing Sweden and Norway. As seen on the period 2019-2022 both of the bars are located in the sub zero range, this also applies for 2021. And in 2022 the bars have shifted and are now situated on both sides of the zero mark.

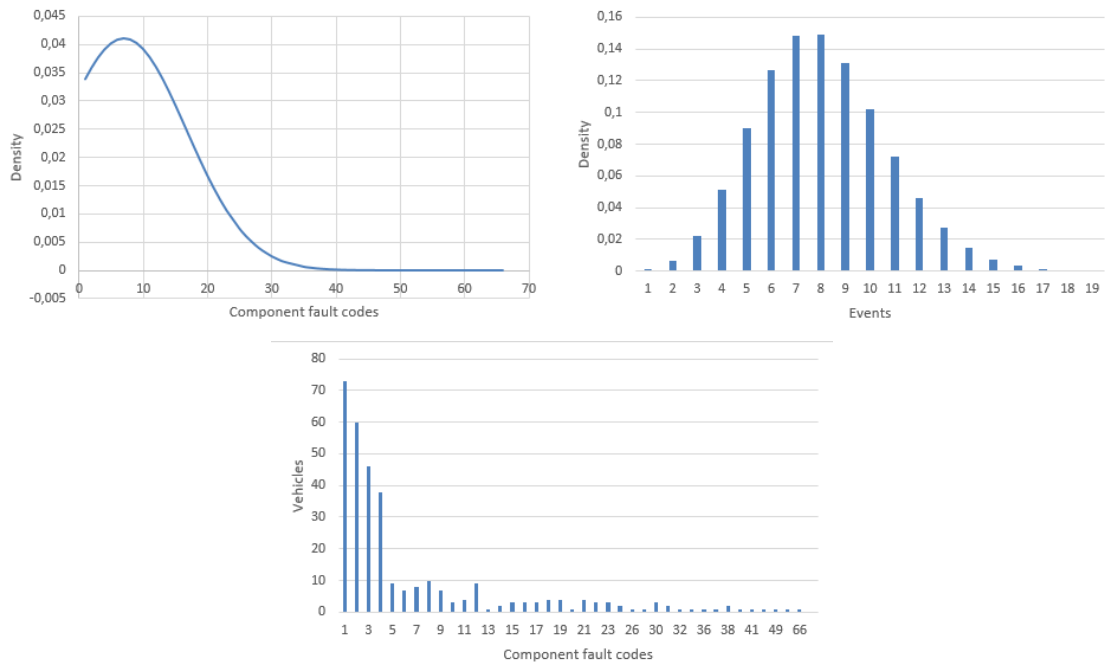
As seen in figure 5.7 the interval is quite skewed in one direction, for the whole period and also for 2021. In 2022 the interval is just between the zero, this means that the error distribution has leveled out and error distribution between the countries is starting to close up. This is also verified by figure 5.6 when you see how the normal distribution between Sweden and Norway year 2022 evens out and approaches each other compared to the full period.

Another interesting statistic was the fault code distribution between countries (figure 5.8), the distribution is fairly even between the countries. The countries with the highest number of vehicles also sits on a larger piece in the pie chart. The only anomaly here would be USA who have the least amount of reported fault codes per vehicle.



**Figure 5.8:** Pie chart with fault code distribution per vehicle and country.

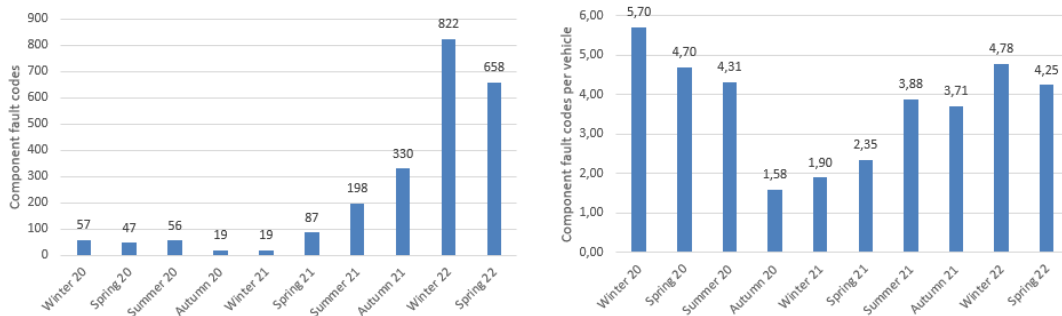
The next thing to be examined were the component fault codes, this to see if there were any anomalies with these fault codes. This time yet another approach was taken regarding the statistics. As with all previous statistical analysis we needed to get the mean value and the standard deviation, this time wanting to see how the component fault codes were distributed using the Poisson distribution.



**Figure 5.9:** Showing the Normal distribution, the Poisson distribution for the component fault codes and the empirical distribution of the fault codes.

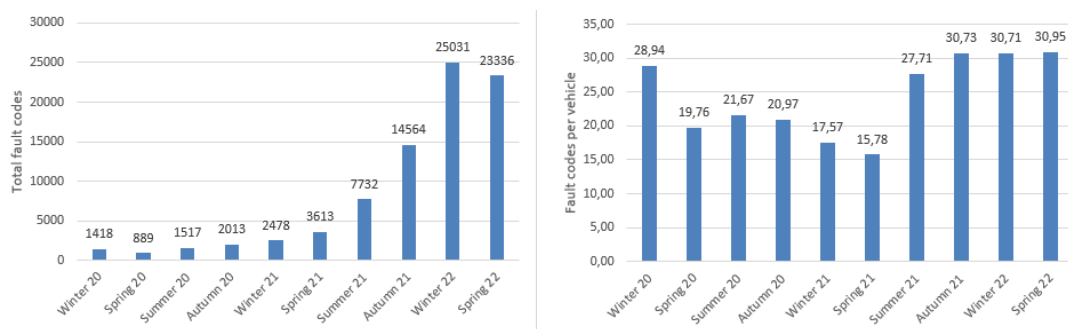
It is clearly visible that the normal distribution graph has a similar look to the previous distribution graphs. The Poisson takes the mean value from the fault code distribution, and after it plots a bar chart with the distribution of errors. This chart looks more like a classic normal distribution graph. The highest density is located around 7-8 events, which is similar to the normal distribution curve. But the empirical distribution of errors, shows that the Poisson distribution should not be used to estimate any distribution regarding component errors. And also this could be applied for general fault codes, since all the previous graphs have shown similar form.

In figure 5.10, we can see that the general component fault code increase significantly in the middle end of the interval, within the investigated period. Although the mean value of the component failures is rather even after spring 2021.



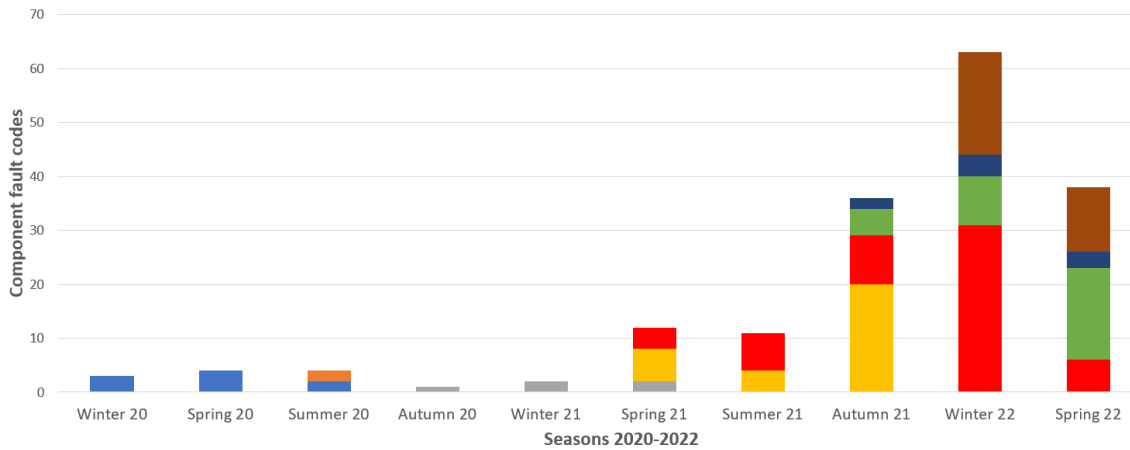
**Figure 5.10:** Showing the total amount of general component fault codes (FTB - 0x09) and the mean value of these, distributed over the seasons.

Because of these graphs combined, this means that there is an increase in vehicles, with these component fault codes.



**Figure 5.11:** Showing the total amount of faults within the fleet and the mean value of these, distributed over all seasons.

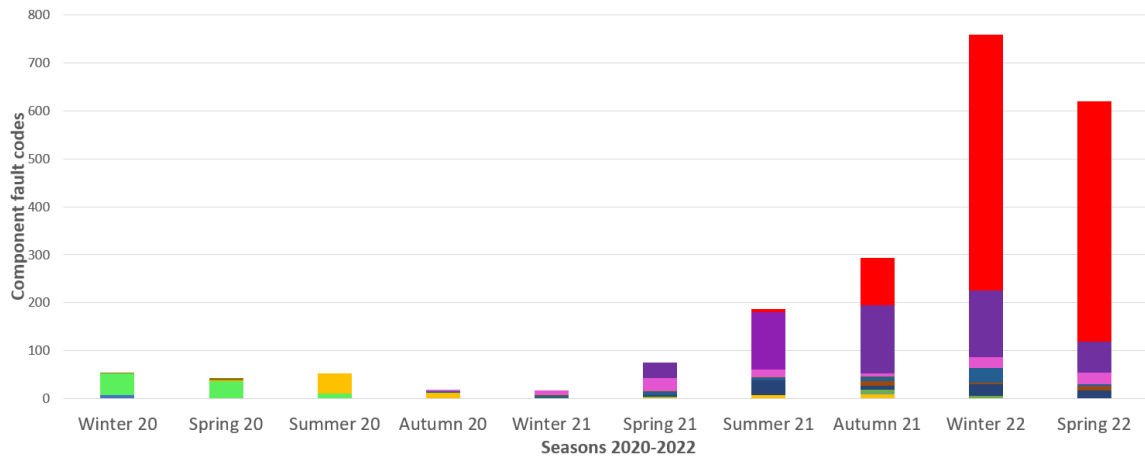
In figure 5.11, we see that the values of all fault codes within the fleet increase similarly to the component fault codes. It also shows that the mean value once again, correspondingly stays the same, similar to the component faults within this period.



**Figure 5.12:** SW for the total amount of general component failure for medium duty trucks (EV/Hybrid), distributed over the seasons.

