

# Developing a traffic load model for bridges based on registered license plates

J.R. van Dam







# Developing a traffic load model for bridges based on registered license plates

Verifying structural reliability of existing structures by using Monte Carlo  
simulations with a load model based on license plates

By

J.R. van Dam

To obtain the degree of Master of Science  
at the Delft University of Technology,  
to be defended publicly on the 14<sup>th</sup> of October 2020.

Student number:	4162137	
Project duration:	September 2019 – October 2020	
Thesis committee:	Dr. ir. O. Morales Napoles,	TU Delft, Chairman
	Dr. ir. C.B.M. Blom	TU Delft / Gemeente Rotterdam
	Ir. J.M. Houben	TU Delft
	M.A. Mendoza-Lugo MSc	TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>





## Preface

This thesis concludes my educational career at the Delft University of Technology, which I started in 2016. During my time at the university, my interest for concrete structures and structural engineering in general grew. Alongside the growth of ambition for structural engineering, I always wondered whether certain processes could be automatized through programming. I am therefore very grateful to Kees Blom and the municipality of Rotterdam who offered me a subject which covered both of my interests.

Therefore, I would like to deeply thank my main supervisor Kees Blom for his expertise regarding doing research, providing the right questions to help me further progress during the research and the meaningful discussions we have had during the course of this thesis. I would like to thank Oswaldo Morales Napoles for his profession regarding probabilistic design, dedicating his time answering my questions and for his time being the chairman of the committee. I would also like to thank Lambert Houben for dedicating his time to attend the meetings and answering my questions and providing important insights regarding axle loads and axle configurations. Finally, I would like to thank Miguel Mendoza-Lugo for his time to answer countless questions regarding extreme value analysis, providing me a workable file of WIM data and suggesting directions of areas to be researched. Further I would like to thank the municipality of Rotterdam for offering me the opportunity to study this subject and to work at their office.

Furthermore, I would like to thank my friends and family and especially my girlfriend Mariëlle who helped me to keep focus and providing emotional support during the course of this thesis.

The subject of this thesis surrounds the structural reliability of existing bridges based on alternate load models other than the ones prescribed in the Eurocode. The main load model suggested in this thesis is based on information based on license plate registrations and on WIM data, which I hope serves as a steppingstone for involving different sources of data to construct traffic loads.

To the reader, please enjoy reading this thesis as much as I enjoyed working on it.

*J.R. van Dam  
Katwijk, September 2020*



[This page is intentionally left blank]

## Summary

New and existing bridges in the Netherlands must abide by structural safety codes, such as the Eurocode. In this code, structural safety is expressed through the reliability index  $\beta$ . For certain reference periods a threshold value for  $\beta$  exists. When applying prescribed load models given in the Eurocode, the structure is guaranteed to at least fulfil to this threshold value. However, these prescribed load models are deterministic in nature and can be rather conservative for bridges in urban areas.

This thesis focusses on creating a probabilistic load model based on actual traffic loading by making use of a camera system that registers license plates to check whether the vehicle is allowed to enter the inner city of Rotterdam due to environmental zones. From this camera system data, technical information such as wheelbase, legally allowed axle loads, gross vehicle weight and such can be extracted since they are coupled to license plates. This technical information is then used to create load models based on actual registered traffic. This load model represents trucks as point loads with interspatial axle distances. In total, one year of collected data by the camera system is stored, called the LP data. This load model is then used in a probabilistic reliability analysis as a load variable input.

When comparing the LP data with available weigh-in-motion (WIM) data from two measurement locations in Rotterdam, it turned out that the LP data does not incorporate under- and overloaded axles and was overestimating the accompanying reliability index. Hence to account for this, an axle load factor  $\eta_i$  is introduced to simulate under- and overloaded axles. This factor  $\eta_i$  is based on the WIM data and is different for each vehicle type. With the use of this factor, a second, improved load model is constructed. This is referred to as the modified LP data.

A third and final load model was constructed from the available WIM data, called the WIM model. For each of these three load models the load effects were calculated, and distributions were fitted accordingly for simulating several 25 year periods of traffic. The output of these load models is a load-effect maxima distribution that can serve as a direct input in a probabilistic reliability analysis.

With these three load models, a hypothetical slab with a span length of 10 m was probabilistically analysed where the LP model, the modified LP model and the WIM model resulted in reliability indexes of 4.8, 4.1 and 3.7 respectively. When compared to the requirement in the Eurocode, all load models comply. Concluding from this, the modified LP model suggested in this thesis can be used as a load input in a probabilistic verification for this very considered bridge location. For this load model to be applicable to multiple bridges, more research must be done since only one location was considered in this thesis. However, the suggested approach to construct load models based on license plates can be used to verify the applicability to multiple bridges.

[This page is intentionally left blank]



## List of figures

Figure 1 - Location of measurements.....	7
Figure 2 - Indicated location of measurement systems .....	8
Figure 3 - Excerpt of raw data as given by the camera system .....	8
Figure 4 - Excerpt of RDW database. Weights in kg, distances in cm. ....	11
Figure 5 - Flow chart for coupling license plates and excerpt of script.....	12
Figure 6 - Database for location 4400-C with only N3 vehicles classified. Weights in kg, distances in cm. ....	13
Figure 7 - Excerpt of RDW database for all registered trailers (O4).....	13
Figure 8 - Pie chart of registered O4 vehicle classes for location 4400-C. ....	14
Figure 9 - Database for location 4400-C with only O4 vehicles classified. Weights in kg, distances in cm. ....	14
Figure 10 - Database of registered N3-vehicles for location 4400-C.....	16
Figure 11 - Example of a 2-axled semi-trailer.....	16
Figure 12 - Pictures of different trailer types which can be carried by full trucks. Left: ADFT. Right: CCT.....	17
Figure 13 - Excerpt of python code of writing function $Y(L)$ .....	19
Figure 14 - $y$ matrices for 3-axled N3-O4 combinations.....	20
Figure 15 - Excerpt of python code to determine maximum load effect.....	21
Figure 16 - Matrices containing calculated load effects .....	21
Figure 17 - Matrix containing all maximum occurring load effect for each location. The 0's indicate the supports.....	22
Figure 18 - Bending moment vs GVW for a simply supported bridge with a span length of 10 m.....	22
Figure 19 - Frequency histogram of occurring bending moments.....	23
Figure 20 - Frequency histogram for location of maximum occurring load effect .....	23
Figure 21 - Compilation of 3 figures. Top left: relation between number of axles contributing to the maximum load effect and the total number of axles. Top right: Contribution of axles to the maximum load effect with added colour scale. Table: numerical values. ....	24
Figure 22 - Comparison between WIM data and LP data for measurement location Beukelsebrug. ..	25
Figure 23 - Comparison between WIM data and LP data for measurement location Oost-Abtsbrug..	26
Figure 24 – Caused load effect by vehicles with a GVW higher than 50t.....	26
Figure 25 – Comparison between axle loads for tipper trucks in the WIM data (left) and LP data (right).....	27
Figure 26 - Configuration of axles for tipper truck from WIM data. ....	27
Figure 27 - Distribution of $\eta_i$ for a 2-axled semi-truck carrying a single axle trailer.....	30
Figure 28 - Bending moments vs GVW for both the LP data and the WIM data. ....	31
Figure 29 - Bending moments vs GVW for the modified LP data and the WIM data. ....	31
Figure 30 - Comparison between the 3 load models. The black circle indicates vehicles falling in the extreme heavy combinations (EHC). ....	32
Figure 31 - Figures for bending moment vs GVW for a span length of 10 m for the modified LP data, containing axle load factor $\eta$ . Results are compared to the tandem load from $LM1$ . ....	32
Figure 32 - Figures for bending moment vs GVW for a span length of 10 m for the modified LP data, containing axle load factor $\eta$ . Results are compared to the full load model $LM1$ . ....	33
Figure 33 - Example of a non-linear cumulative distribution function. ....	34
Figure 34 - Example of the limit state function using a Monte Carlo simulation.....	35
Figure 35 - Example of results obtained by following suggested approach. Left: icdf's for different load models. Right: pfd's for different load models.....	36

Figure 36 - Figure of different parameters for different concrete grades. ....	37
Figure 37 - Distribution types and parameters for different concrete parameters.....	38
Figure 38 - Excerpt of python code to determine force in steel $F_{steel}$ . ....	39
Figure 39 - Schematically representation of acting forces in a concrete cross-section.....	39
Figure 40 - Excerpt of python code to determine force in the concrete $F_{conc}$ .....	39
Figure 41 - Excerpt of python code to calculate bending moment resistance $MRd$ . ....	40
Figure 42 - Excerpt of python code of running a Monte Carlo simulation.....	41
Figure 43 - Load scheme of LM1 from NEN-EN 1991-2.....	42
Figure 44 - Relative frequency histograms and fitted distributions for daily maxima for the modified LP load model. ....	44
Figure 45 - Fitted inverted cumulative distribution functions for daily maxima for the modified LP load model.....	45
Figure 46 - Relative frequency histograms and fitted distributions for monthly maxima for the modified LP load model.....	45
Figure 47 - Fitted inverted cumulative distribution functions for monthly maxima for the modified LP load model.....	46
Figure 48 - Relative frequency histograms and fitted distributions for yearly maxima for the modified LP load model. ....	46
Figure 49 - Fitted inverted cumulative distribution functions for yearly maxima for the modified LP load model.....	47
Figure 50 - Relative frequency histogram of distribution of resistance $MRd$ .....	47
Figure 51 – Results obtained from Monte Carlo simulation using the modified LP model as a load model input. ....	48
Figure 52 - Distributions for $S(x)$ and $R(x)$ with uncertainty factors $\theta S$ and $\theta R$ incorporated.....	48
Figure 53 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using modified LP model as a load model input. ....	49
Figure 54 - Probability density functions for $Sx$ and $R(x)$ used in the Monte Carlo simulation.....	49
Figure 55 - Results obtained from Monte Carlo simulation using the LP model as a load model input. ....	50
Figure 56 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using LP model as a load model input.....	50
Figure 57 - Probability density functions for $S(x)$ and $R(x)$ using the WIM model as a load model input. ....	51
Figure 58 - Results obtained from Monte Carlo simulation using the WIM model as a load model input. ....	51
Figure 59 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using WIM model as a load model input.....	52
Figure 60 - Overview of probability density functions for $S(x)$ for different span lengths.....	54
Figure 61 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using modified LP model as a load model input for different span lengths. ....	54
Figure 62 - Excerpt of available information through RDW.....	64
Figure 63 - Excerpt of section "Algemeen". General information of a 5-axled tipper truck.....	64
Figure 64 - Excerpt of section "Gewichten". Information about masses of a 5-axled tipper truck.....	64
Figure 65 - Excerpt of section "Technisch". Technical information of a 5-axled tipper truck.....	65
Figure 66 - Excerpt of section "Assen". Technical information of a 5-axled tipper truck. ....	65
Figure 67 -Subclassification of vehicles in the WIM data.....	66
Figure 68 - Frequency distribution of axle loads for 2- and 3-axled semi-trucks for LP and WIM data.....	67
Figure 69 - Frequency distribution of axle loads for 2-axled semi-trucks carrying 4-axled trailers.....	68