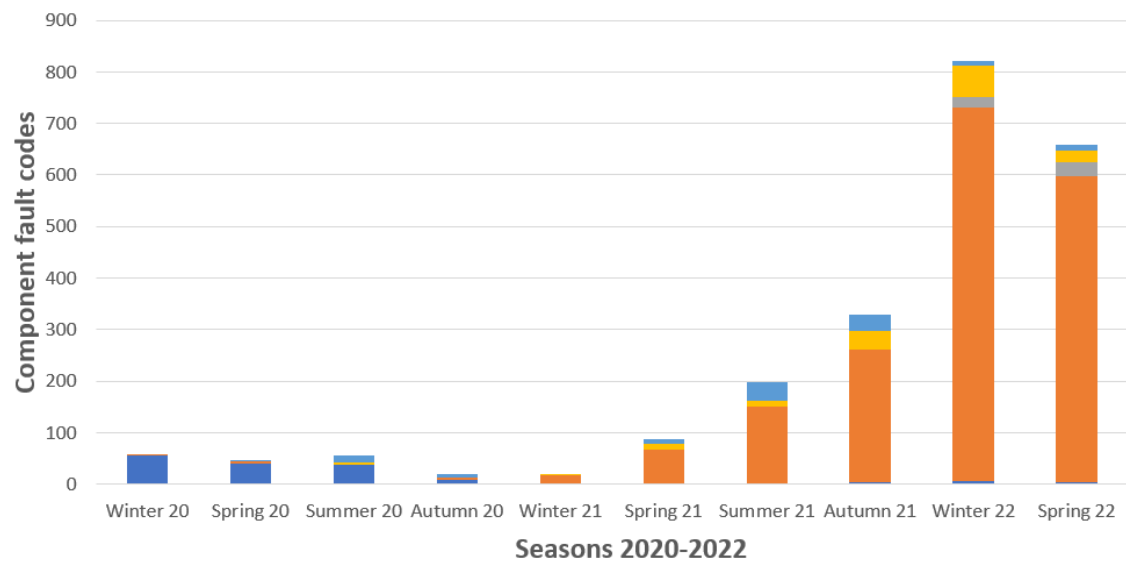
To further investigate we wanted to see what kind of software version was running at the time. This, in order to see if any software versions were standing out after summer 2021. In figures 5.12 and 5.13, we divided the trucks and buses separately, this to differentiate the different software versions.

In the trucks graph 5.12, we see different variations of the software within the bars of the interval, after summer 2021. This in turn does not show any major abnormality in the software versions, like the buses do.



**Figure 5.13:** SW for the total amount of general component failure for buses (EV/Hybrid), distributed over the seasons.

As we can see in the bus figure 5.13, the number of errors with the new software in red colour is increasing. This does not necessarily mean that there is something wrong with the software used, which could simply be due to many other causes. A probable reason for the high values at the end, is that the number of active vehicles on the street is increasing.



**Figure 5.14:** General component failure code (FTB - 0x09) from node source within trucks and buses, distributed over the seasons.

We wanted to go on an even deeper and see which node family these values originated from. In this case, the values of these bars seem to mostly come from the battery node family, which has an orange colour and can be seen in figure 5.14.



# 6

## Conclusion

In this thesis report a statistical analysis was carried out with the existing pool of medium duty trucks and busses, within Volvo Group eMob. With the scope selected one limitation was not having enough data for a full statistical analysis, meaning there weren't enough fault codes, since we are in the early era of electrified trucks and buses and the vehicle fleet is rather limited and reliability is high. This could pose a problem when you want to investigate certain fault codes, if there is not sufficient amount of data, one needs to investigate errors one by one, to see if there are any anomalies. This issue will likely not pose a problem in the future, since the vehicle fleet will increase in size.

Approaching diagnostic trouble code analysis from a statistical point of view facilitates statistical anomalies and tendencies in the data. With a growing vehicle fleet this will become possible to do. The statistics can then be used to find out if there are any differences between clusters of vehicles in different countries, on the northern or southern hemisphere, even regions within a country. With these statistical methods one can easily examine different faults codes using different data to get the tendencies of that certain fault code.

This can be seen in figure 5.7, when using interval estimation. We can see that there is a difference for the full period, where the difference in 2022 has decreased to be just within the interval. Looking at the normal distribution graphs in figure 5.5 and 5.6, its clearly noticeable that the graph for the full period has a larger offset than the graph for 2022.

This work can be seen as a compliment to the work that Volvo Group already is conducting in the field of diagnostic trouble code analysis. Since the pool of vehicles is quite limited, the statistical analysis could not be implemented to its fullest extent. But with time this could be a useful tool in finding tendencies and anomalies between clusters of vehicles within the fleet.

The number of fault codes shown in this report may not be representative for the fleet of vehicles since this is dependent on the fault code design. This means the fault code count in this report has no correlation to the severity of a fault or a fault code.



# Bibliography

- [1] Denso, “A history of Engine Management Systems according to DENSO” 2017. [Online]. Available: <https://www.denso-am.eu/news/march-2017-newsletter-a-history-of-engine-management-systems-according-to-denso> (accessed on: 2022-04-16).
- [2] Multi Analyzer, “Error-DTC Codes” n.d. [Online]. Available: <http://www.multianalyzer.com/en/error-dtc-codes/> (accessed on: 2022-04-16).
- [3] A. Singh and S. Mutreja, “Automotive electronic control unit (ecu) market“, Allied Market Research, Portland, OR, USA, A01934, 2022, [Online]. Available: <https://www.alliedmarketresearch.com/automotive-electronic-control-unit-ecu-market>, Accessed on: 2022-04-18.
- [4] G.M. Smith, “What Is CAN Bus (Controller Area Network) and How It Compares to Other Vehicle Bus Networks”, Dewesoft. [Online]. Feb. 19, 2021. Available: <https://dewesoft.com/daq/what-is-can-bus> (accessed on: 2022-04-18).
- [5] M. Falch, “UDS Explained - A Simple Intro (Unified Diagnostic Services)” CSS Electronics. [Online]. Apr, 2022. Available: <https://www.csselectronics.com/pages/uds-protocol-tutorial-unified-diagnostic-services> (accessed on: 2022-04-21).
- [6] N. Dung, “Overview of Unified Diagnostic Services Protocol” Github. [Online]. Dec. 17, 2021. Available: <https://nvdungx.github.io/unified-diagnostic-protocol-overview/> (accessed on: 2022-04-21).
- [7] M. Falch, “CAN Bus Explained - A Simple Intro [2022]” CSS Electronics. [Online]. Apr, 2022. Available: <https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial> (accessed on: 2022-04-19).
- [8] M. Falch, “OBD2 Explained - A Simple Intro [2022]” CSS Electronics. [Online]. Apr, 2022. Available: <https://www.csselectronics.com/pages/obd2-explained-simple-intro> (accessed on: 2022-04-19).

- [9] T. Miller, “OBD1 vs. OBD2: Definite interpretation and comparison” OBD Planet. [Online]. Apr. 23, 2019. Available: <https://obdplanet.com/obd1-vs-obd2/> (accessed on: 2022-04-19).
- [10] Samsara, “A Guide to Understanding DTC Codes” 2021. [Online]. Available: <https://www.samsara.com/guides/dtc-codes/> (accessed on: 2022-04-22).
- [11] R. Lovetere, “How to Read and Understand Check Engine Light (OBD-II) Codes” Your Mechanic. [Online]. Jun. 7, 2016. Available: <https://www.yourmechanic.com/article/how-to-read-and-understand-check-engine-light-codes-by-jason-unrau> (accessed on: 2022-04-22).
- [12] Motive, “Everything you need to know about DTC codes.” 2022. [Online]. Available: <https://gomotive.com/blog/dtc-codes/> (2022-04-23).
- [13] Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics – Part 6: Diagnostic trouble code definitions, Stockholm, Sweden: Svenska Institutet för Standarder, 2005. [Online]. Available: <https://www.sis.se/api/document/preview/906710/>, Accessed on: 2022-04-23.
- [14] Numeralkod, “Failure Mode Identifier (FMI) Codes on J1939 data link,” n.d. [Online]. Available: <https://lnx.numeralkod.com/wordpress/docs/errors-index/failure-mode-identifier-fmi-codes-on-j1939-data-link/> (accessed on: 2022-04-24).
- [15] Python Institute, “Python® – the language of today and tomorrow” n.d. [Online]. Available: <https://www.pythoninstitute.org/about-python> (accessed on: 2022-04-24).
- [16] Pandas, “pandas” n.d. [Online]. Available: <https://pandas.pydata.org/> (accessed on: 2022-04-25).
- [17] Pandas, “About pandas” n.d. [Online]. Available: <https://pandas.pydata.org/about/> (accessed on: 2022-04-25).
- [18] Jupyter, “Free software, open standards, and web services for interactive computing across all programming languages” n.d. [Online]. Available: <https://jupyter.org/> (2022-04-25).
- [19] DIRECTIVE 2006/66/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL (2006), OJ L 266/4. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:>

32006L0066&from=EN, Accessed on: 2022-04-27.

- [20] European Union, "Sverige får nyckelroll när Europa ska bli fossilfritt med hållbara batterier" n.d. [Online]. Available: [https://sweden.representation.ec.europa.eu/strategi-och-prioriteringar/goda-exempel/sverige-far-nyckelroll-nar-europa-ska-bli-fossilfritt-med-hallbara-batterier\\_sv](https://sweden.representation.ec.europa.eu/strategi-och-prioriteringar/goda-exempel/sverige-far-nyckelroll-nar-europa-ska-bli-fossilfritt-med-hallbara-batterier_sv) (accessed on: 2022-04-29).
- [21] World Nuclear, "Where does our electricity come from?" n.d. [Online]. Available: <https://world-nuclear.org/nuclear-essentials/where-does-our-electricity-come-from.aspx> (accessed on: 2022-04-29).
- [22] H. Ritchie and M. Roser, "Global fossil fuel consumption" Our World in Data. [Online]. Jan. 1, 2019. Available: <https://ourworldindata.org/fossil-fuels> (accessed on: 2022-04-30).
- [23] G. Zeiss, "Energy Efficiency of Fossil Fuel Power Generation" Geospatial.blogs. [Online]. Jan. 1, 2010. Available: <https://geospatial.blogs.com/geospatial/2010/01/energy-efficiency-of-fossil-fuel-power-generation.html> (accessed on: 2022-04-30).
- [24] Strål Sakerhets Myndigheten, "Forsmarks kärnkraftverk" 2017. [Online]. Available: <https://www.stralsakerhetsmyndigheten.se/omraden/karnkraft/karntekniska-anlaggningar-i-drift-i-sverige/forsmark/> (accessed on: 2022-04-30).
- [25] Jordbruksaktuellt, "Bensindiesel kan öka verkningsgraden" 2010. [Online]. Available: <https://www.ja.se/artikel/34647/bensindiesel-kan-oka-verkningsgraden.html> (accessed on: 2022-04-30).
- [26] Wipo, "What is Intellectual Property?" n.d. [Online]. Available: <https://www.wipo.int/about-ip/en/> (accessed on: 2022-05-10).
- [27] J. Bördin, "Volvo's road to businessdriven IP management" Konsert. [Online]. Oct. n.d, 2016. Available: <https://www.konsert.com/content/uploads/2019/01/iammagazineseptember2016-volvosroadtobusiness-drivenipmanagement.pdf> (accessed on: 2022-05-10).
- [28] Wikipedia, "Dataskyddsförordningen" 2022. [Online]. Available: <https://sv.wikipedia.org/wiki/Dataskyddsf%C3%B6rordningen> (accessed on: 2022-05-12).