Figure 70 - Frequency distribution of axle loads for 3-axled semi-trucks for both LP data and WIM data.....	70
Figure 71 - Frequency distribution of interspatial axle distance DT34. Left: 3-axled semi-truck carrying 2-axled semi-trailer. Right: 3-axled semi-truck carrying 3-axled semi-trailer.....	71
Figure 72 - Frequency distribution for axle loads for 4-axled tipper trucks for both LP data and WIM data. Right figure is zoomed in to properly display WIM data.....	72
Figure 73 - Frequency distribution for axle loads for 5-axled tipper trucks for both LP data and WIM data. Right figure is zoomed in to properly display WIM data.....	73
Figure 74 - Frequency distributions for 4-axled mobile cranes and tipper trucks for both LP data and WIM data. Left: total vehicle length. Right: total wheelbase. ....	74
Figure 75 - Frequency distribution for axle loads of 4- and 5-axled mobile cranes for both LP data and WIM data.....	75
Figure 76 - Bending moment vs GVW for 4- and 5-axled mobile cranes acting on a span length of 10 m. ....	75
Figure 77 - Frequency distribution for axle loads for 2-axled full trucks for both LP data and WIM data. Right figure is zoomed in to properly distribution WIM data.....	76
Figure 78 - Frequency distributions of axle loads for 2-axled full trucks carrying 1-, 2-, or 3-axled trailers. ....	77
Figure 79 - Frequency distributions of axle loads for 3-axled full trucks for both LP data and WIM data. Right figure is zoomed in to properly display WIM data.....	79
Figure 80 - Frequency distributions of axle loads for 3-axled full trucks carrying 2- or 3-axled trailers for both LP data and WIM data.....	80
Figure 81 - Frequency distribution of axle loads for 4-axled full trucks for both LP data and WIM data. ....	81
Figure 82 - Frequency distributions for axle loads of 4-axled full trucks. Top left: No trailer. Top right: 2-axled trailer. Bottom left: 3-axled trailer. ....	82
Figure 83 - Bending moment vs GVW for vehicle category EHC. WIM results grouped by denoted criteria. LP data not grouped.....	84
Figure 84 - Relation between number of axles contributing to the maximum load effect and the total number of axles.....	87
Figure 85 - Frequency distributions for location of maximum load effect. Left: LP data. Right: WIM data.....	88
Figure 86 - Bending moment vs span length for 3 load models: mobile cranes, EHC, tandem load of LM1. ....	88
Figure 87 - Excerpt of WIM data showing contribution of axles to maximum load effect for 3-axled vehicle. ....	90
Figure 88 – Excerpt of WIM data showing contribution of axles to maximum load effect for 8-axled vehicle. ....	90
Figure 89 - 3D representation of contribution of axles to maximum load effect by adding colour scale. ....	90
Figure 90 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for daily maxima. Bottom left: Enhanced vision of tail of relative frequency. Bottom right: icdf of daily maxima and fitted distributions. ....	91
Figure 91 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for monthly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of monthly maxima and fitted distributions. ....	93



Figure 92 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for yearly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of yearly maxima and fitted distributions. ....	94
Figure 93 - Relative frequency histograms of daily-, monthly- and yearly maxima together with their respective fitted distribution.....	95
Figure 94 - Left: Relative frequency histogram of occurring load effect using LP model. Right: Relative frequency histogram with fitted distributions. ....	96
Figure 95 - Left: Tail of relative frequency histogram of daily maxima using LP model. Right: inverted cumulative distribution of daily maxima from LP data and fitted distributions. ....	96
Figure 96 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for monthly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of monthly maxima and fitted distributions. ....	97
Figure 97 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for yearly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of yearly maxima and fitted distributions. ....	98
Figure 98 – Top left: Relative frequency histogram of occurring load effect using WIM model. Top right: Relative frequency histogram with fitted distributions. Bottom left: Tail of relative frequency histogram of daily maxima using WIM model. Bottom right: inverted cumulative distribution of daily maxima from WIM model and fitted distributions. ....	99
Figure 99 – Top left: Relative frequency histogram of monthly maxima using WIM model. Top right: Relative frequency histogram with fitted distributions. Bottom left: Tail of relative frequency histogram of monthly maxima using WIM model. Bottom right: inverted cumulative distribution of monthly maxima from WIM model and fitted distributions.....	100
Figure 100 - Top left: Relative frequency histogram of yearly maxima using WIM model. Top right: Relative frequency histogram with fitted distributions. Bottom left: Tail of relative frequency histogram of yearly maxima using WIM model. Bottom right: inverted cumulative distribution of yearly maxima from WIM model and fitted distributions. ....	101
Figure 101- Distribution of $\eta_i$ for a 2-axled semi-truck carrying a single axle trailer. ....	102
Figure 102 - Distribution of $\eta_i$ for a 3-axled semi-truck carrying a 2-axled trailer. ....	103
Figure 103 – Distribution of $\eta_i$ for a 2-axled full truck carrying no trailer. ....	103
Figure 104 – Distribution of $\eta_i$ for a 3-axled full truck carrying no trailer. ....	104
Figure 105 – Distribution of $\eta_i$ for a 4-axled tipper/dump truck carrying no trailer. ....	104
Figure 106 – Distribution of $\eta_i$ for a 5-mobile crane carrying no trailer.....	105

## List of Tables

Table 1 - Description of vehicle categories. ....	4
Table 2 - Description of vehicle types, taken from the European guideline. ....	5
Table 3 - Explanation of column entries in RAW data given by camera system. ....	9
Table 4 - Explanation of parameters and characteristics .....	11
Table 5 - Clarification of used parameters .....	16
Table 6 - Statistical values for frequency histogram for location of maximum occurring load effect. ..	23
Table 7 - Summary of conclusions for comparison of axle loads between LP data and WIM data. ....	28
Table 8 - Summary of conclusions for comparison of interspatial axle distances between LP data and WIM data. ....	29
Table 9 - Distribution parameters for different steel parameters. ....	38
Table 10 - Distribution parameters for steel area. ....	38
Table 11 - Distribution parameters for model uncertainties $\theta S$ and $\theta R$ . ....	40
Table 12 - Parameters for yearly maxima distribution. ....	47
Table 13 - Input parameters for MCsim function. ....	47
Table 14 - Parameter input for different span lengths .....	53
Table 15 - Distribution parameters for different span lengths .....	53
Table 16 - Probability of failure and reliability index for different span lengths. ....	55
Table 17 – Overview of results .....	55
Table 18 - Overview of axle loads for 2-axled semi-trucks carrying 1-, 2-, or 3-axled trailers for both LP data and WIM data. ....	68
Table 19 - Overview of axle loads for 2-axled semi-trucks carrying 4-axled trailers for WIM data only. ....	68
Table 20 - Overview of interspatial axle distances for 2-axled semi-trucks carrying trailers for both LP data and WIM data. ....	69
Table 21 - Overview of axle loads for 3-axled semi-trucks carrying 1-, 2-, or 3-axled trailers for both LP data and WIM data. ....	70
Table 22 - Overview of interspatial axle distances for 3-axled semi-trucks carrying 1-, 2- or 3-axled trailers for both LP data and WIM data. ....	71
Table 23 - Overview of axle loads for 4-axled tipper trucks for both LP data and WIM data. ....	72
Table 24 - Overview of interspatial axle distances for 4-axled tipper trucks for both LP data and WIM data. ....	73
Table 25 - Overview of axle loads for 5-axled tipper trucks for both LP data and WIM data. ....	73
Table 26 - Overview of interspatial axle distances for 4-axled tipper trucks for both LP data and WIM data. ....	74
Table 27 - Overview of axle loads for 4- and 5-axled mobile cranes for both LP data and WIM data. ....	75
Table 28 - Overview of interspatial axle distances for 4- and 5-axled mobile cranes. ....	76
Table 29 - Overview of axle loads for 2-axled full trucks for both LP data and WIM data. ....	77
Table 30 - Overview of axle loads for 2-axled full trucks carrying 1-, 2-, or 3-axled trailers. ....	78
Table 31 - Overview of interspatial axle distances for 2-axled full trucks carrying 1-, 2-, or 3-axled trailers. ....	78
Table 32 - Overview of axle loads for 3-axled full trucks for both LP data and WIM data. ....	79
Table 33 - Overview of axle loads for 3-axled full trucks carrying 2- or 3-axled trailers for both LP data and WIM data. ....	80
Table 34 - Overview of interspatial axle distances for 3-axled full trucks carrying no, 2- or 3-axled trailers for both LP data and WIM data. ....	81
Table 35 - Overview of axle loads for 4-axled full trucks for both LP data and WIM data. ....	82

Table 36 - Overview of axle loads for 4-axled full trucks carrying no, 2- or 3-axled trailers for both LP data and WIM data.....	83
Table 37 - Overview of interspatial axle distances for 4-axled full trucks carrying no, 2- or 3-axled trailers for both LP data and WIM data.....	83
Table 38 - Overview of statistical values for bending moments for vehicle category EHC.....	84
Table 39 - Numerical overview of number of axles contribution to maximum load effect.....	87
Table 40 - Statistical values for locations of maximum load effects for both LP data and WIM data. .	88
Table 41 - Distribution parameters for fitted daily maxima distribution.....	92
Table 42 - Distribution parameters for fitted monthly maxima distribution.....	93
Table 43 - Distribution parameters for fitted yearly maxima distribution.....	95
Table 44 - Distribution parameters for fitted daily maxima distribution for LP model. ....	96
Table 45 - Distribution parameters for fitted monthly maxima distribution for LP model. ....	97
Table 46 - Distribution parameters for best fitted yearly maxima distribution for LP model. ....	98
Table 47 - Distribution parameters for best fitted daily maxima distribution for WIM model. ....	99
Table 48 - Distribution parameters for best fitted monthly maxima distribution for WIM model. ...	100
Table 49 - Distribution parameters for best fitted yearly maxima distribution for WIM model. ....	101



# 1 Contents

List of figures .....	v
List of Tables.....	ix
1 Contents .....	xi
2 Introduction.....	1
2.1 Introduction to the problem .....	1
2.2 Problem description .....	1
2.3 Problem statement.....	2
2.4 Goal and objectives .....	2
2.5 Traffic loading.....	3
2.6 Methodology .....	3
2.6.1 Literature study .....	3
2.6.2 Field research .....	3
2.6.3 Case study.....	3
3 Background and pre-processing.....	4
3.1 RDW Database.....	4
3.1.1 Vehicle category .....	4
3.1.2 Points of attention.....	5
4 Measurements .....	7
4.1 Background.....	7
4.1.1 Origin of data.....	7
4.1.2 Location of measurements.....	7
4.1.3 Raw data.....	8
4.2 Data of interest.....	9
4.2.1 Filtering criteria .....	9
4.2.2 Number of axles .....	10
4.2.3 Vehicle type and classification .....	10
5 Processing data.....	15
5.1 Creating load samples .....	15
5.1.1 Fundamental assumptions .....	15
5.1.2 Adopted strategy.....	15
5.1.3 Coupling N3-O4 vehicles.....	15
5.2 Calculating load effects .....	19
5.2.1 Approach .....	19
5.2.2 Processing.....	22
5.2.3 Results .....	22

5.3	Comparison with other data .....	25
5.3.1	WIM data.....	25
5.3.2	Results of comparison between LP- and WIM data.....	28
5.4	Adjustments of approach .....	29
5.4.1	Interspatial axle distances.....	29
5.4.2	Axle load factors $\eta$ .....	30
5.5	Post-processing .....	30
5.6	Influence of span length.....	32
6	Monte Carlo Simulation .....	34
6.1	Introduction.....	34
6.1.1	Basic principle.....	34
6.1.2	Reliability index .....	35
6.2	Approach .....	35
6.2.1	Determining load distribution $S(x)$ .....	35
6.2.2	Determining resistance distribution $R(x)$ .....	36
6.2.3	Method of simulating.....	40
6.3	Simulation.....	41
6.4	Post processing.....	41
7	Case study.....	42
7.1	Introduction.....	42
7.2	Case .....	42
7.2.1	Approach .....	42
7.2.2	Initial design .....	42
7.3	Probabilistic verification modified LP model.....	44
7.3.1	Load distribution $S(x)$ .....	44
7.3.2	Resistance distribution $R(x)$ .....	47
7.3.3	Simulation.....	48
7.3.4	Results .....	48
7.4	Probabilistic verification of various load models .....	49
7.4.1	LP data .....	49
7.4.2	WIM data.....	50
7.5	Dutch National Annex – Reliability index .....	52
7.6	Different span lengths.....	52
7.6.1	Initial design .....	52
7.6.2	Load distributions $S(x)$ .....	53
7.6.3	Simulation.....	54