- [29] Volvo Group, “Privacy” n.d. [Online]. Available: <https://www.volvogroup.com/en/tools/privacy.html> (accessed on: 2022-05-12).
- [30] The Battery Challenge, “BATTERIET – EN 200 ÅR GAMMAL UPPFINNING” n.d. [Online]. Available: <http://www.thebatterychallenge.se/fakta/> (accessed on: 2022-05-13).
- [31] Eba250, “EU LEGISLATION & DIRECTIVES” n.d. [Online]. Available: <https://www.eba250.com/legislation-market/eu-legislation/> (accessed on: 2022-05-13).
- [32] Deloitte, “New Battery Directive and implications for the automotive sector” n.d. [Online]. Available: <https://www2.deloitte.com/nl/nl/pages/legal/articles/new-battery-directive-implications-for-automotive-sector.html> (accessed on: 2022-05-13).
- [33] Volvo Group, “Volvokoncernens arbete med trafiksäkerhet” n.d. [Online]. Available: <https://www.volvogroup.com/se/about-us/traffic-safety.html> (accessed on: 2022-05-14).
- [34] Volvo Group, “Volvo Group Safety Vision” n.d. [Online]. Available: <https://www.volvogroup.com/content/dam/volvo/volvo-group/markets/global/en-en/about-us/traffic-safety/volvo-group-safety-vision-poster.pdf> (accessed on: 2022-05-14).
- [35] S. Kockum, R. Örtlund, A. Ekfjorden and P. Wells, “Volvo Trucks Safety Report 2017” Volvo Group, Gothenburg, Sweden, n.d, n.d, 2017. [Online]. Available: <https://www.volvogroup.com/content/dam/volvo/volvo-group/markets/global/en-en/about-us/traffic-safety/Safety-report-2017.pdf>, Accessed on: 2022-05-14.
- [36] Volvo Trucks, “STANNA. Titta. Vinka.” n.d. [Online]. Available: <https://www.volvotrucks.se/sv-se/about-us/safety/stop-look-wave.html> (accessed on: 2022-05-14).
- [37] Wikipedia\*, “United Nations Framework Convention on Climate Change” 2021. [Online]. Available: [https://sv.wikipedia.org/wiki/United\\_Nations\\_Framework\\_Convention\\_on\\_Climate\\_Change](https://sv.wikipedia.org/wiki/United_Nations_Framework_Convention_on_Climate_Change) (accessed on: 2022-05-15)
- [38] Wikipedia, “Kyotoprotokollet” 2021. [Online]. Available: <https://sv.wikipedia.org/wiki/Kyotoprotokollet> (accessed on: 2022-05-15).

- [39] Wikipedia, “Förenta nationernas klimatkonferens 2009” 2021. [Online]. Available: [https://sv.wikipedia.org/wiki/F%C3%B6renta\\_nationernas\\_klimatkonferens\\_2009](https://sv.wikipedia.org/wiki/F%C3%B6renta_nationernas_klimatkonferens_2009) (accessed on: 2022-05-15).
- [40] Wikipedia, “Parisavtalet (Förenta nationernas klimatavtal)” 2022. [Online]. Available: [https://sv.wikipedia.org/wiki/Parisavtalet\\_\(F%C3%B6renta\\_nationernas\\_klimatavtal\)](https://sv.wikipedia.org/wiki/Parisavtalet_(F%C3%B6renta_nationernas_klimatavtal)) (accessed on: 2022-05-15).
- [41] REGULATION (EU) 2019/1242 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL (2019), L 198/202. [Online]. Available: [https://ec.europa.eu/clima/eu-action/transport-emissions/road-transport-reducing-co2-emissions-vehicles/reducing-co2-emissions-heavy-duty-vehicles\\_en](https://ec.europa.eu/clima/eu-action/transport-emissions/road-transport-reducing-co2-emissions-vehicles/reducing-co2-emissions-heavy-duty-vehicles_en), Accessed on: 2022-05-15.
- [42] U. Blomqvist, Matematisk statistik: Ulla Dahlbom. Göteborg, Sweden, 2014.





# A

## Appendix 1

Collection of formulas, graphs and code

Central limit theorem:

$$\frac{\mu x - \xi}{\sigma \sqrt{x}} = q$$

$$\frac{yx - p}{z \sqrt{x}} = q$$

$$x^2y^2 - 2xyp + p^2 - xq^2z^2 = 0$$

$$x^2y^2 - 2xyp - xq^2z^2 + p^2 = 0$$

$$x^2 \underbrace{y^2}_{\text{a}} \underbrace{-2yp - q^2z^2}_{\text{b}} x + \underbrace{p^2}_{\text{c}} = 0$$

$$x^2a + bx + c = 0$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x_{1,2} = \frac{2py + z^2q^2 \pm zq\sqrt{4py + z^2q^2}}{2y^2}$$

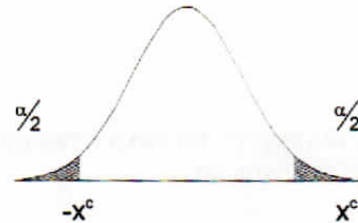
$$n_{1,2} = \frac{2\xi\mu + \sigma^2q^2 \pm \sigma q\sqrt{4\xi\mu + \sigma^2q^2}}{2\mu^2}$$

Normalfördelningens fördelningsfunktion  $\Phi(z)$  $\Phi(z) = P(Z \leq z)$  där  $Z$  är  $N(0, 1)$ För negativa värden, utnyttja att  $\Phi(-z) = 1 - \Phi(z)$ 

z	0	1	2	3	4	5	6	7	8	9
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
3.5	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998
3.6	0.9998	0.9998	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.7	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.8	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

## Tabell över t-fördelningen

kritiska värden,  $x^c$ , för olika frihetsgrader (df)  
och signifikansnivån  $\alpha$



df	$\alpha$	0.20	0.10	0.05	0.02	0.01	0.002	0.001
1		3.08	6.31	12.71	31.82	63.66	318.31	636.61
2		1.89	2.92	4.30	6.96	9.93	22.33	31.60
3		1.64	2.35	3.18	4.54	5.84	10.21	12.92
4		1.53	2.13	2.78	3.75	4.60	7.17	8.61
5		1.48	2.02	2.57	3.36	4.03	5.89	6.87
6		1.44	1.94	2.45	3.14	3.71	5.21	5.96
7		1.41	1.89	2.37	3.00	3.50	4.79	5.41
8		1.40	1.86	2.31	2.90	3.36	4.50	5.04
9		1.38	1.83	2.26	2.82	3.25	4.30	4.78
10		1.37	1.81	2.23	2.76	3.17	4.14	4.59
11		1.36	1.80	2.20	2.72	3.11	4.02	4.44
12		1.36	1.78	2.18	2.68	3.06	3.93	4.32
13		1.35	1.77	2.16	2.65	3.01	3.85	4.22
14		1.34	1.76	2.15	2.62	2.98	3.79	4.14
15		1.34	1.75	2.13	2.60	2.95	3.73	4.07
16		1.34	1.75	2.12	2.58	2.92	3.69	4.02
17		1.33	1.74	2.11	2.57	2.90	3.65	3.97
18		1.33	1.73	2.10	2.55	2.88	3.61	3.92
19		1.33	1.73	2.09	2.54	2.86	3.58	3.88
20		1.33	1.72	2.09	2.53	2.85	3.55	3.85
21		1.32	1.72	2.08	2.52	2.83	3.53	3.82
22		1.32	1.72	2.07	2.51	2.82	3.51	3.79
23		1.32	1.71	2.07	2.50	2.81	3.48	3.77
24		1.32	1.71	2.06	2.49	2.80	3.47	3.75
25		1.32	1.71	2.06	2.49	2.79	3.45	3.73
26		1.32	1.71	2.06	2.48	2.78	3.44	3.71
27		1.31	1.70	2.05	2.47	2.77	3.42	3.69
28		1.31	1.70	2.05	2.47	2.76	3.41	3.67
29		1.31	1.70	2.05	2.46	2.76	3.40	3.66
30		1.31	1.70	2.04	2.46	2.75	3.39	3.65
40		1.30	1.68	2.02	2.42	2.70	3.31	3.55
60		1.30	1.67	2.00	2.39	2.66	3.23	3.46
120		1.29	1.66	1.98	2.36	2.62	3.16	3.37
$\infty$		1.28	1.64	1.96	2.33	2.58	3.09	3.29

### Sorting Code

July 8, 2022

#### 1 Initiering

```
[1]: import os

import pandas as pd
import numpy as np

from datetime import timedelta
from datetime import datetime

import matplotlib.pyplot as plt
```

#### 2 Load dataframe

```
[2]: #df = pd.read_excel("DTCFaultCodes.xlsx")
df = pd.read_parquet("VehicleFaultCodes.parquet")
#Loading the database and place it in df
```

#### 3 Drop data not needed

Removing sender framework, and trucks not delivered

```
[4]: df.drop(df[df['Domain'] == 'U'].index, inplace = True)
#drop networking data
df.drop(df[df['Delivery_status'] == 'Not Delivered'].index, inplace = True)
#drop trucks that havent been delivered
df.drop_duplicates(['Vehicle_Id', 'DtcId', 'FirstFailureTime', 'FailureCount'],
    ↵inplace = True)
#dropping duplicates
```

#### 4 Remove autumn 2019

```
[9]: df = df.loc[~((df['ReadoutYear'] == 2019) & (df['Season'] == 'Autumn')),:]
      #removing autumn 2019

[10]: df = df.loc[~((df['ReadoutYear'] == 2019) & (df['Season'] == 'Summer')),:]
      #removing summer 2019
```

#### 5 Filter out error count and errors

Filters out total errors on the vehicle pool

```
[12]: # value_counts as dataframe
      df_veh_19_22 = df['Vehicle_Id'].value_counts().to_frame()

[13]: df_veh_19_22.columns.values[0] = 'Faults'
      df_veh_19_22 = df_veh_19_22.rename_axis('Vehicle_Id').reset_index()
      df_veh_19_22.index.name='Vehicle'
      df_veh_19_22 = df_veh_19_22[['Faults']]
      #count faults 2021-2022
      #counting faults per vehicle

[14]: df_veh_19_22.to_excel('Faults per vehicle 2019-2022.xlsx')
```

#### 6 Filter out seasons

```
[15]: #copy seasons
      #dfautumn19 = df.copy()
      dfwinter22 = dfspring22 = df.copy()
      dfwinter21 = dfspring21 = dfsummer21 = dfautumn21 = df.copy()
      dfwinter20 = dfspring20 = dfsummer20 = dfautumn20 = df.copy()

[16]: #remove wrong season
      dfspring22 = dfspring22[dfspring22['Season']=='Spring']
      dfwinter22 = dfwinter22[dfwinter22['Season']=='Winter']

      dfspring21 = dfspring21[dfspring21['Season']=='Spring']
      dfsummer21 = dfsummer21[dfsummer21['Season']=='Summer']
      dfautumn21 = dfautumn21[dfautumn21['Season']=='Autumn']
      dfwinter21 = dfwinter21[dfwinter21['Season']=='Winter']

      dfspring20 = dfspring20[dfspring20['Season']=='Spring']
      dfsummer20 = dfsummer20[dfsummer20['Season']=='Summer']
      dfautumn20 = dfautumn20[dfautumn20['Season']=='Autumn']
      dfwinter20 = dfwinter20[dfwinter20['Season']=='Winter']
```

```
[17]: dfspring22 = dfspring22[dfspring22['ReadoutYear']==2022]
      #keep year 2022
      dfspring21 = dfspring21[dfspring21['ReadoutYear']==2021]
      dfsummer21 = dfsummer21[dfsummer21['ReadoutYear']==2021]
      dfautumn21 = dfautumn21[dfautumn21['ReadoutYear']==2021]
      #keep year 2021
      dfspring20 = dfspring20[dfspring20['ReadoutYear']==2020]
      dfsummer20 = dfsummer20[dfsummer20['ReadoutYear']==2020]
      dfautumn20 = dfautumn20[dfautumn20['ReadoutYear']==2020]
      #keep year 2020

[18]: dfwinter22 = dfwinter22[dfwinter22['ReadoutYear']!=2019]
      dfwinter22 = dfwinter22[dfwinter22['ReadoutYear']!=2020]
      #remove year 2019 & 2020
      dfwinter21 = dfwinter21[dfwinter21['ReadoutYear']!=2019]
      dfwinter21 = dfwinter21[dfwinter21['ReadoutYear']!=2022]
      #remove year 2019 & 2022
      dfwinter20 = dfwinter20[dfwinter20['ReadoutYear']!=2021]
      dfwinter20 = dfwinter20[dfwinter20['ReadoutYear']!=2022]
      #remove year 2021 & 2022

[19]: dfwinter22 = dfwinter22.loc[~((dfwinter22['ReadoutYear'] == 2021) &
      ↪(dfwinter22['ReadoutWeek'] <= 10)),:]
      dfwinter22 = dfwinter22.loc[~((dfwinter22['ReadoutYear'] == 2022) &
      ↪(dfwinter22['ReadoutWeek'] >= 35)),:]
      #remove weeks that belonging to year 2021 & 2022
      dfwinter21 = dfwinter21.loc[~((dfwinter21['ReadoutYear'] == 2020) &
      ↪(dfwinter21['ReadoutWeek'] <= 10)),:]
      dfwinter21 = dfwinter21.loc[~((dfwinter21['ReadoutYear'] == 2021) &
      ↪(dfwinter21['ReadoutWeek'] >= 35)),:]
      #remove weeks that belonging to year 2020 & 2021
      dfwinter20 = dfwinter20.loc[~((dfwinter20['ReadoutYear'] == 2020) &
      ↪(dfwinter20['ReadoutWeek'] >= 35)),:]
      #remove weeks belonging to winter 2020
```