7.6.4	Results .....	54
7.7	Summary of results.....	55
7.8	Intermediate conclusion.....	55
8	Conclusion .....	57
9	Discussion .....	59
9.1	Source of data .....	59
9.2	Coupling license plates and transforming data to load models.....	59
9.3	Generated load model and application.....	60
10	Recommendations.....	61
11	References.....	63
	Appendix A - RDW-excerpt.....	64
	Appendix B - Subclassification of vehicles.....	66
	Appendix C – Comparison between different vehicle types between LP data and WIM data .....	67
	Appendix D – Overloaded axles in WIM data.....	90
	Appendix E – Modified LP data: Distribution fitting to maxima.....	91
	Appendix F – LP data: Distribution fitting .....	96
	Appendix G – WIM data: Distribution fitting .....	99
	Appendix H – Examples of $\eta_i$ for different vehicle types.....	102
	Appendix I – Python codes .....	106

## 2 Introduction

### 2.1 Introduction to the problem

The Netherlands contains a lot of existing bridges which are built in the 1960s and 1970s. At the age of 50 years, some of these bridges need maintenance. Whenever maintenance is needed, the owner of these bridges must prove that they are still safe and have a remaining safe service lifetime. To prove this, the design has to be verified using the current load models. However, these current load models are representative for loads on highway bridges and are therefore likely unrealistic for more local bridges. Hence these current load models may result in economically unrealistic measures. Currently, there are exceptional rules for loads on existing bridges in the Netherlands. These exceptions are described in several national applied documents and they incorporate some load reducing factors for existing bridges. The current approach to evaluate an existing bridge is deterministic of nature, which implies that the output of the evaluation is fully determined by the input.

Recently, driven by environmental reasons, several cities throughout the Netherlands have set a so-called environmental zone. The aim of setting these environmental zones is to greatly reduce the air pollution and to improve general air quality. Hence cars with a diesel engine and a manufacturing date older than 01-01-2001 are not allowed to enter the inner city. Throughout the city of Rotterdam there are cameras installed that check license plates of vehicles to validate whether they are allowed to enter the inner city or not. This is done by checking the manufacturing date of the vehicles in a vehicle registration database owned by the Rijkdienst Wegverkeer (RDW), which contains technical, environmental, counter reading and much more information. Among this information is also the maximum allowed weight of a vehicle.

Hence, indirectly, this fine system is collecting a lot of data regarding cars using the infrastructure of the inner city of Rotterdam. This means that a great amount of data is obtained, which isn't yet utilized to its full potential. Here lies a great opportunity to get detailed insight into the actual traffic intensity and therefore, the actual traffic load on these local bridges. Using this data, it is possible to create a more realistic representation of the traffic loads for existing bridges based on traffic measurements and thus, a more economical measure can be taken.

### 2.2 Problem description

The main reason to evaluate existing structures within the municipality of Rotterdam is due to the increase in traffic loading as stated in the current norms, namely the Eurocodes. The owner of an existing structure, in this case the municipality of Rotterdam, has to prove that a specific structure is still safe and that it remains safe for its remaining service lifetime. For engineers, the most common way to evaluate an existing structure is to consult existing codes, apply reduced loading- and combination factors, apply increased resistance factors of applied materials and check whether the resistance of the structure exceeds the imposed load on the structure, i.e.:

$$R_d > E_d$$

Where  $R_d$  is the design value of the resistance and  $E_d$  is the design value of the load.

This current approach is deterministic, which implies that the output of such a calculation method is solely based on the input. The input for this calculation method is prescribed in the Eurocodes and has always to be applicable to all cases in the Netherlands. This directly implies the deterministic approach is conservative of nature since it is not bound to one specific structure. Due to the conservative nature

of this approach, the strengthening measures, when needed, are also likely to be conservative and unproportionally.

### 2.3 Problem statement

This raises the question whether it is possible to replace the current conservative approach, although relatively fast and easy to apply, with a more advanced approach that incorporates site-specific traffic loading. It is therefore of interest to investigate whether a probabilistic approach to determine the structural safety of an existing structure can provide a reliable basis to evaluate its structural safety. This leads to the following problem statement.

*In the coming years, several existing bridges in the city of Rotterdam have to be evaluated regarding their structural safety and their remaining structural safety during their service lifetime. It is expected that a majority of these structures will not comply with the deterministic approach as prescribed in the Eurocode. However, the conservative nature of the deterministic approach leaves an opportunity to optimize the structure to its full potential. Other, more sophisticated methods, such as a probabilistic approach, are yet fully researched to their potential.*

### 2.4 Goal and objectives

The goal of this research is to gain more insight in actual traffic loads occurring on existing bridges in the inner city of Rotterdam. The overall aim of this research is whether the obtained data can be used as a solid and substantiated basis to differentiate from the regular load models as written in the codes and provide a more realistic and site-specific distribution for traffic loads through live measurements, using car license plate registrations. With this aim the following goals and tasks are defined.

- 1) Objective 1: Understand the mechanism of current load models.
  - a. Which special measures can be taken for existing structures and how does this affect the current load models?
  - b. Has any research been done to determine the actual live load on existing bridges using on-site measurements?
- 2) Objective 2: Achieve a detailed insight in current traffic loadings on bridges throughout the inner city of Rotterdam.
  - a. Convert car license data to load models.
  - b. Obtain most severe loading scenarios using Monte Carlo simulations.
  - c. Redefine values for existing load models based on b).
- 3) Objective 3: Case study – Application of the defined load model to an existing bridge in Rotterdam and compare the results.
  - a. Calculation of an existing bridge in Rotterdam according to the following calculation methods:
    - i. Based on Eurocodes (assuming a hypothetical new design)
    - ii. Based on Dutch national documents NEN8700 / NEN8701.
    - iii. Based on previous research methods, preferably based on WIM data.
    - iv. Based on probabilistic design using the car license data.

## 2.5 Traffic loading

A key advantage of this approach is that it uses site specific data to provide a more realistic loading on an existing bridge. In a previous research done by Hellebrandt [1], locally weigh-in-motion data has been obtained. In this study, a more general approach using the registration of car license plates will be applied.

## 2.6 Methodology

### 2.6.1 Literature study

The first step is to understand what the current design codes prescribe for loads models for new and existing bridges. The literature study is ought to make clear how these load models are constructed and on which data they are based. This also includes the reduction factors written in mainly national annexes or national applied documents. For the second part, also research will be done to in which way the data will be obtained. Previous studies to this subject will also be consulted. This literature study will not be explicitly written out in the thesis, however.

### 2.6.2 Field research

For the second part of the research, the field research, it is necessary to have the data well-ordered. At this point in time<sup>1</sup>, it is unclear how the raw data will be represented. It is clear that the mass of passing vehicles is the most important parameter, but also the lane on which the vehicle was driving is of high interest as well as the number of axles. When the raw data is transformed into more usable information, it is possible to create histograms and distributions and accordingly mass probability functions and cumulative distribution functions in order to classify the probability of masses to occur. These functions serve as input for the traffic load. Then finally, using Monte Carlo simulations, the reliability analysis of an existing bridge can be determined.

### 2.6.3 Case study

For the case study, the data from the field research is applied correctly and a representative load model has been constructed. From that point onwards, the calculation procedure of an existing bridge will be pretty much straight forward. The intention is to make four, different calculations. First one considering the current, unreduced design values for a hypothetical new bridge. Secondly one that incorporates reduced partial factors, as stated in Dutch national applied documents. Thirdly one that incorporates a calculation method as prescribed in previous research, preferably based on WIM data. Finally, a probabilistic approach using car license plate registration which is further to be explained in this report.

---

<sup>1</sup> Within the first weeks of starting this thesis.

### 3 Background and pre-processing

This research is completely based on how the data is represented and which parameters are available in the data. In previous researches usually WIM data was used, which is more specified. In this research however, the acquired data is a much more global representation of the load traffic on the bridge since it uses legally allowed masses for the gross vehicle weight as well as the individual axle loads. In this chapter it is tried to gain more insight in how the data is presented, and how to account for certain uncertainties, defects of the current system and other points of attention. In this part of the research it is tried to transform the raw data into useable information.

#### 3.1 RDW Database

As stated before, the RDW registers all motorised vehicles sold or imported in the Netherlands in one, freely accessible database. This includes all motorised vehicles, ranging from a 2-axled moped to an 8-axled self-driving mobile crane. On the website [ovi.rdw.nl](http://ovi.rdw.nl), one can enter a license plate and find all information coupled to that particular vehicle. This database only contains vehicles with a Dutch license plate. Imported vehicles with a foreign license plates are not registered in this database.

##### 3.1.1 Vehicle category

The Dutch Vehicle Authority (RDW) registers vehicles in the Netherlands based on the European Vehicle Category. This classification is normalized throughout members of the European Union. The table below briefly describes all occurring vehicle categories.

Category	Vehicle(s)	Description
<b>M</b>	Passenger cars,buses and coaches	Motor vehicles with at least four tires designed and constructed for the carriage of passengers.
<b>N</b>	Commercial vehicles	Motor vehicles with at least four wheels designed and constructed for the carriage of goods
<b>O</b>	Trailers	Trailers (including semi-trailers)
<b>L</b>	2- and 3-wheeled vehicles	Mopeds, motorcycles, trikes and quads.
<b>T</b>	Wheeled tractors	Wheeled tractors
<b>C</b>	Track-laying tractors	Track-laying tractors
<b>R</b>	Agricultural trailers	Agricultural trailers
<b>S</b>	Interchangeable towed machinery	Interchangeable towed machinery

Table 1 - Description of vehicle categories<sup>2</sup>.

For urban areas it seems trivial to consider vehicle categories T, C, R and S. Also, vehicles in category L have a very low impact on traffic loading for bridges. As stated in TNO [13], only trucks or vehicles with a GVW > 3.5t are expected to have a great load impact. Hence vehicles falling in these categories are further explained. These are vehicle categories M, N and O. The description follows from the Directive 2007/46/EC of the European guideline<sup>3</sup>.

<sup>2</sup> Source: <https://www.rdw.nl/zakelijk/branches/fabrikanten-en-importeurs/typegoedkeuring-aanvragen/typegoedkeuren-voertuigen/voertuigcategorieen>

<sup>3</sup> Source: <https://eur-lex.europa.eu/legal-content/NL/TXT/HTML/?uri=CELEX:32007L0046&from=EN>

Category	Vehicle(s)	Description
<b>M1</b>	Passenger cars	Vehicles designed and constructed for the carriage of passengers and comprising no more than eight seats in addition to the driver's seat.
<b>M2</b>	Buses	Vehicles designed and constructed for the carriage of passengers, comprising more than eight seats in addition to the driver's seat, and having a maximum mass not exceeding 5 tonnes.
<b>M3</b>	Buses	Vehicles designed and constructed for the carriage of passengers, comprising more than eight seats in addition to the driver's seat, and having a maximum mass exceeding 5 tonnes.
<b>N1</b>	Light commercial trucks	Vehicles designed and constructed for the carriage of goods and having a maximum mass not exceeding 3,5 tonnes.
<b>N2</b>	Commercial trucks	Vehicles designed and constructed for the carriage of goods and having a maximum mass exceeding 3,5 tonnes but not exceeding 12 tonnes.
<b>N3</b>	Heavy commercial trucks	Vehicles designed and constructed for the carriage of goods and having a maximum mass exceeding 12 tonnes.
<b>O1</b>	Light trailers	Trailers with a maximum mass not exceeding 0,75 tonnes
<b>O2</b>	Light trailers	Trailers with a maximum mass exceeding 0,75 tonnes but not exceeding 3,5 tonnes.
<b>O3</b>	Trailers	Trailers with a maximum mass exceeding 3,5 tonnes but not exceeding 10 tonnes.
<b>O4</b>	Heavy trailers	Trailers with a maximum mass exceeding 10 tonnes.