## 7 Copy df and keep 21 and 22

```
[20]: df2019 = df2020 = df2021 = df2022 = df.copy()
      dfnor2020 = dfnor2021 = dfnor2022 = dfswe2020 = dfswe2021 = dfswe2022 = df.
      ↪copy()
      #copy df to separate df's
      df2019 = df2019[df2019['ReadoutYear']==2019]
      #year 2021 to df 2020
      df2020 = df2020[df2020['ReadoutYear']==2020]
      #year 2021 to df 2020
```

```
df2021 = df2021[df2021['ReadoutYear']==2021]
#year 2021 to df 2021
df2022 = df2022[df2022['ReadoutYear']==2022]
#year 2022 to df 2022
```

```
[21]: df2019.to_excel('DTCFaultcodes2019.xlsx', index=False)
df2020.to_excel('DTCFaultcodes2010.xlsx', index=False)
df2021.to_excel('DTCFaultcodes2021.xlsx', index=False)
df2022.to_excel('DTCFaultcodes2022.xlsx', index=False)
#export to excel
```

## 8 Counting total amount of vehicles and faults per country

```
[22]: df_vehicle=df.drop_duplicates(subset=['Vehicle_Id'])
#dropping duplicate vehicles
```

```
[23]: df_vehicle_cnt_2122 = df_vehicle['Country'].value_counts().to_frame().
↳reset_index()
#count the amount of vehicles in each country
```

```
[24]: df_fault_cnt_2122 = df['Country'].value_counts().to_frame().reset_index()
df_fault_cnt_2122.columns.values[0] = 'Country'
df_fault_cnt_2122.columns.values[1] = 'Faults'
df_fault_cnt_2122 = df_fault_cnt_2122.rename_axis('index')
#counting faults per country

df_vehicle_cnt_2122 = df_vehicle['Country'].value_counts().to_frame().
↳reset_index()
df_vehicle_cnt_2122.columns.values[0] = 'Country'
df_vehicle_cnt_2122.columns.values[1] = 'Vehicles'
df_vehicle_cnt_2122 = df_vehicle_cnt_2122.rename_axis('index')
#count the amount of vehicles in each country
```

## 9 Merge vehicles and faults then send it so excel

```
[25]: faults_vehicles_merged_21_22 = pd.merge(df_vehicle_cnt_2122, df_fault_cnt_2122)
```

```
[26]: faults_vehicles_merged_21_22.to_excel('Merged faults and vehicles per country_
↳19-22.xlsx', index=False)
```



## 10 Sort Sweden and Norway

```
[27]: dfnor = dfswe = df.copy()
      #Copy df to dfswe and dfnor
      dfswe = dfswe[dfsw['Country']=='SWE']
      #remove all but swe and put in dfswe
      dfnor = dfnor[dfnor['Country']=='NOR']
      #remove all but nor and put in dfnor

[28]: df_veh_swe = dfswe['Vehicle_Id'].value_counts().to_frame()
      df_veh_nor = dfnor['Vehicle_Id'].value_counts().to_frame()
      #Count faults per vehicle per country (Sweden and Norway in this case)

[29]: df_veh_swe.columns.values[0] = 'Faults'
      df_veh_swe = df_veh_swe.rename_axis('Vehicle_Id').reset_index()
      #counting faults per vehicle in Sweden

      df_veh_nor.columns.values[0] = 'Faults'
      df_veh_nor = df_veh_nor.rename_axis('Vehicle_Id').reset_index()
      #counting faults per vehicle in Norway

[30]: df_veh_swe.index.name='Vehicle'
      df_veh_swe = df_veh_swe[['Faults']]
      #drop vehicle id sweden
      df_veh_nor.index.name='Vehicle'
      df_veh_nor = df_veh_nor[['Faults']]
      #drop vehicle id sweden

[31]: dfswenor = faults_vehicles_merged_21_22.iloc[:2]
      #drop all countrys besides Norway and Sweden

[32]: import scipy.stats as stats

      def norm(s):
          return stats.norm.pdf(s, s.mean(), s.std())

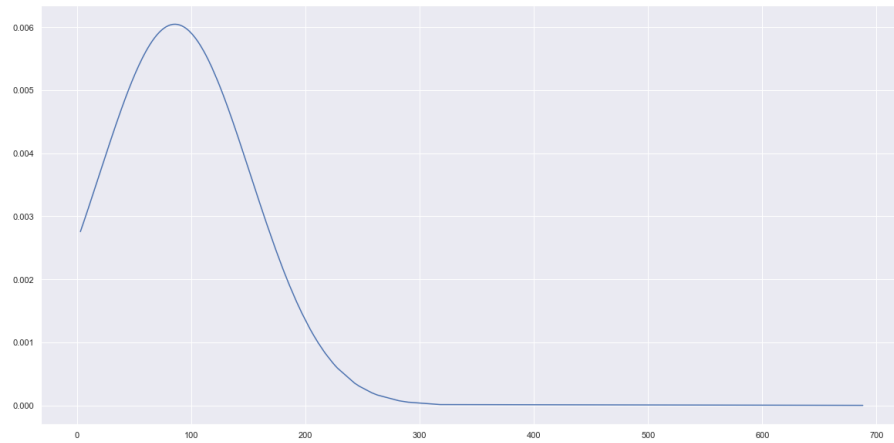
      test_swe = df_veh_swe.copy()
      test_swe['Norm'] = norm(test_swe['Faults'])

[33]: import matplotlib.pyplot as plt
      import seaborn as sns

      sns.set()

      plt.figure(figsize=(20, 10))
      plt.plot('Faults', 'Norm', data=test_swe)
      plt.show()
```





```
[34]: import scipy.stats as stats

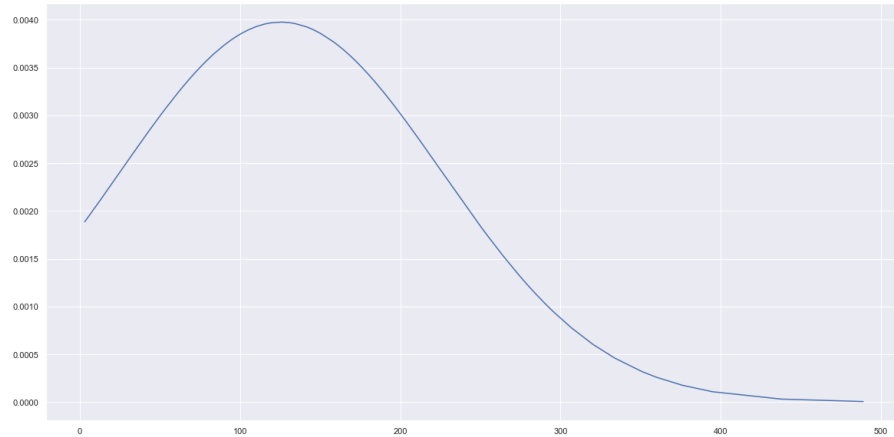
def medel(n):
    return stats.norm.pdf(n, n.mean(), n.std())

test_nor = df_veh_nor.copy()
test_nor['Norm'] = medel(test_nor['Faults'])
```

```
[35]: import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

plt.figure(figsize=(20, 10))
plt.plot('Faults', 'Norm', data=test_nor)
plt.show()
```



```
[36]: with pd.ExcelWriter('Faults on vehicles Sweden Norway.xlsx') as writer:
        df_veh_swe.to_excel(writer, sheet_name='Faults Swe Nor', startrow=1,
        ↪startcol=0)
        df_veh_nor.to_excel(writer, sheet_name='Faults Swe Nor', startrow=1,
        ↪startcol=3)
        dfswenor.to_excel(writer, sheet_name='Faults Swe Nor', startrow=1,
        ↪startcol=6, index=False)
        writer.save()
        #print out to one excel sheet
```

## 11 Sweden and Norway 2021 and 2022

```
[37]: dfnor2020 = dfnor2020[dfnor2020['ReadoutYear']==2020]
        dfswe2020 = dfswe2020[dfswe2020['ReadoutYear']==2020]
        #year 2021 to dfnor and dfswe
        dfnor2021 = dfnor2021[dfnor2021['ReadoutYear']==2021]
        dfswe2021 = dfswe2021[dfswe2021['ReadoutYear']==2021]
        #year 2021 to dfnor and dfswe
        dfnor2022 = dfnor2022[dfnor2022['ReadoutYear']==2022]
        dfswe2022 = dfswe2022[dfswe2022['ReadoutYear']==2022]
        #year 2022 to dfnor and dfswe
```

```
[38]: #Keep NOR and SWE to the dfs
        dfswe2020 = dfswe2020[dfswe2020['Country']=='SWE']
        dfswe2021 = dfswe2021[dfswe2021['Country']=='SWE']
        dfswe2022 = dfswe2022[dfswe2022['Country']=='SWE']
        #remove all but swe and put in dfswe
```

```
dfnor2020 = dfnor2020[dfnor2020['Country']=='NOR']
dfnor2021 = dfnor2021[dfnor2021['Country']=='NOR']
dfnor2022 = dfnor2022[dfnor2022['Country']=='NOR']
#remove all but nor and put in dfnor
```

```
[39]: df_veh_swe_20 = dfswe2020['Vehicle_Id'].value_counts().to_frame()
df_veh_swe_21 = dfswe2021['Vehicle_Id'].value_counts().to_frame()
df_veh_swe_22 = dfswe2022['Vehicle_Id'].value_counts().to_frame()
#Count faults per vehicle per country (Sweden 2021 and 2022 in this case)
df_veh_nor_20 = dfnor2020['Vehicle_Id'].value_counts().to_frame()
df_veh_nor_21 = dfnor2021['Vehicle_Id'].value_counts().to_frame()
df_veh_nor_22 = dfnor2022['Vehicle_Id'].value_counts().to_frame()
#Count faults per vehicle per country (Norway 2021 and 2022 in this case)
```

```
[40]: df_veh_swe_20.columns.values[0] = 'Faults'
df_veh_swe_20 = df_veh_swe_20.rename_axis('Vehicle_Id').reset_index()
#counting faults per vehicle in Sweden 2020
df_veh_swe_21.columns.values[0] = 'Faults'
df_veh_swe_21 = df_veh_swe_21.rename_axis('Vehicle_Id').reset_index()
#counting faults per vehicle in Sweden 2021
df_veh_swe_22.columns.values[0] = 'Faults'
df_veh_swe_22 = df_veh_swe_22.rename_axis('Vehicle_Id').reset_index()
#counting faults per vehicle in Sweden 2022

df_veh_nor_20.columns.values[0] = 'Faults'
df_veh_nor_20 = df_veh_nor_20.rename_axis('Vehicle_Id').reset_index()
#counting faults per vehicle in Norway 2021
df_veh_nor_21.columns.values[0] = 'Faults'
df_veh_nor_21 = df_veh_nor_21.rename_axis('Vehicle_Id').reset_index()
#counting faults per vehicle in Norway 2021
df_veh_nor_22.columns.values[0] = 'Faults'
df_veh_nor_22 = df_veh_nor_22.rename_axis('Vehicle_Id').reset_index()
#counting faults per vehicle in Norway 2022
```

```
[41]: df_veh_swe_20.index.name='Vehicle se 20'
df_veh_swe_20 = df_veh_swe_20[['Faults']]

df_veh_swe_21.index.name='Vehicle se 21'
df_veh_swe_21 = df_veh_swe_21[['Faults']]

df_veh_swe_22.index.name='Vehicle se 22'
df_veh_swe_22 = df_veh_swe_22[['Faults']]
#drop vehicle id Sweden
df_veh_nor_20.index.name='Vehicle no 20'
df_veh_nor_20 = df_veh_nor_20[['Faults']]

df_veh_nor_21.index.name='Vehicle no 21'
```

```
df_veh_nor_21 = df_veh_nor_21[['Faults']]

df_veh_nor_22.index.name='Vehicle no 22'
df_veh_nor_22 = df_veh_nor_22[['Faults']]
#drop vehicle id Norway
```

```
[42]: with pd.ExcelWriter('Summary Sweden and Norway.xlsx') as writer:
        df_veh_swe_20.to_excel(writer, sheet_name='Sweden, Norway 2021-2022',
        startrow=1, startcol=0)
        df_veh_nor_20.to_excel(writer, sheet_name='Sweden, Norway 2021-2022',
        startrow=1, startcol=3)
        df_veh_swe_21.to_excel(writer, sheet_name='Sweden, Norway 2021-2022',
        startrow=1, startcol=6)
        df_veh_nor_21.to_excel(writer, sheet_name='Sweden, Norway 2021-2022',
        startrow=1, startcol=9)
        df_veh_swe_22.to_excel(writer, sheet_name='Sweden, Norway 2021-2022',
        startrow=1, startcol=12)
        df_veh_nor_22.to_excel(writer, sheet_name='Sweden, Norway 2021-2022',
        startrow=1, startcol=15)
        writer.save()
#print out to one excel sheet
```