Table 2 - Description of vehicle types, taken from the European guideline.

Since trucks of category N3 and trailers of O3 and O4 can be heavily loaded, measurements considering any of these vehicles are of interest. As shown by Hellebrandt [1] and TNO [12], the number of axles of a category N3 truck can span from 3 to 8. Thus, a more distinct categorization in the N3, O3 and O4 vehicle-types must be done. TNO [12] categorized heavy vehicles by the number of axles.

### 3.1.2 Points of attention

In appendix A an excerpt of the available data given by the RDW is elaborated. Due to the way this information is presented, a few points of attention arise. These points do not have to pose a great threat; however, they should be treated carefully.

- Whenever a vehicle is registered, the actual weight is not known. It can potentially range from the kerb mass to the maximum allowed. This can even be higher if vehicles are overloaded. Consequently, the actual axle loads are also not known. For the individual axle loads, no information is found about the range of allowed axle loads. Hence, only the static value for the axle loads is used.
- The total wheelbase of a vehicle is given, but whenever the vehicle has more than 2 axles, the spatial distances between axes is not. This means that solely from this information, it is not possible to reconstruct a load model purely based on the information given by the RDW. Hence measures need to be taken to account for this.



- There is no information about contact areas of the tires. This means that it is not possible to construct a load model (LM2) directly from the given information. In other words, only a global load model (LM1) can be constructed.
- There is no information given about the configuration of each individual axle, other than the track width of the axle. So, the axle can exist of 2 tires (single tires configuration) or 4 tires (double tire configuration). As shown by TNO [12], many different axle configurations are possible.

## 4 Measurements

As stated before, this research is completely based on how the data is represented and whether it is possible to transform the data into useable information. In the following paragraphs the source of data is presented as well as how the data is presented. According to paragraph 3.1.2, several measures are taken to make the data more usable.

### 4.1 Background

#### 4.1.1 Origin of data

Throughout the entire city of Rotterdam several cameras with different functions register vehicles that are passing by. In principle only the license plates, the manufacturing date and the type of combustion engine are registered. This information is directly taken from the database of the RDW, in which all motorized vehicles in the Netherlands are registered. This current system functions properly and is used by the municipality of Rotterdam.

However, the database of the RDW contains a lot more information considering technical aspects of a vehicle. These additional technical aspects of a vehicle are now added as additional information to the current system. This now somewhat modified system is used to gain more insight about traffic crossings, and in particular the rate of heavy vehicles at certain points in Rotterdam.

#### 4.1.2 Location of measurements

Basically, every location where a camera system is operating in the city of Rotterdam can be selected for measurements. In this study, a cluster of three camera systems closely located to the main highway A13 are considered. In the picture below these are shown in detail.



Figure 1 - Location of measurements<sup>4</sup>.

Every location can be opted for to do this kind of research. In this study however, a traffic analysis is used to gain more detailed insight in traffic loading on existing bridges. Camera 4400-C, as depicted above, is located a few meters from an existing bridge in the N209. In the picture below, the red circle indicates the operating camera system 4400-C.

---

<sup>4</sup> Source: municipality of Rotterdam.



Figure 2 - Indicated location of measurement systems<sup>5</sup>

#### 4.1.3 Raw data

The raw data is presented in a csv-file. CSV-files are files that are in a tabulated form, separated by commas and are only used to transfer text-only data from one system to another. Because the csv-files become large and will contain more than 1 million rows, the csv-files are being read using the programming language Python rather than using Microsoft Excel. All acts, modifications and calculations are being scripted in Python. Smaller, more convenient modifications to the dataset are done in Excel. The reader is assumed to have basic knowledge of programming. In the picture below, an example of a few entries is shown. Each column is briefly elaborated in the table given below.

ID	First(EuropeanVehicleCategory)	First(VehicleType)	First(KerbMass)	First(MaxAuthMassOfVehicle)	First(TechnicalMaxMassVehicle)	First(WheelBase)	First(CamId)	First(LocationCode)	First(LocationName)	First(VehicleLength)	Date	Time
0	M1	Personenauto	1015.0	1380.0	1380.0	230.0	2	1321	GRT_4400-C	357.0	1-1-2019	10:00:06 AM
441	N3	Bedrijfsauto	11870.0	19500.0	21000.0	600.0	2	1321	GRT_4400-C	1097.0	1-1-2019	11:42:57 PM
1324	N3	Bedrijfsauto	13840.0	28000.0	28000.0	697.0	2	1321	GRT_4400-C	1100.0	1-1-2019	1:49:55 PM
3670	N3	Bedrijfsauto	15870.0	49000.0	49000.0	695.0	2	1321	GRT_4400-C	933.0	1-1-2019	5:32:06 PM
114	O4	Oplegger	NaN	39000	39000.0	898.0	1	1325	GRT_4400-D	1402.0	1-1-2019	10:13:05 PM

Figure 3 - Excerpt of raw data as given by the camera system

<sup>5</sup> Source: Google Maps.

Column entry	Explanation
<b>ID</b>	This column shows the index of the measurement. This is automatically done by <i>Python Pandas</i> .
<b>First(EuropeanVehicleCategory)</b>	This column shows the European Vehicle Category in which the passing vehicle is categorized. The European categorization of vehicles is discussed on the next page.
<b>First(EuropeanVehicleType)</b>	This column shows the vehicle type of the passing vehicle (in Dutch). This is closely related to the European Vehicle Categorization and is also explained on the next page.
<b>First(KerbMass)</b>	This column shows the kerb mass in kilograms of the passing vehicle. The kerb mass is defined <sup>6</sup> as the mass of an empty vehicle, plus the mass of a fuel tank filled for 90% (25 kg) and the mass of the driver (75 kg).
<b>First(MaxAuthMassOfVehicle)</b>	This column shows the maximum allowed mass of the vehicle in kilograms as stated by the Dutch legislation.
<b>First(TechnicalMaxMassVehicle)</b>	This column shows the maximum allowed mass of the vehicle in kilograms as stated by the manufacturer of the vehicle. This value can be higher than the value stated by law.
<b>First(WheelBase)</b>	This column shows the total wheelbase of a vehicle in centimeters. For 2-axled this is trivial. For 3+ - axled vehicles, this value represents the distance between the two most outer axles.
<b>First(CamId)</b>	This column shows the camera ID that registered the passing vehicle. The value 2 is given for a vehicle driving on the slow lane. The value 1 is given for a vehicle driving on the fast lane.
<b>First(LocationCode)</b>	This column shows the location code used by the municipality of Rotterdam. This is of no further interest for the study in itself.
<b>First(LocationName)</b>	This column shows the location name that is coupled with the location code in the previous column. This value also isn't of real interest for the study in itself.
<b>First*(VehicleLength)</b>	This column shows the total vehicle length of the passing vehicle in centimeters.
<b>Date</b>	This column shows the date of the registered vehicle as MM/DD/YYYY.
<b>Time</b>	This column shows the time of the registered vehicle in a 12-hour convention.

Table 3 - Explanation of column entries in RAW data given by camera system.

The obtained information is stored in an CSV file. Due to the restricted workability of MS Excel, the programming will take place in Python. Since a lot of visual aspects are expected to be shown and used, the user-friendly Jupyter Notebook will be opted for to program in. All codes are given in appendix H.

## 4.2 Data of interest

### 4.2.1 Filtering criteria

In previously done studies by Hellebrandt [1], TNO [13], it turned out that it is very important to have axle loads taken from WIM data to gain more insight in recreating traffic loads. Also, it turned out that

<sup>6</sup> <https://www.rdw.nl/particulier/voertuigen/auto/het-kentekenbewijs/over-het-kentekenbewijsdocument/gegeven-massa-rijklaar>

passenger vehicles are not of interest when assessing an existing bridge, hence all passenger vehicles are filtered out.

In studies where WIM data is the source, the axle loads were known, but the type of vehicle was unknown. Therefore, based on these axle loads, it was guessed which type of vehicle was passing through. In this case it is the other way around; the type of vehicle is known (presumably with the number of axles), but the actual axle loads itself are unknown. Hence, a similar approach is used as shown in TNO [11]. Only N3- and O4-vehicles are filtered. Vehicles falling in the M2/M3- category are not well represented in the data. In the following paragraph the classification system of N3-vehicles will be discussed.

#### 4.2.2 Number of axles

As is shown in fig 3, the system does not give the number of axles that a passing vehicle has. As stated in [1], and shown by previous studies, the specific axle load is of interest for recreating traffic loading based on WIM measurements. However, in this case there is no specified axle load, but rather a theoretical maximum allowed axle load.

For M1 -vehicles, usually the number of axles is 2. For M2- or M3-vehicles (buses and coaches), the number of axles can range from 2 to 4. For N2- and N3-vehicles, the number of axles can range from 2 to 5. For O4-vehicles the number of axles can range from 1 up to more than 5.

#### 4.2.3 Vehicle type and classification

As shown in fig 3, currently the system does not give the property *vehicle type* to the current measured entry. As was shown by TNO [13] and Hellebrandt [1], the vehicle type of N3-vehicles is rather important for constructing traffic load models. Another study done by TNO [12] with the main goal to recalibrate traffic loading based on WIM data, shows that the population of heavy vehicles (including M2- and M3-vehicles) can be divided up in 59 sub-classes. This is shown in appendix B.

It must be noted that these vehicle classes are based on WIM data, and not necessarily the data used in this research. Firstly, the five-vehicle classification done by TNO [13] is applied in this research. Whenever it turns out that a more comprehensive classification is needed, this will be adopted in the approaching methods.

So, the following classes within N3-vehicles are adopted:

- Semi-truck: 2 axles
- Semi-truck: 3 axles
- Trucks: 2 axles
- Trucks: 3 axles
- Tipper trucks / cement trucks etc.

The system also does not give information about the vehicle type for O4-vehicles. For O4-vehicles there are also a few classifications to be done. As shown by the publicly accessible database by RDW. In total three main vehicle types of O4-vehicles exist:

- Semi-trailers
- A-frame drawbar trailer
- Close-coupled trailer

Since the aim of this research is to try and develop traffic load models based on car license registrations, the number of axles, the individual axle load and the gross vehicle weight of vehicles must be known. To calculate load effects caused by vehicles, these three properties are important, as

well as the interspatial axle distancing. Therefore, the vehicle type for N3- and O4-vehicles must be known. In the following subparagraphs a strategy to classify N3- and O4- vehicles is shown.

#### 4.2.3.1 Classification of N3-vehicles

As stated before, due to privacy reasons, the automated registering system does not directly give a license plate as output. However, certain vehicle characteristics are registered, shown in paragraph 4.1.3, making it possible to retroactively couple registered vehicles to a known existing vehicle. This requires two databases: one with measurements characteristics (MEA), and one with existing characteristics (RDW). The database containing known characteristics of vehicles is freely accessible through the website of the RDW. When removing some unnecessary columns, the following RDW-database is created, containing all registered N3-vehicles in the Netherlands as of January 2020.