## 12 Total country and faults seperated 2019 to 2022

```
[43]: df_vehicle_2019=df2019.drop_duplicates(subset=['Vehicle_Id'])
df_vehicle_2020=df2020.drop_duplicates(subset=['Vehicle_Id'])
df_vehicle_2021=df2021.drop_duplicates(subset=['Vehicle_Id'])
df_vehicle_2022=df2022.drop_duplicates(subset=['Vehicle_Id'])
#drop duplicate vehicles 2021 and 2022
```

```
[44]: df_fault_cnt_2019 = df2019['Country'].value_counts().to_frame().reset_index()
df_fault_cnt_2019.columns.values[0] = 'Country'
df_fault_cnt_2019.columns.values[1] = 'Faults'
df_fault_cnt_2019 = df_fault_cnt_2019.rename_axis('index')
#counting faults per country 2020
#
df_fault_cnt_2020 = df2020['Country'].value_counts().to_frame().reset_index()
df_fault_cnt_2020.columns.values[0] = 'Country'
df_fault_cnt_2020.columns.values[1] = 'Faults'
df_fault_cnt_2020 = df_fault_cnt_2020.rename_axis('index')
#counting faults per country 2020
#
df_fault_cnt_2021 = df2021['Country'].value_counts().to_frame().reset_index()
df_fault_cnt_2021.columns.values[0] = 'Country'
df_fault_cnt_2021.columns.values[1] = 'Faults'
```

```

df_fault_cnt_2021 = df_fault_cnt_2021.rename_axis('index')
#counting faults per country 2021
#
df_fault_cnt_2022 = df2022['Country'].value_counts().to_frame().reset_index()
df_fault_cnt_2022.columns.values[0] = 'Country'
df_fault_cnt_2022.columns.values[1] = 'Faults'
df_fault_cnt_2022 = df_fault_cnt_2022.rename_axis('index')
#counting faults per country 2022

df_vehicle_cnt_2019 = df_vehicle_2019['Country'].value_counts().to_frame().
    ↪reset_index()
df_vehicle_cnt_2019.columns.values[0] = 'Country'
df_vehicle_cnt_2019.columns.values[1] = 'Vehicles'
df_vehicle_cnt_2019 = df_vehicle_cnt_2019.rename_axis('index')
#count the amount of vehicles in each country

df_vehicle_cnt_2020 = df_vehicle_2020['Country'].value_counts().to_frame().
    ↪reset_index()
df_vehicle_cnt_2020.columns.values[0] = 'Country'
df_vehicle_cnt_2020.columns.values[1] = 'Vehicles'
df_vehicle_cnt_2020 = df_vehicle_cnt_2020.rename_axis('index')
#count the amount of vehicles in each country

df_vehicle_cnt_2021 = df_vehicle_2021['Country'].value_counts().to_frame().
    ↪reset_index()
df_vehicle_cnt_2021.columns.values[0] = 'Country'
df_vehicle_cnt_2021.columns.values[1] = 'Vehicles'
df_vehicle_cnt_2021 = df_vehicle_cnt_2021.rename_axis('index')
#count the amount of vehicles in each country

df_vehicle_cnt_2022 = df_vehicle_2022['Country'].value_counts().to_frame().
    ↪reset_index()
df_vehicle_cnt_2022.columns.values[0] = 'Country'
df_vehicle_cnt_2022.columns.values[1] = 'Vehicles'
df_vehicle_cnt_2022 = df_vehicle_cnt_2022.rename_axis('index')
#count the amount of vehicles in each country

```

### 13 Merge and excel

```

[45]: faults_vehicles_merged_19 = pd.merge(df_vehicle_cnt_2019, df_fault_cnt_2019)
#Merge country and faults 2021
faults_vehicles_merged_20 = pd.merge(df_vehicle_cnt_2020, df_fault_cnt_2020)
#Merge country and faults 2022
faults_vehicles_merged_21 = pd.merge(df_vehicle_cnt_2021, df_fault_cnt_2021)
#Merge country and faults 2021
faults_vehicles_merged_22 = pd.merge(df_vehicle_cnt_2022, df_fault_cnt_2022)

```

```
#Merge country and faults 2022

[46]: faults_vehicles_merged_19.to_excel('Merged fault and vehicles per country 2019.
      ↪xlsx', index=False)
      faults_vehicles_merged_20.to_excel('Merged fault and vehicles per country 2020.
      ↪xlsx', index=False)
      faults_vehicles_merged_21.to_excel('Merged fault and vehicles per country 2021.
      ↪xlsx', index=False)
      faults_vehicles_merged_22.to_excel('Merged fault and vehicles per country 2022.
      ↪xlsx', index=False)

[47]: with pd.ExcelWriter('Summary faults per year.xlsx') as writer:
      faults_vehicles_merged_19.to_excel(writer, sheet_name='Faults 2019-2022',
      ↪index=False, startrow=1, startcol=0)
      faults_vehicles_merged_20.to_excel(writer, sheet_name='Faults 2019-2022',
      ↪index=False, startrow=1, startcol=4)
      faults_vehicles_merged_21.to_excel(writer, sheet_name='Faults 2019-2022',
      ↪index=False, startrow=1, startcol=8)
      faults_vehicles_merged_22.to_excel(writer, sheet_name='Faults 2019-2022',
      ↪index=False, startrow=1, startcol=12)
      writer.save()
      #print out to one excel sheet
```

Check faultcount

## 14 Country by season

Sorting Faults per country and vehicles per country divided into seasons

```
[49]: df_vehicle_spring20=dfspring20.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles spring 2021
      df_vehicle_summer20=dfsummer20.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles summer 2021
      df_vehicle_autumn20=dfautumn20.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles autumn 2021
      df_vehicle_winter20=dfwinter20.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles winter 2021
      df_vehicle_spring21=dfspring21.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles spring 2021
      df_vehicle_summer21=dfsummer21.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles summer 2021
      df_vehicle_autumn21=dfautumn21.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles autumn 2021
      df_vehicle_winter21=dfwinter21.drop_duplicates(subset=['Vehicle_Id'])
      #drop duplicate vehicles winter 2021
      df_vehicle_spring22=dfspring22.drop_duplicates(subset=['Vehicle_Id'])
```

```

#drop duplicate vehicles spring 2022
df_vehicle_winter22=dfwinter22.drop_duplicates(subset=['Vehicle_Id'])
#drop duplicate vehicles winter 2022

[51]: df_fault_cnt_spring20 = dfspring20['Country'].value_counts().to_frame().
      ↪reset_index()
df_fault_cnt_spring20.columns.values[0] = 'Country'
df_fault_cnt_spring20.columns.values[1] = 'Faults'
df_fault_cnt_spring20 = df_fault_cnt_spring20.rename_axis('index')
#counting faults per country spring 2020
#
df_fault_cnt_summer20 = dfsummer20['Country'].value_counts().to_frame().
      ↪reset_index()
df_fault_cnt_summer20.columns.values[0] = 'Country'
df_fault_cnt_summer20.columns.values[1] = 'Faults'
df_fault_cnt_summer20 = df_fault_cnt_summer20.rename_axis('index')
#counting faults per country summer 2020
#
df_fault_cnt_autumn20 = dfautumn20['Country'].value_counts().to_frame().
      ↪reset_index()
df_fault_cnt_autumn20.columns.values[0] = 'Country'
df_fault_cnt_autumn20.columns.values[1] = 'Faults'
df_fault_cnt_autumn20 = df_fault_cnt_autumn20.rename_axis('index')
#counting faults per country autumn 2020
#
df_fault_cnt_winter20 = dfwinter20['Country'].value_counts().to_frame().
      ↪reset_index()
df_fault_cnt_winter20.columns.values[0] = 'Country'
df_fault_cnt_winter20.columns.values[1] = 'Faults'
df_fault_cnt_winter20 = df_fault_cnt_winter20.rename_axis('index')
#counting faults per country winter 2020
#
df_fault_cnt_spring21 = dfspring21['Country'].value_counts().to_frame().
      ↪reset_index()
df_fault_cnt_spring21.columns.values[0] = 'Country'
df_fault_cnt_spring21.columns.values[1] = 'Faults'
df_fault_cnt_spring21 = df_fault_cnt_spring21.rename_axis('index')
#counting faults per country spring 2021
#
df_fault_cnt_summer21 = dfsummer21['Country'].value_counts().to_frame().
      ↪reset_index()
df_fault_cnt_summer21.columns.values[0] = 'Country'
df_fault_cnt_summer21.columns.values[1] = 'Faults'
df_fault_cnt_summer21 = df_fault_cnt_summer21.rename_axis('index')
#counting faults per country summer 2021
#

```

```
df_fault_cnt_autumn21 = dfautumn21['Country'].value_counts().to_frame().
↳reset_index()
df_fault_cnt_autumn21.columns.values[0] = 'Country'
df_fault_cnt_autumn21.columns.values[1] = 'Faults'
df_fault_cnt_autumn21 = df_fault_cnt_autumn21.rename_axis('index')
#counting faults per country autumn 2021
#
df_fault_cnt_winter21 = dfwinter21['Country'].value_counts().to_frame().
↳reset_index()
df_fault_cnt_winter21.columns.values[0] = 'Country'
df_fault_cnt_winter21.columns.values[1] = 'Faults'
df_fault_cnt_winter21 = df_fault_cnt_winter21.rename_axis('index')
#counting faults per country winter 2021
#
df_fault_cnt_winter22 = dfwinter22['Country'].value_counts().to_frame().
↳reset_index()
df_fault_cnt_winter22.columns.values[0] = 'Country'
df_fault_cnt_winter22.columns.values[1] = 'Faults'
df_fault_cnt_winter22 = df_fault_cnt_winter22.rename_axis('index')
#counting faults per country winter 2022
#
df_fault_cnt_spring22 = dfspring22['Country'].value_counts().to_frame().
↳reset_index()
df_fault_cnt_spring22.columns.values[0] = 'Country'
df_fault_cnt_spring22.columns.values[1] = 'Faults'
df_fault_cnt_spring22 = df_fault_cnt_spring22.rename_axis('index')
#counting faults per country spring 2022
#-----
df_vehicle_cnt_spring20 = df_vehicle_spring20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_spring20.columns.values[0] = 'Country'
df_vehicle_cnt_spring20.columns.values[1] = 'Vehicles'
df_vehicle_cnt_spring20 = df_vehicle_cnt_spring20.rename_axis('index')
#count the amount of vehicles in each country spring 2020
#
df_vehicle_cnt_summer20 = df_vehicle_summer20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_summer20.columns.values[0] = 'Country'
df_vehicle_cnt_summer20.columns.values[1] = 'Vehicles'
df_vehicle_cnt_summer20 = df_vehicle_cnt_summer20.rename_axis('index')
#count the amount of vehicles in each country summer 2020
#
df_vehicle_cnt_autumn20 = df_vehicle_autumn20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_autumn20.columns.values[0] = 'Country'
df_vehicle_cnt_autumn20.columns.values[1] = 'Vehicles'
```



```

df_vehicle_cnt_autumn20 = df_vehicle_cnt_autumn20.rename_axis('index')
#count the amount of vehicles in each country autumn 2020
#
df_vehicle_cnt_winter20 = df_vehicle_winter20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_winter20.columns.values[0] = 'Country'
df_vehicle_cnt_winter20.columns.values[1] = 'Vehicles'
df_vehicle_cnt_winter20 = df_vehicle_cnt_winter20.rename_axis('index')
#count the amount of vehicles in each country winter 2020
#
df_vehicle_cnt_spring21 = df_vehicle_spring21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_spring21.columns.values[0] = 'Country'
df_vehicle_cnt_spring21.columns.values[1] = 'Vehicles'
df_vehicle_cnt_spring21 = df_vehicle_cnt_spring21.rename_axis('index')
#count the amount of vehicles in each country spring 2021
#
df_vehicle_cnt_summer21 = df_vehicle_summer21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_summer21.columns.values[0] = 'Country'
df_vehicle_cnt_summer21.columns.values[1] = 'Vehicles'
df_vehicle_cnt_summer21 = df_vehicle_cnt_summer21.rename_axis('index')
#count the amount of vehicles in each country summer 2021
#
df_vehicle_cnt_autumn21 = df_vehicle_autumn21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_autumn21.columns.values[0] = 'Country'
df_vehicle_cnt_autumn21.columns.values[1] = 'Vehicles'
df_vehicle_cnt_autumn21 = df_vehicle_cnt_autumn21.rename_axis('index')
#count the amount of vehicles in each country autumn 2021
#
df_vehicle_cnt_winter21 = df_vehicle_winter21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_winter21.columns.values[0] = 'Country'
df_vehicle_cnt_winter21.columns.values[1] = 'Vehicles'
df_vehicle_cnt_winter21 = df_vehicle_cnt_winter21.rename_axis('index')
#count the amount of vehicles in each country winter 2021
#
df_vehicle_cnt_winter22 = df_vehicle_winter22['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_winter22.columns.values[0] = 'Country'
df_vehicle_cnt_winter22.columns.values[1] = 'Vehicles'
df_vehicle_cnt_winter22 = df_vehicle_cnt_winter22.rename_axis('index')
#count the amount of vehicles in each country winter 2022
#

```

```
df_vehicle_cnt_spring22 = df_vehicle_spring22['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_cnt_spring22.columns.values[0] = 'Country'
df_vehicle_cnt_spring22.columns.values[1] = 'Vehicles'
df_vehicle_cnt_spring22 = df_vehicle_cnt_spring22.rename_axis('index')
#count the amount of vehicles in each country spring 2022
```

## 15 Merge data and send to excel

```
[53]: faults_vehicles_merged_winter20 = pd.merge(df_vehicle_cnt_winter20,↳
↳df_fault_cnt_winter20)
faults_vehicles_merged_spring20 = pd.merge(df_vehicle_cnt_spring20,↳
↳df_fault_cnt_spring20)
faults_vehicles_merged_summer20 = pd.merge(df_vehicle_cnt_summer20,↳
↳df_fault_cnt_summer20)
faults_vehicles_merged_autumn20 = pd.merge(df_vehicle_cnt_autumn20,↳
↳df_fault_cnt_autumn20)
#merged data by seasons 2020
faults_vehicles_merged_winter21 = pd.merge(df_vehicle_cnt_winter21,↳
↳df_fault_cnt_winter21)
faults_vehicles_merged_spring21 = pd.merge(df_vehicle_cnt_spring21,↳
↳df_fault_cnt_spring21)
faults_vehicles_merged_summer21 = pd.merge(df_vehicle_cnt_summer21,↳
↳df_fault_cnt_summer21)
faults_vehicles_merged_autumn21 = pd.merge(df_vehicle_cnt_autumn21,↳
↳df_fault_cnt_autumn21)
#merged data by seasons 2021
faults_vehicles_merged_winter22 = pd.merge(df_vehicle_cnt_winter22,↳
↳df_fault_cnt_winter22)
faults_vehicles_merged_spring22 = pd.merge(df_vehicle_cnt_spring22,↳
↳df_fault_cnt_spring22)
#merged data by seasons 2022
```

```
[54]: faults_vehicles_merged_winter21.to_excel('Merged faults season winter 2021.
↳xlsx', index=False)
faults_vehicles_merged_spring21.to_excel('Merged faults season spring 2021.
↳xlsx', index=False)
faults_vehicles_merged_summer21.to_excel('Merged faults season summer 2021.
↳xlsx', index=False)
faults_vehicles_merged_autumn21.to_excel('Merged faults season sutumn 2021.
↳xlsx', index=False)
#export merged season 2021 data to excel
faults_vehicles_merged_winter22.to_excel('Merged faults season winter 2022.
↳xlsx', index=False)
```

```

faults_vehicles_merged_spring22.to_excel('Merged faults season spring 2022.
↳xlsx', index=False)
#export merged season 2022 data to excel

```

```

[55]: with pd.ExcelWriter('Summary Faults by country all season.xlsx') as writer:
        faults_vehicles_merged_21_22.to_excel(writer, sheet_name='Faults seasons',
        ↳index=False, startrow=1, startcol=0)

        faults_vehicles_merged_winter20.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=4)
        faults_vehicles_merged_spring20.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=8)
        faults_vehicles_merged_summer20.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=12)
        faults_vehicles_merged_autumn20.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=16)

        faults_vehicles_merged_winter21.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=20)
        faults_vehicles_merged_spring21.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=24)
        faults_vehicles_merged_summer21.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=28)
        faults_vehicles_merged_autumn21.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=32)

        faults_vehicles_merged_winter22.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=36)
        faults_vehicles_merged_spring22.to_excel(writer, sheet_name='Faults_
        ↳seasons', index=False, startrow=1, startcol=40)
        writer.save()
#print out to one excel sheet

```

## 16 Have a look at components all seasons

```

[56]: df_ftb_comp_full_period = df[df['FailureTypeDescription']=='Component Failures']
#single out component failures full period
df_comp_vehicle_full_period=df_ftb_comp_full_period.
↳drop_duplicates(subset=['Vehicle_Id'])
#Drop duplicate vehicles

df_fault_comp_full_period = df_ftb_comp_full_period['Country'].value_counts().
↳to_frame().reset_index()
df_fault_comp_full_period.columns.values[0] = 'Country'
df_fault_comp_full_period.columns.values[1] = 'Faults'

```

```
df_fault_comp_full_period = df_fault_comp_full_period.rename_axis('index')
#counting faults per country spring 2022
#-----
df_vehicle_comp_full_period = df_comp_vehicle_full_period['Country'].
    ↳value_counts().to_frame().reset_index()
df_vehicle_comp_full_period.columns.values[0] = 'Country'
df_vehicle_comp_full_period.columns.values[1] = 'Vehicles'
df_vehicle_comp_full_period = df_vehicle_comp_full_period.rename_axis('index')
#count the amount of vehicles in each country spring 2021
```

```
[57]: df_ftb_comp_veh_fp = df[df['FailureTypeDescription']=='Component Failures']
#single out component failures full period
#-----
df_comp_veh_1922 = df_ftb_comp_veh_fp['Vehicle_Id'].value_counts().to_frame()

df_comp_veh_1922.columns.values[0] = 'Faults'
df_comp_veh_1922 = df_comp_veh_1922.rename_axis('Vehicle_Id').reset_index()

df_comp_veh_1922.index.name='Vehicle'
df_comp_veh_1922 = df_comp_veh_1922[['Faults']].reset_index()
```

```
[58]: comp_vehicles_merged_full_period = pd.merge(df_vehicle_comp_full_period,␣
    ↳df_fault_comp_full_period)
#merge data
```

Component errors over the seasons

```
[61]: dfspring20_ftb_comp =␣
    ↳dfspring20[dfspring20['FailureTypeDescription']=='Component Failures']
dfsummer20_ftb_comp =␣
    ↳dfsummer20[dfsummer20['FailureTypeDescription']=='Component Failures']
dfaumn20_ftb_comp =␣
    ↳dfaumn20[dfaumn20['FailureTypeDescription']=='Component Failures']
dfwinter20_ftb_comp =␣
    ↳dfwinter20[dfwinter20['FailureTypeDescription']=='Component Failures']
#season 2020
dfspring21_ftb_comp =␣
    ↳dfspring21[dfspring21['FailureTypeDescription']=='Component Failures']
dfsummer21_ftb_comp =␣
    ↳dfsummer21[dfsummer21['FailureTypeDescription']=='Component Failures']
dfaumn21_ftb_comp =␣
    ↳dfaumn21[dfaumn21['FailureTypeDescription']=='Component Failures']
dfwinter21_ftb_comp =␣
    ↳dfwinter21[dfwinter21['FailureTypeDescription']=='Component Failures']
#season 2021
```

```

dfspring22_ftb_comp =
↳dfspring22[dfspring22['FailureTypeDescription']=='Component Failures']
dfwinter22_ftb_comp =
↳dfwinter22[dfwinter22['FailureTypeDescription']=='Component Failures']
#season 2022
#Singles out component errors over seasons

```

```

[62]: df_comp_vehicle_spring20=dfspring20_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_summer20=dfsummer20_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_autumn20=dfautumn20_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_winter20=dfwinter20_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
#remove duplicate vehicles
df_comp_vehicle_spring21=dfspring21_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_summer21=dfsummer21_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_autumn21=dfautumn21_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_winter21=dfwinter21_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
#remove duplicate vehicles
df_comp_vehicle_spring22=dfspring22_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])
df_comp_vehicle_winter22=dfwinter22_ftb_comp.
↳drop_duplicates(subset=['Vehicle_Id'])