	LP	AX1	AX2	AX3	AX4	AX5	AX6	AX7	AX8	AX9	GVW	LENGTH	WHEELBASE	GVW COMB	TYPE	KERB MASS	AXLES
0	BE7852	7500	10652.0	6548.0	0.0	0.0	0.0	0.0	0.0	0.0	21500.0	740.0	512.0	21500.0	opleggertrekker	8310	3
1	BZSX88	5600	10400.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	14500.0	740.0	386.0	23000.0	brandweerwagen	9585	2
2	76BNX1	9000	8000.0	9500.0	9500.0	8000.0	0.0	0.0	0.0	0.0	45000.0	850.0	641.0	50000.0	opleggertrekker	15800	5
3	89BFK3	8000	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19500.0	616.0	380.0	50000.0	opleggertrekker	8221	2
4	BXPR63	9000	11500.0	7500.0	0.0	0.0	0.0	0.0	0.0	0.0	28000.0	815.0	585.0	0.0	voertuig met haakarm	10790	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
162486	BXRS79	9000	11500.0	7500.0	0.0	0.0	0.0	0.0	0.0	0.0	28000.0	1075.0	620.0	0.0	open wagen	16690	3
162487	BDGR96	7100	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	18600.0	0.0	350.0	40000.0	opleggertrekker	5983	2
162488	BXLF57	7500	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19000.0	617.0	380.0	40000.0	opleggertrekker	6749	2
162489	BZSP08	9000	12000.0	12000.0	0.0	0.0	0.0	0.0	0.0	0.0	33000.0	1010.0	590.0	36495.0	speciale groep	32600	3
162490	00BFV6	6000	10500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	16000.0	735.0	400.0	0.0	brandweerwagen	9180	2

Figure 4 - Excerpt of RDW database. Weights in kg, distances in cm.<sup>7</sup>

Now it is possible to classify each measured vehicle, based on the following four mutual known conditions. For each measured vehicle, if all four characteristics match with one of the vehicles in the RDW-database a match is made, and a license plate is given as output. The characteristics are shown below.

Characteristic	MEA	RDW
Gross vehicle weight	First(MaxAuthMassOfVehicle)	GVW
Kerb Mass	First(KerbMass)	KERB MASS
Wheelbase	First(WheelBase)	WHEELBASE
Length	First*(VehicleLenght)	LENGTH

Table 4 - Explanation of parameters and characteristics

For understanding purposes, in the picture below a flow-chart is shown to clarify the piece of code shown next to it.

<sup>7</sup> Source: <https://opendata.rdw.nl/browse>



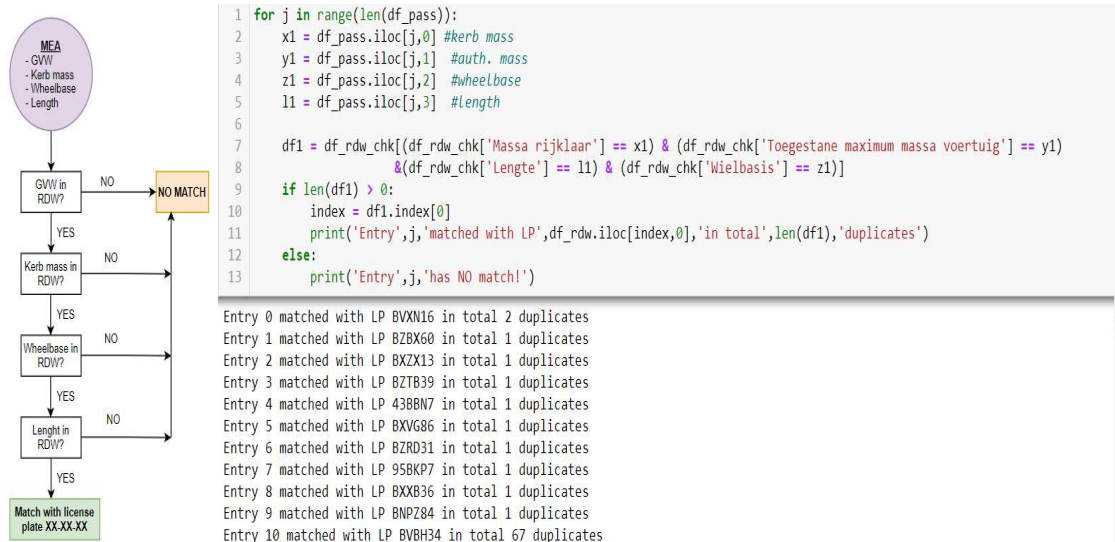


Figure 5 - Flow chart for coupling license plates and excerpt of script

This procedure was applied for measurements regarding location 4400-C. In total 35.236 entries could be coupled license plates. 997 entries were not able to be coupled to an existing vehicle. The piechart below shows the occurring vehicle classes as used by the RDW-database.

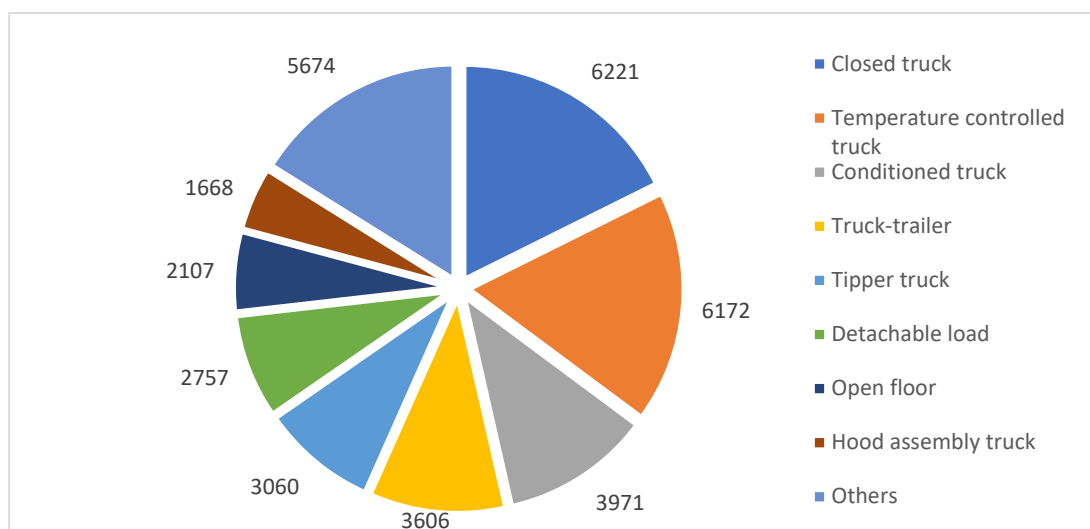


Figure 6 - Pie chart of registered N3 vehicles classes for location 4400-C.

As can be seen from the pie-chart, the majority of the passing trucks were able to be classified. The accompanying license plates of the entries are now coupled to their data regarding the measurement, resulting in the following database, shown in *Figure*. The added column *TRAILER* shows if a truck is allowed to carry additional weight through any kind of trailer.

Date	Time	LP	AX1	AX2	AX3	AX4	AX5	AX6	AX7	AX8	AX9	GVW	LENGTH	WHEELBASE	GVW COMB	TYPE	KERB MASS	TRAILER	A
1/1/2019	11:42:57 PM		8000	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19500.0	1097.0	600.0	0.0	gecond. met temperatuurreg.	11870	0.0	
1/1/2019	1:49:55 PM		9000	11500.0	7500.0	0.0	0.0	0.0	0.0	0.0	0.0	28000.0	1100.0	697.0	0.0	gecond. met temperatuurreg.	13840	0.0	
1/1/2019	5:32:06 PM		10000	10000.0	10000.0	9500.0	9500.0	0.0	0.0	0.0	0.0	49000.0	933.0	695.0	50000.0	afneembare bovenbouw	15870	1000.0	
1/10/2019	8:22:44 AM		10000	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	21500.0	825.0	530.0	26500.0	gesloten opbouw	18831	5000.0	
1/10/2019	4:41:49 PM		9000	11500.0	7500.0	0.0	0.0	0.0	0.0	0.0	0.0	28000.0	1023.0	670.0	0.0	open laadvloer	14045	0.0	
...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9/9/2019	5:25:38 PM		7500	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19000.0	965.0	578.0	0.0	gecond. met temperatuurreg.	9855	0.0	

Figure 6 - Database for location 4400-C with only N3 vehicles classified. Weights in kg, distances in cm.

#### 4.2.3.2 Classification of O4-vehicles

The same procedure for N3-vehicles can also be applied for O4-vehicles. A similar RDW-database for trailers can be constructed from the RDW-website as well. This results in the following RDW-database, shown below.

	LP	AX1	AX2	AX3	AX4	AX5	AX6	AX7	AX8	AX9	TYPE	GVW	LENGTH	WHEELBASE	KERB MASS
0	OP24KP	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	39000.0	0.0	725.0	5000.0
1	OR02JG	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	41000.0	1386.0	879.0	6339.0
2	WR27SH	10000	10000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Aanhangwagen	20000.0	0.0	452.0	4960.0
3	OH91TS	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	39000.0	1389.0	911.0	6600.0
4	ON35BK	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	42000.0	1404.0	876.0	11690.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
181541	ON52GF	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	42000.0	1400.0	891.0	8488.0
181542	OP41VS	9000	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	30000.0	1382.0	896.0	8270.0
181543	ON40BB	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	39000.0	1389.0	901.0	6709.0
181544	OR66DK	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	39000.0	1389.0	901.0	7104.0
181545	OH85SP	9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	39000.0	1390.0	902.0	7200.0

181546 rows × 15 columns

Figure 7 - Excerpt of RDW database for all registered trailers (O4).

In this RDW-database all trailers registered in the Netherlands as of January 2020 are given. It must be noted that some trailers have a total length of 0 cm. The explanation for this is that some trailers (or vehicles in general) have information stored in separated columns, not directly available through the RDW website, in which particularities are given. For example, the first trailer can be part of a larger truck-trailer combination and thus no length is given. The wheelbase however is usually given.

For the O4-vehicles, the exact same procedure as for N3-vehicles is followed. However, only value for kerb mass is not checked since the current system does not register kerb masses of O4-vehicles. For location 4400-C, in 2019 a total of 54.210 O4-vehicles were registered, of which 48.244 could be recognized and coupled to an existing vehicle. The trailer population is shown in the pie chart below.



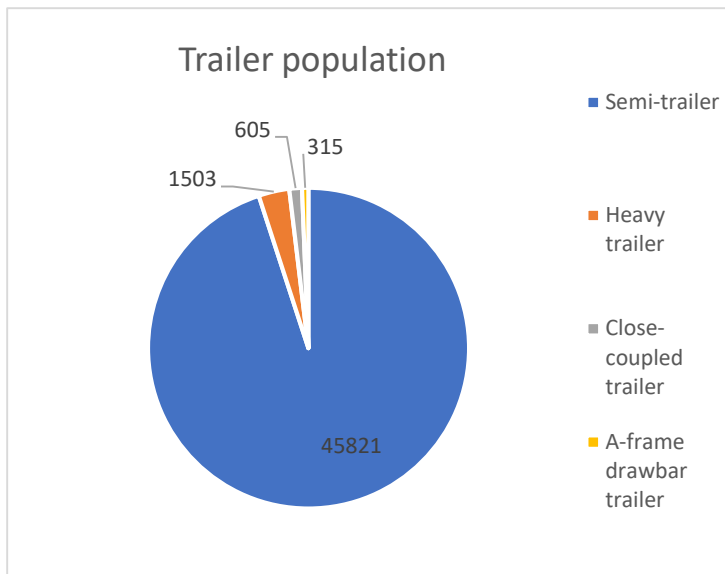


Figure 8 - Pie chart of registered O4 vehicle classes for location 4400-C.

As can be clearly seen, the overall majority (>95 %) of the O4-vehicles can be classified as semi-trailers. Finally, the license plates are coupled to the measured entries in the MEA-database. This results in the following MEA-database for O4-Vehicles.

Date Time	Lane	LP	AX1	AX2	AX3	AX4	AX5	AX6	AX7	AX8	AX9	TYPE	GVW	LENGTH	WHEELBASE	KERB MASS
1/1/2019 11:04:45 PM	2		9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	39000.0	0.0	881.0	4560.0
1/1/2019 12:35:35 PM	2		9000	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	30000.0	1283.0	771.0	9260.0
1/1/2019 2:32:33 PM	2		9000	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	33000.0	1280.0	769.0	9180.0
1/1/2019 3:03:50 PM	2		9000	9000.0	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	41000.0	1363.0	915.0	9780.0
1/1/2019 3:12:27 PM	2		9000	9000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Oplegger	30000.0	1281.0	821.0	8700.0

Figure 9 - Database for location 4400-C with only O4 vehicles classified. Weights in kg, distances in cm.

## 5 Processing data

### 5.1 Creating load samples

#### 5.1.1 Fundamental assumptions

To create load samples and to arrange certain load samples, a few fundamental assumptions and rules must be made at first hand. These assumptions are shown below.

- Any registered vehicle is assumed to be fully loaded, i.e. the maximum allowed axle load is always present.<sup>8</sup>
- If any N3-vehicle is allowed to carry a trailer of any kind, it is assumed it is always carrying a trailer.<sup>9</sup>
- It is assumed that the initial value of the individual axle loads remains constant while causing a load effect. No dynamic amplification factor is considered.
- Whenever the axle configuration of any type of vehicle is not known, an assumption is made. Each vehicle type is further elaborated in paragraph 5.1.3.
- Based on the previous point, whenever an axle of a vehicle is not present on the bridge anymore, it immediately does not cause any load effect on the bridge deck.

#### 5.1.2 Adopted strategy

Considering the fundamental assumptions stated in the previous paragraph, the following strategy is adopted:

- The registered N3 vehicles act as the starting input.
- The information given by the RDW system is automatically coupled to the registered vehicle.
- If a N3 vehicle is allowed to carry a trailer, it will be assumed that it carries a trailer. The method for this is further elaborated in paragraph 5.1.3. If it carries a trailer, the N3-O4 combination is taken as one single load sample. The values for the axle loads are given by the RDW data.
- For interspatial axle distances, per vehicle type an assumption is made. This is further elaborated in paragraph 5.1.3

#### 5.1.3 Coupling N3-O4 vehicles

One could easily imagine that whenever a truck-trailer combination is measured, both the truck and the trailer should be registered. However, as it turned out, the MEA-database does not show any chronologic correlation between registered N3- and O4 vehicles. To account for this a new method is adopted, where a theoretical N3-O4 combination is made based on data registered by the camera system. Both N3- and O4-vehicles are registered and logically the recorded amount of both categories should be the same. However, apparently more O4-vehicles (54.210) have been registered than N3-vehicles (36.390) in the year of 2019. So, the minimum amount of N3-O4 combinations that have passed should be at least 54.210.

So, to simulate a N3-O4 combination, whenever the gross vehicle combination (depicted as *GVW COMB* in the figure below), is greater than the kerb mass itself, a theoretical O4-vehicle is attached of the resulting GVW. So, in the figure below, entry 52 would be coupled to an O4-vehicle with a GVW of 13.000 kg at most.

---

<sup>8</sup> This assumption is conservative, since the cargo of trucks differ throughout their sort. Tipper trucks carrying concrete are heavier than tipper trucks carrying styrofoam.

<sup>9</sup> This is a conservative assumption since trucks do not always carry trailers, especially semi-trucks.

Lane	Date	Time	LP	AX1	AX2	AX3	AX4	AX5	AX6	AX7	AX8	AX9	GVW	LENGTH	WHEELBASE	GVW COMB	TYPE	KERB MASS
50	2	1/10/2019	12:02:14 PM		7500	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	19000.0	970.0	550.0	0.0	gesloten opbouw	8840
51	2	1/10/2019	9:18:39 AM		5800	9440.0	0.0	0.0	0.0	0.0	0.0	0.0	14000.0	995.0	562.0	17500.0	gesloten opbouw	7245
52	2	1/10/2019	4:47:22 PM		6100	10500.0	0.0	0.0	0.0	0.0	0.0	0.0	15000.0	858.0	476.0	28000.0	gecond. met temperatuurreg.	9220
53	1	1/10/2019	11:16:21 AM		7500	11500.0	0.0	0.0	0.0	0.0	0.0	0.0	19000.0	970.0	550.0	49080.0	afneembare bovenbouw	8461

Figure 10 - Database of registered N3-vehicles for location 4400-C.

As stated in 8.2.3, only 5 classes of N3-vehicle types are adopted. For each of these different vehicle types a different approach must be used to transform the registered vehicle(s) to a representative load model. In the following subparagraphs this is done for each load model. For some vehicle types, certain parameters considering technical information about the vehicle is used. These are listed below.

Notation	Explanation	Unit
$w_{rnd}$	Randomized interspatial axle distance drawn from a normal distribution with the values given for each vehicle type individually.	m
$l_{O4}$	Total vehicle length of the trailer	m
$w_{O4}$	Total wheelbase of the trailer	m
$w_i$	The $i$ -th interspatial axle distance.	m
$w_{tot}$	The total wheelbase of the vehicle (combination)	m

Table 5 - Clarification of used parameters

#### 5.1.3.1 Axial distances

Interspatial axle distances have a large contribution in the occurring load effect from any vehicle. Hence it is important to note that for 3- or more axled vehicles these interspatial distances are assumed per vehicle type. This is individually described in each subparagraph. To be clear: the numbering of the axles is done based on the direction of travel. So, the *first* axle is the axle that comes first.

#### 5.1.3.2 Semi-trucks

For all semi-trucks, basically the same method is assumed. However, there are some slight differences between the 2- and 3-axled semi-trucks. Considering the semi-trailers, they are assumed all to be the same type: a group of axles located at the back of the trailer. The number of axles can span from 1 to 4. The figure below shows an example of a semi-trailer.



Figure 11 - Example of a 2-axled semi-trailer.<sup>10</sup>

##### 5.1.3.2.1 2-axled

For 2-axled semi-trucks the wheelbase of the truck itself is known. The individual axle weights are also known. Whenever a 2-axled semi-truck is registered, it is coupled to a semi-trailer with the condition stated in 5.1.3. For the semi-trailer, the following assumptions are made:

<sup>10</sup> Source: <https://www.shutterstock.com/g/contributor/554422?searchterm=semi-trailer>

- The wheelbase of the O4-vehicle is the distance between the *last* axle of the O4-vehicle and the *last* axle of the N3-vehicle.
- Any other interspatial axle distances  $w_{rnd}$  are generated randomly between the range of 1,25 and 1,35 m.

#### 5.1.3.2.2 3-axled

For 3-axled semi-trucks the same principal applies as for the 2-axled semi-trucks. However, the following assumptions are made:

- The wheelbase of the N3-vehicle is the distance between the *last* axle of the N3-vehicle and the *first* axle.
- The interspatial axle distance  $w_{rnd}$  between the *last* and the *second* axle is a number generated between 1,25m and 1,35m.

For semi-trailers, the same assumptions are used as for the 2-axled semi-trucks.

#### 5.1.3.3 Full trucks

For full trucks, a similar approach is used for the semi-trucks. In terms of a load model, a full truck is basically a semi-truck with a larger interspatial axle distance. This is true for both 2-axled and 3-axled full trucks. Considering the trailers however, a noticeable difference occurs, since full trucks can carry both close-coupled trailers (CCT) and A-frame-drawbar trailers (ADFT). Both these trailers have quite a different axial spacing. CCT have a group of closely spaced axles, usually around halfway the vehicle. ADFT have (a group of) axles located at both ends of the trailer. This is shown in the figure below.



Figure 12 - Pictures of different trailer types which can be carried by full trucks. Left: ADFT. Right: CCT.<sup>11</sup>

#### 5.1.3.3.1 2-axled

For 2-axled full trucks, both the wheelbase and the individual axle loads are known. If a registered full-truck is allowed to carry a trailer, a distinction is made between CCT- and ADFT-trailers, as described in 5.1.3.3. The following assumptions for CCT-trailers are made:

- The interspatial axle distance of the CCT is randomly taken between 1,25m and 1,35m for all axles.
- The distance between the *first* axle of the O4-vehicle and the *last* axle of the N3-vehicle is taken as  $\frac{(l_{O4}-w_{O4})}{2} + w_{rnd}$  where  $w_{rnd}$  is randomly taken between 1,25m and 1,35m.

For ADFT-trailers the following assumptions are made:

- The interspatial axle distance of the ADFT is randomly taken between 1,25m and 1,35m for all axles.

<sup>11</sup> Sources: <https://bit.ly/3huQJwP>, <https://bit.ly/3fru1nE>

- The distance between the *first* axle of the O4-vehicle and the *last* axle of the N3-vehicle is taken as  $\frac{(l_{O4}-w_{O4})}{2} + w_{rnd}$  where  $w_{rnd}$  is randomly taken between 1,25m and 1,35m.

#### 5.1.3.3.2 3-axled

For 3-axled full trucks, the interspatial axle distance is not known between the *last* and the *second* axle. Hence, the same assumptions are adopted as stated in paragraph 5.1.3.3. Just like their 2-axled counterpart, 3-axled full trucks can carry either CCT-or ADFT-trailers. The following assumptions are made:

- The wheelbase of the N3-vehicle is the distance between the *last* axle of the N3-vehicle and the *first* axle.
- The interspatial axle distance  $w_{rnd}$  between the *last* and the *second* axle is a number generated between 1,25m and 1,35m.

For CCT- and ADFT-trailers the same assumptions are made for 2-axled full trucks carrying trailers.

#### 5.1.3.4 Tipper trucks

Whenever a tipper truck is registered, the composition of the axle loads is very dependent on the number of axles the corresponding truck has. The number of axles for a tipper truck can range from 2 to 5. For each individual axle, the accompanying axle loads are known. For each number of axles, the following assumptions are made:

- For 2-axled trucks: the wheelbase is the distance between the *first* and the *last* axle.
- For 3-axled trucks: the wheelbase is the distance between the *first* and the *last* axle. For the distance between the *last* and the *second* axle a randomly number generated between 1,25m and 1,35m is applied.
- For 4- or more axled trucks: all interspatial axle distances are assumed to be the total wheelbase divided by the number of axles  $n$  minus one, i.e.  $w_i = \frac{w_{tot}}{n-1}$ .

It must be noted that most tipper trucks are not allowed to carry any trailers. Hence, in this method a tipper truck will not carry any trailer.

#### 5.1.3.5 Mobile cranes

A great portion of the heaviest recorded vehicles are mobile cranes; hence they are classified individually. The axle loads can go as high as 12.000 kg with a GVW COMB of 85t with a relatively small total length, and thus relatively small interspatial axle distances as well. The number of axles for mobile cranes can range from 2 to 6. For each axle, the individual axle loads are known. For the interspatial axle distances the following assumptions are made:

- For 2-axled cranes: the wheelbase is the distance between the *first* and the *last* axle.
- For 3-axled cranes: the same assumption as for 3-axled semi-trucks is made, however the variable  $w_{rnd}$  can range from 1,65m to 1,75m.
- For 4- or more axled cranes, it is assumed that the axles are evenly distributed along the total wheelbase, i.e.  $w_i = \frac{w_{tot}}{n-1}$

In some cases, a mobile crane is allowed to carry a trailer containing its contra weight. For the heaviest cranes, the gross vehicle weight of this trailer can be as high as 27t. To account for this, a theoretical (existing) trailer is constructed considering the following assumptions:

- A trailer of three axles is attached to the mobile crane
- Each individual axle load is 8.000 kg

- The interspatial axle distance between the *first* axle of the trailer and the *last* axle of the mobile crane is taken as 2.8 m.
- The interspatial axle distance between the other axles of the trailers is assumed to be 1.95m.