```

```

[64]: dfspring21_ftb_comp.to_excel('Components Spring 2021.xlsx', index=False)
dfspring21_ftb_comp.to_excel('Components Summer 2021.xlsx', index=False)
dfautumn21_ftb_comp.to_excel('Components Autumn 2021.xlsx', index=False)
dfwinter21_ftb_comp.to_excel('Components Winter 2021.xlsx', index=False)
dfspring22_ftb_comp.to_excel('Components Spring 2022.xlsx', index=False)
dfwinter22_ftb_comp.to_excel('Components Winter 2022.xlsx', index=False)
#Send the fault 9 to excel

```

```

[65]: df_fault_comp_spring20 = dfspring20_ftb_comp['Country'].value_counts().
↳to_frame().reset_index()
df_fault_comp_spring20.columns.values[0] = 'Country'
df_fault_comp_spring20.columns.values[1] = 'Faults'
df_fault_comp_spring20 = df_fault_comp_spring20.rename_axis('index')
#counting faults per country spring 2020
#

```

```
df_fault_comp_summer20 = dfsummer20_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_summer20.columns.values[0] = 'Country'  
df_fault_comp_summer20.columns.values[1] = 'Faults'  
df_fault_comp_summer20 = df_fault_comp_summer20.rename_axis('index')  
#counting faults per country summer 2020  
#  
df_fault_comp_autumn20 = dfautumn20_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_autumn20.columns.values[0] = 'Country'  
df_fault_comp_autumn20.columns.values[1] = 'Faults'  
df_fault_comp_autumn20 = df_fault_comp_autumn20.rename_axis('index')  
#counting faults per country autumn 2020  
#  
df_fault_comp_winter20 = dfwinter20_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_winter20.columns.values[0] = 'Country'  
df_fault_comp_winter20.columns.values[1] = 'Faults'  
df_fault_comp_winter20 = df_fault_comp_winter20.rename_axis('index')  
#counting faults per country winter 2020  
#  
df_fault_comp_spring21 = dfspring21_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_spring21.columns.values[0] = 'Country'  
df_fault_comp_spring21.columns.values[1] = 'Faults'  
df_fault_comp_spring21 = df_fault_comp_spring21.rename_axis('index')  
#counting faults per country spring 2021  
#  
df_fault_comp_summer21 = dfsummer21_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_summer21.columns.values[0] = 'Country'  
df_fault_comp_summer21.columns.values[1] = 'Faults'  
df_fault_comp_summer21 = df_fault_comp_summer21.rename_axis('index')  
#counting faults per country summer 2021  
#  
df_fault_comp_autumn21 = dfautumn21_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_autumn21.columns.values[0] = 'Country'  
df_fault_comp_autumn21.columns.values[1] = 'Faults'  
df_fault_comp_autumn21 = df_fault_comp_autumn21.rename_axis('index')  
#counting faults per country autumn 2021  
#  
df_fault_comp_winter21 = dfwinter21_ftb_comp['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_fault_comp_winter21.columns.values[0] = 'Country'  
df_fault_comp_winter21.columns.values[1] = 'Faults'
```

```

df_fault_comp_winter21 = df_fault_comp_winter21.rename_axis('index')
#counting faults per country winter 2021
#
df_fault_comp_spring22 = dfspring22_ftb_comp['Country'].value_counts().
↳to_frame().reset_index()
df_fault_comp_spring22.columns.values[0] = 'Country'
df_fault_comp_spring22.columns.values[1] = 'Faults'
df_fault_comp_spring22 = df_fault_comp_spring22.rename_axis('index')
#counting faults per country winter 2022
#
df_fault_comp_winter22 = dfwinter22_ftb_comp['Country'].value_counts().
↳to_frame().reset_index()
df_fault_comp_winter22.columns.values[0] = 'Country'
df_fault_comp_winter22.columns.values[1] = 'Faults'
df_fault_comp_winter22 = df_fault_comp_winter22.rename_axis('index')
#counting faults per country spring 2022
#-----
df_vehicle_comp_spring20 = df_comp_vehicle_spring20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_spring20.columns.values[0] = 'Country'
df_vehicle_comp_spring20.columns.values[1] = 'Vehicles'
df_vehicle_comp_spring20 = df_vehicle_comp_spring20.rename_axis('index')
#count the amount of vehicles in each country spring 2020
#
df_vehicle_comp_summer20 = df_comp_vehicle_summer20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_summer20.columns.values[0] = 'Country'
df_vehicle_comp_summer20.columns.values[1] = 'Vehicles'
df_vehicle_comp_summer20 = df_vehicle_comp_summer20.rename_axis('index')
#count the amount of vehicles in each country summer 2020
#
df_vehicle_comp_autumn20 = df_comp_vehicle_autumn20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_autumn20.columns.values[0] = 'Country'
df_vehicle_comp_autumn20.columns.values[1] = 'Vehicles'
df_vehicle_comp_autumn20 = df_vehicle_comp_autumn20.rename_axis('index')
#count the amount of vehicles in each country autumn 2020
#
df_vehicle_comp_winter20 = df_comp_vehicle_winter20['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_winter20.columns.values[0] = 'Country'
df_vehicle_comp_winter20.columns.values[1] = 'Vehicles'
df_vehicle_comp_winter20 = df_vehicle_comp_winter20.rename_axis('index')
#count the amount of vehicles in each country winter 2020
#

```

```
df_vehicle_comp_spring21 = df_comp_vehicle_spring21['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_vehicle_comp_spring21.columns.values[0] = 'Country'  
df_vehicle_comp_spring21.columns.values[1] = 'Vehicles'  
df_vehicle_comp_spring21 = df_vehicle_comp_spring21.rename_axis('index')  
#count the amount of vehicles in each country spring 2021  
#  
df_vehicle_comp_summer21 = df_comp_vehicle_summer21['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_vehicle_comp_summer21.columns.values[0] = 'Country'  
df_vehicle_comp_summer21.columns.values[1] = 'Vehicles'  
df_vehicle_comp_summer21 = df_vehicle_comp_summer21.rename_axis('index')  
#count the amount of vehicles in each country summer 2021  
#  
df_vehicle_comp_autumn21 = df_comp_vehicle_autumn21['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_vehicle_comp_autumn21.columns.values[0] = 'Country'  
df_vehicle_comp_autumn21.columns.values[1] = 'Vehicles'  
df_vehicle_comp_autumn21 = df_vehicle_comp_autumn21.rename_axis('index')  
#count the amount of vehicles in each country autumn 2021  
#  
df_vehicle_comp_winter21 = df_comp_vehicle_winter21['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_vehicle_comp_winter21.columns.values[0] = 'Country'  
df_vehicle_comp_winter21.columns.values[1] = 'Vehicles'  
df_vehicle_comp_winter21 = df_vehicle_comp_winter21.rename_axis('index')  
#count the amount of vehicles in each country winter 2021  
#  
df_vehicle_comp_spring22 = df_comp_vehicle_spring22['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_vehicle_comp_spring22.columns.values[0] = 'Country'  
df_vehicle_comp_spring22.columns.values[1] = 'Vehicles'  
df_vehicle_comp_spring22 = df_vehicle_comp_spring22.rename_axis('index')  
#count the amount of vehicles in each country winter 2022  
#  
df_vehicle_comp_winter22 = df_comp_vehicle_winter22['Country'].value_counts().  
    ↳to_frame().reset_index()  
df_vehicle_comp_winter22.columns.values[0] = 'Country'  
df_vehicle_comp_winter22.columns.values[1] = 'Vehicles'  
df_vehicle_comp_winter22 = df_vehicle_comp_winter22.rename_axis('index')  
#count the amount of vehicles in each country spring 2022
```

Merge faults and country

```
[66]: comp_vehicles_merged_winter20 = pd.merge(df_vehicle_comp_winter20, ↳  
    ↳df_fault_comp_winter20)
```



```

df_vehicle_comp_spring21 = df_comp_vehicle_spring21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_spring21.columns.values[0] = 'Country'
df_vehicle_comp_spring21.columns.values[1] = 'Vehicles'
df_vehicle_comp_spring21 = df_vehicle_comp_spring21.rename_axis('index')
#count the amount of vehicles in each country spring 2021
#
df_vehicle_comp_summer21 = df_comp_vehicle_summer21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_summer21.columns.values[0] = 'Country'
df_vehicle_comp_summer21.columns.values[1] = 'Vehicles'
df_vehicle_comp_summer21 = df_vehicle_comp_summer21.rename_axis('index')
#count the amount of vehicles in each country summer 2021
#
df_vehicle_comp_autumn21 = df_comp_vehicle_autumn21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_autumn21.columns.values[0] = 'Country'
df_vehicle_comp_autumn21.columns.values[1] = 'Vehicles'
df_vehicle_comp_autumn21 = df_vehicle_comp_autumn21.rename_axis('index')
#count the amount of vehicles in each country autumn 2021
#
df_vehicle_comp_winter21 = df_comp_vehicle_winter21['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_winter21.columns.values[0] = 'Country'
df_vehicle_comp_winter21.columns.values[1] = 'Vehicles'
df_vehicle_comp_winter21 = df_vehicle_comp_winter21.rename_axis('index')
#count the amount of vehicles in each country winter 2021
#
df_vehicle_comp_spring22 = df_comp_vehicle_spring22['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_spring22.columns.values[0] = 'Country'
df_vehicle_comp_spring22.columns.values[1] = 'Vehicles'
df_vehicle_comp_spring22 = df_vehicle_comp_spring22.rename_axis('index')
#count the amount of vehicles in each country winter 2022
#
df_vehicle_comp_winter22 = df_comp_vehicle_winter22['Country'].value_counts().
↳to_frame().reset_index()
df_vehicle_comp_winter22.columns.values[0] = 'Country'
df_vehicle_comp_winter22.columns.values[1] = 'Vehicles'
df_vehicle_comp_winter22 = df_vehicle_comp_winter22.rename_axis('index')
#count the amount of vehicles in each country spring 2022

```

Merge faults and country

```

[66]: comp_vehicles_merged_winter20 = pd.merge(df_vehicle_comp_winter20,↳
↳df_fault_comp_winter20)

```

```

comp_vehicles_merged_spring20 = pd.merge(df_vehicle_comp_spring20,
↳df_fault_comp_spring20)
comp_vehicles_merged_summer20 = pd.merge(df_vehicle_comp_summer20,
↳df_fault_comp_summer20)
comp_vehicles_merged_autumn20 = pd.merge(df_vehicle_comp_autumn20,
↳df_fault_comp_autumn20)
#merged data by seasons 2020
comp_vehicles_merged_winter21 = pd.merge(df_vehicle_comp_winter21,
↳df_fault_comp_winter21)
comp_vehicles_merged_spring21 = pd.merge(df_vehicle_comp_spring21,
↳df_fault_comp_spring21)
comp_vehicles_merged_summer21 = pd.merge(df_vehicle_comp_summer21,
↳df_fault_comp_summer21)
comp_vehicles_merged_autumn21 = pd.merge(df_vehicle_comp_autumn21,
↳df_fault_comp_autumn21)
#merged data by seasons 2021
comp_vehicles_merged_winter22 = pd.merge(df_vehicle_comp_winter22,
↳df_fault_comp_winter22)
comp_vehicles_merged_spring22 = pd.merge(df_vehicle_comp_spring22,
↳df_fault_comp_spring22)
#merged data by seasons 2022