### 5.1.3.6 Extreme heavy combinations

For this category, the registered O4-vehicles are governing. Some registered O4-vehicles can have a GVW COMB of more than 100t and thus become governing for constructing load models. An *extreme heavy combination* (EHC) is classified as whenever a O4-vehicle with a GVW > 70t is registered. For the constructing of the load models, the following assumptions are made:

- The N3-vehicle in an EHC is always a 4-axled semi-truck, with known individual axle loads. The total wheelbase is the wheelbase between the *first* and the *last* axle of the truck. For the *fourth-third* and *third-second* axles the interspatial axle distances  $w_{rnd}$  can range from 1,25m to 1,35m.
- For 5-axled trailers: it is assumed that the axles are evenly distributed along the total wheelbase, i.e.  $w_i = \frac{w_{tot}}{n-1}$
- For 6-axled trailers: it is assumed that 6 axles are divided among two groups of three closely spaced axles. Where the total wheelbase is the distance between the *last* axle and the *first* axle. All other interspatial axle distances  $w_{rnd}$  can range from 1,25m to 1,35m.
- For 7-axled trailers: the same assumption is adopted for 5-axled trailers.
- For 8-axled trailers: it is assumed that the axles are divided among two groups of axles. One group considering three closely spaced axles at the front of the trailer. The second group considering five closely spaced axles at the back of the trailer. The total wheelbase is the distance between the *last* axle and the *first* axle. For all interspatial axle distances  $w_{rnd}$  can range from 1,25m to 1,35m.
- For 9-axled trailers: the same assumption as for 8-axled trailers is adopted, however the second group of axles at the back consist of 6 axles instead of 5.

## 5.2 Calculating load effects

### 5.2.1 Approach

For simplicity, a bridge is modelled as a simply supported beam. For now, only the bending moments are calculated. To calculate the bending moments, the total length of the bridge as well as the total length of the passing vehicle is of importance, since it is possible that all axles of a single vehicle don't cause a load effect at the same time.

To account for this, a simple but rather effective algorithm is written, shown in figure 14.  $Y(L)$  is written, where the variable  $L$  is used for bridge length input. The basic principle of this function  $Y(L)$  is to determine the location of each individual axle on the bridge relative to the *first* axle of the vehicle(-combination). In simple terms, the movement of a vehicle across a bridge is simulated.

```
def Y(L): # j = Len(df)
    x = np.linspace(0,L+25,((L+25)*10)+1) #add fictional length to account for all axles
    y = np.zeros(len(x)) #create an empty array
    y1 = x # y1 = x
    y2 = y1 - df.iloc[j,18] # y2 = y1 - w1. Needed to start the for-Loop.
    y = np.vstack((y1,y2))
    for r in range(16): #Fill the y-array with all known w_i. Ends on column 35
        c = y[r+1] - df.iloc[j,r+19]
        y = np.vstack((y,c))
    y = np.where(y<0,0,y) #Delete negative values for y
    y = np.where(y>L,0,y) #Delete values for y > L.
    return y
```

Figure 13 - Excerpt of python code of writing function  $Y(L)$





```

df_adds = df.iloc[:,36:46]
df = df.iloc[:,0:36]
McMax = np.zeros(len(df))
F_matrix = (df_fw.iloc[:, : 18]*(10/1000)).to_numpy() #Selecting all axle loads. kg --> kN
for j in range(len(df)): #Len (df)
    print(j)
    y_matrix = Y(L) #y1 = y_matrix[1]

    Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
    Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

    #Av
    for r in range(18): #In total 18 axles, so 18 reaction forces.
        Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
    #Mc
    for r in range(18):
        Mc[r] = np.where(y_matrix[r] < (L/2),
                        (Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
                        Av[r]*(L/2))
        Mc[r] = np.where(Mc[r]<0,0,Mc[r]) #Replace hogging bending moments with 0.

    Mc = Mc.sum(axis=0) #Sum all values column wise
    Mc = np.amax(Mc)
    McMax[j] = Mc
df = pd.concat([df_fw,df_adds], axis =1)

```

Figure 15 - Excerpt of python code to determine maximum load effect

In this method, each axle is assumed to load the bridge individually and that each axle is a point load. For each passing vehicle, for each individual axle, for each individual segment the bending moments are calculated are for all segments along the bridge and are stored in a separate matrix. Then, hogging bending moments are set to 0, as they cannot occur in simply supported bridges. For the 2-axled semi-truck carrying a single-axled trailer, the following results are obtained.

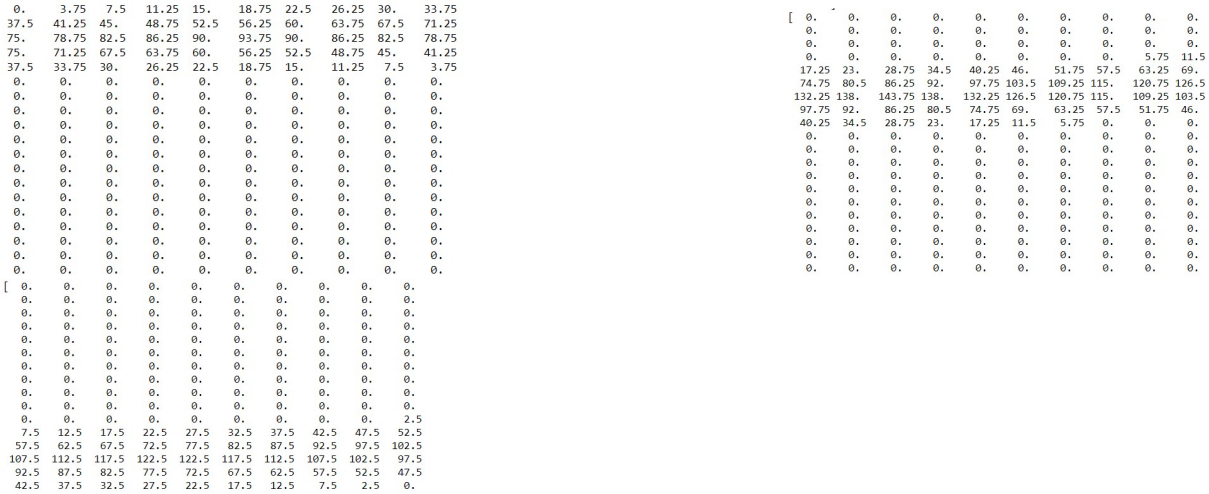


Figure 16 - Matrices containing calculated load effects

Finally, for all segments the bending moments are summed column wise, resulting in a single, maximal value for each segment. This is shown in the figure below. It can be clearly seen that firstly, only the *first* axle of the semi-truck loads the structure. Then, at segment  $x + w_1$  the *second* axle starts to load the structure as well. Then, due to the long interspatial distance between the *second* axle of the semi-truck and the *first* axle of the trailer, the bridge is not loaded at all. This is depicted by the 0's in the vector. Finally, the only axle of the trailer starts to load the structure again, but now solely the trailer.

The maximal value in this vector is the maximum occurring bending moment corresponding with the registered vehicle and is added to the database.



[	0.	3.75	7.5	11.25	15.	18.75	22.5	26.25	30.	33.75
37.5	41.25	45.	48.75	52.5	56.25	60.	63.75	67.5	71.25	
75.	78.75	82.5	86.25	90.	93.75	90.	86.25	82.5	78.75	
75.	71.25	67.5	63.75	60.	56.25	52.5	48.75	50.75	52.75	
54.75	56.75	58.75	60.75	62.75	64.75	66.75	68.75	70.75	72.75	
74.75	80.5	86.25	92.	97.75	103.5	109.25	115.	120.75	126.5	
132.25	138.	143.75	138.	132.25	126.5	120.75	115.	109.25	103.5	
97.75	92.	86.25	80.5	74.75	69.	63.25	57.5	51.75	46.	
40.25	34.5	28.75	23.	17.25	11.5	5.75	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	0.	0.	2.5	
7.5	12.5	17.5	22.5	27.5	32.5	37.5	42.5	47.5	52.5	
57.5	62.5	67.5	72.5	77.5	82.5	87.5	92.5	97.5	102.5	
107.5	112.5	117.5	122.5	122.5	117.5	112.5	107.5	102.5	97.5	
92.5	87.5	82.5	77.5	72.5	67.5	62.5	57.5	52.5	47.5	
42.5	37.5	32.5	27.5	22.5	17.5	12.5	7.5	2.5	0.	

Figure 17 - Matrix containing all maximum occurring load effect for each location. The 0's indicate the supports.

## 5.2.2 Processing

The approach described in the previous paragraph is now adopted for all registered vehicles. The strength of this method is, is that for each arbitrary vehicle combination bending moments can be calculated. During processing and debugging of the algorithm, certain tricks were implemented in this system resulting in a much faster code. At this point, it is very likely that the code is not yet optimized, but with a mid-range laptop with a i5-4310 2.00 GHz quadcore CPU and 8,00 GB RAM running the entire algorithm, from coupling license plates to calculating bending moments, takes about 15 minutes.

## 5.2.3 Results

### 5.2.3.1 Load effect

For this specific location in Rotterdam, a total of 31438 heavy vehicles were recorded and loaded the bridge in the following manner. Considering all approaches and assumptions made above, the following intermediate results are obtained, shown in the scatterplot below.

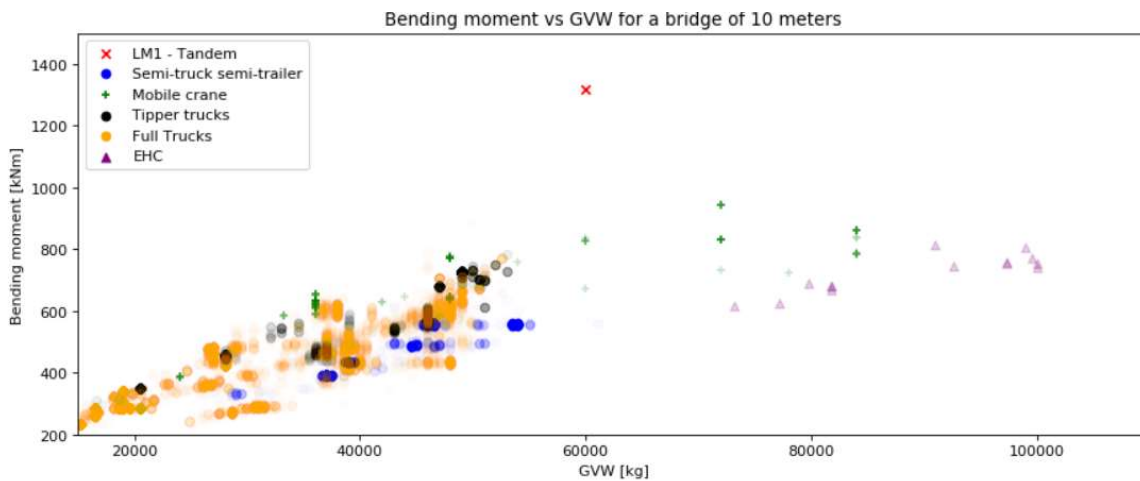


Figure 18 - Bending moment vs GVW for a simply supported bridge with a span length of 10 m.

Figure 19 shows a scatterplot of the occurring bending moments plotted against the GVW. The category mobile cranes exert the highest bending moments in general. LM1 of Eurocode 2 is indicated with the red cross. This value only considers the tandem load model of LM1, and not the uniformly distributed load. As can be seen, the majority of the occurring load effects does not exceed the value of 800 kNm. This is shown in detail in the histogram plot below.