```

Send to excel

```

[68]: comp_vehicles_merged_winter21.to_excel('Merged component faults winter 2021.
↳xlsx', index=False)
comp_vehicles_merged_spring21.to_excel('Merged component faults spring 2021.
↳xlsx', index=False)
comp_vehicles_merged_summer21.to_excel('Merged component faults summer 2021.
↳xlsx', index=False)
comp_vehicles_merged_autumn21.to_excel('Merged component faults sutumn 2021.
↳xlsx', index=False)
#export merged season 2021 data to excel
comp_vehicles_merged_winter22.to_excel('Merged component faults winter 2022.
↳xlsx', index=False)
comp_vehicles_merged_spring22.to_excel('Merged component faults spring 2022.
↳xlsx', index=False)
#export merged season 2022 data to excel

[69]: with pd.ExcelWriter('Summary comp faults season.xlsx') as writer:
    comp_vehicles_merged_full_period.to_excel(writer, sheet_name='Component_
↳faults by seasons', index=False, startrow=1, startcol=0)
    comp_vehicles_merged_winter20.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=4)
    comp_vehicles_merged_spring20.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=8)

```

```

    comp_vehicles_merged_summer20.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=12)
    comp_vehicles_merged_autumn20.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=16)
    comp_vehicles_merged_winter21.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=20)
    comp_vehicles_merged_spring21.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=24)
    comp_vehicles_merged_summer21.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=28)
    comp_vehicles_merged_autumn21.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=32)
    comp_vehicles_merged_winter22.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=36)
    comp_vehicles_merged_spring22.to_excel(writer, sheet_name='Component faults_
↳by seasons', index=False, startrow=1, startcol=40)

    df_comp_veh_1922.to_excel(writer, sheet_name='Comp Faults Per Veh',
↳index=False)
    writer.save()
#print out to one excel sheet

```

## 17 Nodes by season

## 18 Nodes by season to excel

## 19 Count amount of total faults

```

[75]: df_ftb_21_22 = df['FTB'].value_counts().to_frame()
      df_ftb_21_22.columns.values[0] = 'Faults'
      df_ftb_21_22 = df_ftb_21_22.rename_axis('FTB')

```

## 20 Extract to excel

```

[76]: df_ftb_21_22.to_excel('Total Faults 2021-2022.xlsx')

```

## 21 Count amount of faults per season

```

[77]: # value_counts as dataframe
      df_ftb_spring20 = dfspring20['FTB'].value_counts().to_frame()
      df_ftb_summer20 = dfsummer20['FTB'].value_counts().to_frame()
      df_ftb_autumn20 = dfautumn20['FTB'].value_counts().to_frame()
      df_ftb_winter20 = dfwinter20['FTB'].value_counts().to_frame()

```

```
df_ftb_spring21 = dfspring21['FTB'].value_counts().to_frame()
df_ftb_summer21 = dfsummer21['FTB'].value_counts().to_frame()
df_ftb_autumn21 = dfautumn21['FTB'].value_counts().to_frame()
df_ftb_winter21 = dfwinter21['FTB'].value_counts().to_frame()
df_ftb_winter22 = dfwinter22['FTB'].value_counts().to_frame()
df_ftb_spring22 = dfspring22['FTB'].value_counts().to_frame()
```

```
[78]: #df.rename(columns = {"FTB" : "Count"}, inplace = True)
df_ftb_spring20.columns.values[0] = 'Faults'
df_ftb_spring20 = df_ftb_spring20.rename_axis('FTB')
df_ftb_summer20.columns.values[0] = 'Faults'
df_ftb_summer20 = df_ftb_summer20.rename_axis('FTB')
df_ftb_autumn20.columns.values[0] = 'Faults'
df_ftb_autumn20 = df_ftb_autumn20.rename_axis('FTB')
df_ftb_winter20.columns.values[0] = 'Faults'
df_ftb_winter20 = df_ftb_winter20.rename_axis('FTB')
df_ftb_spring21.columns.values[0] = 'Faults'
df_ftb_spring21 = df_ftb_spring21.rename_axis('FTB')
df_ftb_summer21.columns.values[0] = 'Faults'
df_ftb_summer21 = df_ftb_summer21.rename_axis('FTB')
df_ftb_autumn21.columns.values[0] = 'Faults'
df_ftb_autumn21 = df_ftb_autumn21.rename_axis('FTB')
df_ftb_winter21.columns.values[0] = 'Faults'
df_ftb_winter21 = df_ftb_winter21.rename_axis('FTB')
df_ftb_winter22.columns.values[0] = 'Faults'
df_ftb_winter22 = df_ftb_winter22.rename_axis('FTB')
df_ftb_spring22.columns.values[0] = 'Faults'
df_ftb_spring22 = df_ftb_spring22.rename_axis('FTB')
```

## 22 Extract to excel

```
[79]: with pd.ExcelWriter('FTB All Seasons.xlsx') as writer:
    df_ftb_winter20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=0)
    df_ftb_spring20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=3)
    df_ftb_summer20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=6)
    df_ftb_autumn20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=9)
    df_ftb_winter21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=12)
    df_ftb_spring21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=15)
    df_ftb_summer21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
    ↪startcol=18)
```

```

df_ftb_autumn21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=21)
df_ftb_winter22.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=24)
df_ftb_spring22.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=27)
writer.save()
#print out to one excel sheet

```

### 23 Count amount of faults per vehicle

```

[80]: df_veh_winter20 = dfwinter20['Vehicle_Id'].value_counts().to_frame()
df_veh_spring20 = dfspring20['Vehicle_Id'].value_counts().to_frame()
df_veh_summer20 = dfsummer20['Vehicle_Id'].value_counts().to_frame()
df_veh_autumn20 = dfautumn20['Vehicle_Id'].value_counts().to_frame()
#count amount of vehicles each season 2020
df_veh_winter21 = dfwinter21['Vehicle_Id'].value_counts().to_frame()
df_veh_spring21 = dfspring21['Vehicle_Id'].value_counts().to_frame()
df_veh_summer21 = dfsummer21['Vehicle_Id'].value_counts().to_frame()
df_veh_autumn21 = dfautumn21['Vehicle_Id'].value_counts().to_frame()
#count amount of vehicles each season 2021
df_veh_winter22 = dfwinter22['Vehicle_Id'].value_counts().to_frame()
df_veh_spring22 = dfspring22['Vehicle_Id'].value_counts().to_frame()
#count amount of vehicles each season 2022

[81]: #2020
df_veh_autumn20.columns.values[0] = 'Faults'
df_veh_autumn20 = df_veh_autumn20.rename_axis('Vehicle_Id').reset_index()
df_veh_spring20.columns.values[0] = 'Faults'
df_veh_spring20 = df_veh_spring20.rename_axis('Vehicle_Id').reset_index()
df_veh_summer20.columns.values[0] = 'Faults'
df_veh_summer20 = df_veh_summer20.rename_axis('Vehicle_Id').reset_index()
df_veh_winter20.columns.values[0] = 'Faults'
df_veh_winter20 = df_veh_winter20.rename_axis('Vehicle_Id').reset_index()
#2021
df_veh_autumn21.columns.values[0] = 'Faults'
df_veh_autumn21 = df_veh_autumn21.rename_axis('Vehicle_Id').reset_index()
df_veh_spring21.columns.values[0] = 'Faults'
df_veh_spring21 = df_veh_spring21.rename_axis('Vehicle_Id').reset_index()
df_veh_summer21.columns.values[0] = 'Faults'
df_veh_summer21 = df_veh_summer21.rename_axis('Vehicle_Id').reset_index()
df_veh_winter21.columns.values[0] = 'Faults'
df_veh_winter21 = df_veh_winter21.rename_axis('Vehicle_Id').reset_index()
#2022
df_veh_winter22.columns.values[0] = 'Faults'
df_veh_winter22 = df_veh_winter22.rename_axis('Vehicle_Id').reset_index()

```

```
df_veh_spring22.columns.values[0] = 'Faults'
df_veh_spring22 = df_veh_spring22.rename_axis('Vehicle_Id').reset_index()
```

```
[82]: #drop vehicle id
#2020
df_veh_autumn20.index.name='Vehicle'
df_veh_autumn20 = df_veh_autumn20[['Faults']]
df_veh_spring20.index.name='Vehicle'
df_veh_spring20 = df_veh_spring20[['Faults']]
df_veh_summer20.index.name='Vehicle'
df_veh_summer20 = df_veh_summer20[['Faults']]
df_veh_winter20.index.name='Vehicle'
df_veh_winter20 = df_veh_winter20[['Faults']]
#2021
df_veh_autumn21.index.name='Vehicle'
df_veh_autumn21 = df_veh_autumn21[['Faults']]
df_veh_spring21.index.name='Vehicle'
df_veh_spring21 = df_veh_spring21[['Faults']]
df_veh_summer21.index.name='Vehicle'
df_veh_summer21 = df_veh_summer21[['Faults']]
df_veh_winter21.index.name='Vehicle'
df_veh_winter21 = df_veh_winter21[['Faults']]
#2022
df_veh_winter22.index.name='Vehicle'
df_veh_winter22 = df_veh_winter22[['Faults']]
df_veh_spring22.index.name='Vehicle'
df_veh_spring22 = df_veh_spring22[['Faults']]
```

## 24 Extract to excel

```
[83]: with pd.ExcelWriter('Summary faults by season.xlsx') as writer:
    df_veh_19_22.to_excel(writer, sheet_name='Faults seasons', startrow=1,
        ↪startcol=0)

    df_veh_winter20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
        ↪startcol=3)
    df_veh_spring20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
        ↪startcol=6)
    df_veh_summer20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
        ↪startcol=9)
    df_veh_autumn20.to_excel(writer, sheet_name='Faults seasons', startrow=1,
        ↪startcol=12)

    df_veh_winter21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
        ↪startcol=15)
```

```
df_veh_spring21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=18)
df_veh_summer21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=21)
df_veh_autumn21.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=24)

df_veh_winter22.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=27)
df_veh_spring22.to_excel(writer, sheet_name='Faults seasons', startrow=1,
↳startcol=30)
writer.save()
#print out to one excel sheet
```

## 25 Normal distribution graph

Normal distribution for all faults per vehicle

```
[85]: import scipy.stats as stats

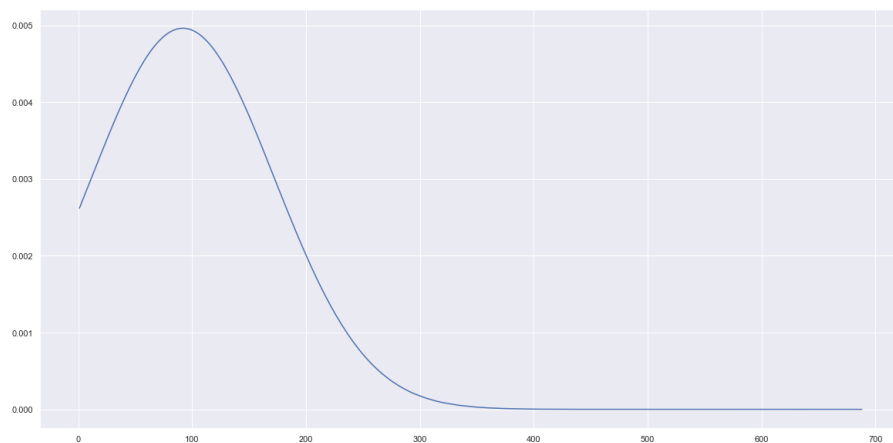
def medel(n):
    return stats.norm.pdf(n, n.mean(), n.std())

norm_dist_all_seasons = df_veh_19_22.copy()
norm_dist_all_seasons['Norm'] = medel(norm_dist_all_seasons['Faults'])

[86]: import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

plt.figure(figsize=(20, 10))
plt.plot('Faults', 'Norm', data=norm_dist_all_seasons)
plt.show()
```



## 26 How's it Going?

```
[87]: print('I am Done!')  
      #Printout when done
```

I am Done!





**CHALMERS**