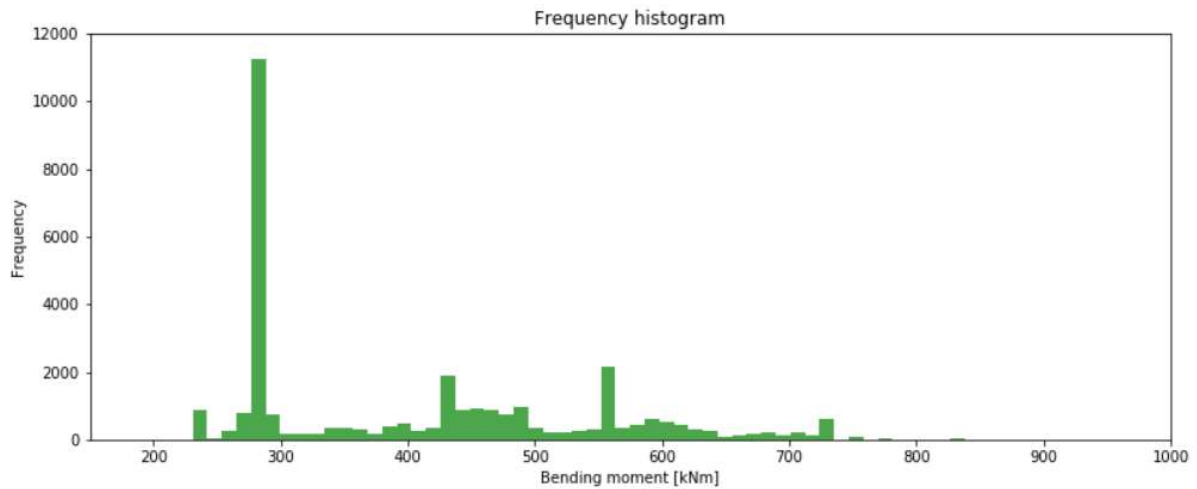
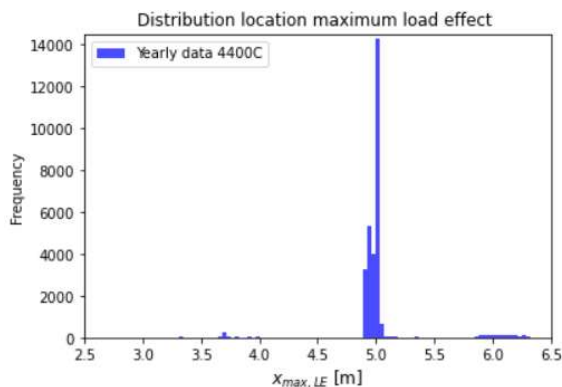


Figure 19 - Frequency histogram of occurring bending moments.

### 5.2.3.2 Location of maximum load effect

The figures above do not give information about the location of the maximum occurring load effect. The maximum occurring load effect however does not necessarily need to be at midspan, since the span is loaded by multiple axles at the same. The figure below shows the distribution for the location of the maximum load effect, denoted with  $x_{max,LE}$  for the recorded vehicles crossing a simply supported bridge with a span length of 10 m.



Unity	Value
Mean	5,01
Median	5,0
Minimum	3,24
Maximum	6,91

Table 6 - Statistical values for frequency histogram for location of maximum occurring load effect.

Figure 20 - Frequency histogram for location of maximum occurring load effect

The distribution clearly shows that for most vehicles  $x_{max,LE}$  occurs at midspan, despite the different axle configurations for vehicles.

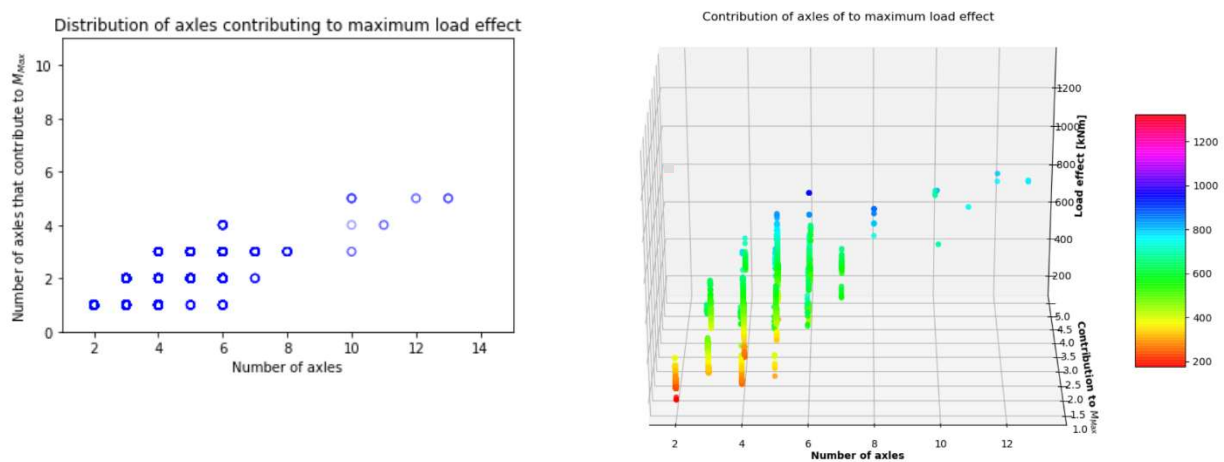
### 5.2.3.3 Contribution of number of axles to maximum load effect

Figure 19 showed that for the LP data the GVW solely is not governing for determining the highest load effect. Hence, the presumption raises that for each vehicle rather a group of (overloaded) axles contribute to the highest occurring load effect, rather than the GVW itself, as well as the total length of a vehicle, or rather the total wheelbase  $w_{tot}$ . This is investigated for the LP data, where for each vehicle the contribution of groups of axles is determined. This is based on the following principle:

1. For each vehicle, the maximum load effect  $M_{Max}$  is determined according to the approach written in paragraph 5.2 and the total number of axles are determined.
2. Then, all relative coordinates of the axles with respect to  $M_{Max}$  are determined.
3. The contribution of **each** axle i.e.  $M_{Axle,i}$  with respect to  $M_{Max}$  is determined and is stored as the ratio  $\eta_{F,i}$ .

4. For any  $n$  – axled vehicle, the threshold value is assumed to be  $\frac{1}{n}$  per axle;<sup>12</sup>
5. If the contribution of an axle is below its assumed  $\frac{1}{n}$  contribution, the axle does not count towards the total amount of axles contributing to the maximum load effect.
6. Finally, for each vehicle the number of axles that is responsible for the highest load effect and the contribution of each axle  $\eta_{F,i}$  to the maximum load effect are determined.

These steps give insight in how the different axles of a vehicle load exert load effect on a simply supported bridge. The results for a simply supported bridge with a span of 10 m loaded by the given LP data are shown below. As can be seen, the highest load effect per vehicle is primarily caused by 1 or 2 axles. For vehicles with more than 4 axles, the contribution of axles seems to shift more towards 2 and 3 axles exerting the load effect. These results show that not primarily the GVW or vehicles with a high number of axles result. Moreover, these results show that a relation exists between the maximum load effect, the individual axle load(s), and the interspatial axle distance.



<b>Contribution axles to maximum load effect</b>	<b>5</b>	0	0	0	0	0	0	0	0	0	0	4	0	2	3
	<b>4</b>	0	0	0	0	0	17	0	0	0	1	2	0	0	0
	<b>3</b>	0	0	0	578	4174	1124	47	17	0	2	0	0	0	0
	<b>2</b>	0	0	2512	4568	1915	360	7	0	0	0	0	0	0	0
	<b>1</b>	0	13792	1007	1305	16	29	0	0	0	0	0	0	0	0
		<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	
<b>Number of axles</b>															

Figure 21 - Compilation of 3 figures. Top left: relation between number of axles contributing to the maximum load effect and the total number of axles. Top right: Contribution of axles to the maximum load effect with added colour scale. Table: numerical values.

The composition of figures above shows the results. The top left figure indicates how many axles of a vehicle caused the highest load effect. The top right figure shows the caused load effect by a vehicle and how many axles contributed to this load effect. The table is a numerical representation of the top left figure. The sum of each column in the table is the number of vehicles recorded with those specific number of axles.

So, 13.792 2-axled vehicles, 3.519 3-axled vehicles, 6.451 4-axled vehicles and so on. The individual values in the table show how the maximum load effect per vehicle is constructed. For example,

<sup>12</sup> So, for 2-axled vehicles, each axle should contribute 50% to the total load effect. For 3-axled vehicle, each axle should contribute 33% etc.

considering the category of 3-axled vehicles. In this category, the maximum load effect was generated 1.007 times by only one axle, and 2.512 times by two axles.

The top right figure is basically the same figure as the top left one, only a colour scale was added to indicate the value of the load effect. This visualizes the range of load effect per recorded vehicle type, or rather n-axled vehicle type. The highest load effect, indicated with blue, is caused by a 6-axled vehicle.

### 5.3 Comparison with other data

#### 5.3.1 WIM data

As described before, for load effects caused by traffic loading, the two most influential parameters are the interspatial axle distance and the individual axle loads. The approach used in this study, assumes certain interspatial axle distances and axle configurations. It is therefore of interest to compare the assumptions made with real obtained values done by a WIM-system. In a study done by Hellebrandt [1] several years ago, the municipality of Rotterdam used WIM data to determine bridge loading due to traffic for the Beukelsbrug and Oost-Abtsbrug. This study used WIM data and interspatial axle distance data given by the WIM system. To validate the obtained results based on the license plates, a comparison has to be made to see whether similar results are obtained. If any deviations in the results are obtained, an explanation has to be found why.

To achieve this, the same algorithm as described in paragraph 5.2 is used but with input data from WIM for both bridges. The results are plotted in figures 23 and 24. Load effects based on license plates are indicated with LP. In both figures, the highest occurring load effect for LP-vehicles are due to the tandem load model from LM1. This value serves as a threshold.

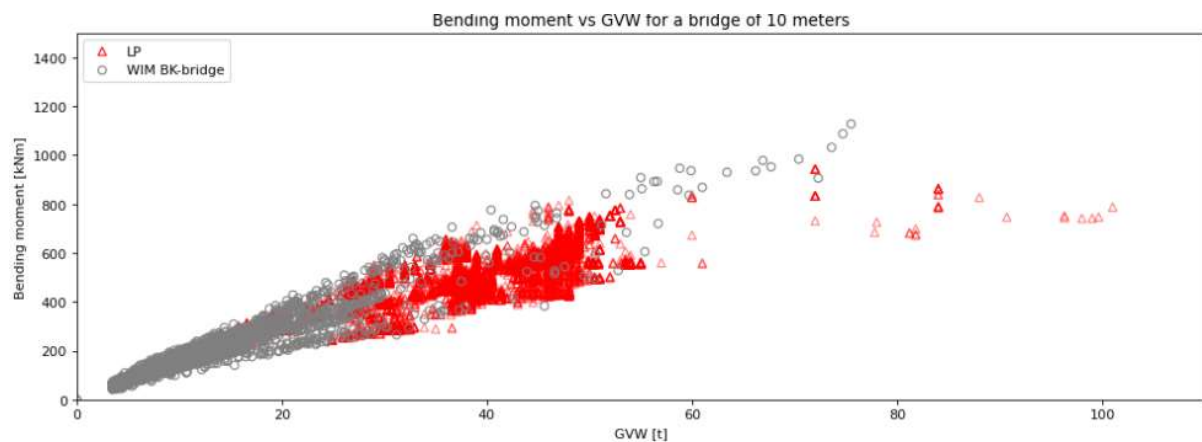


Figure 22 - Comparison between WIM data and LP data for measurement location *Beukelsebrug*.

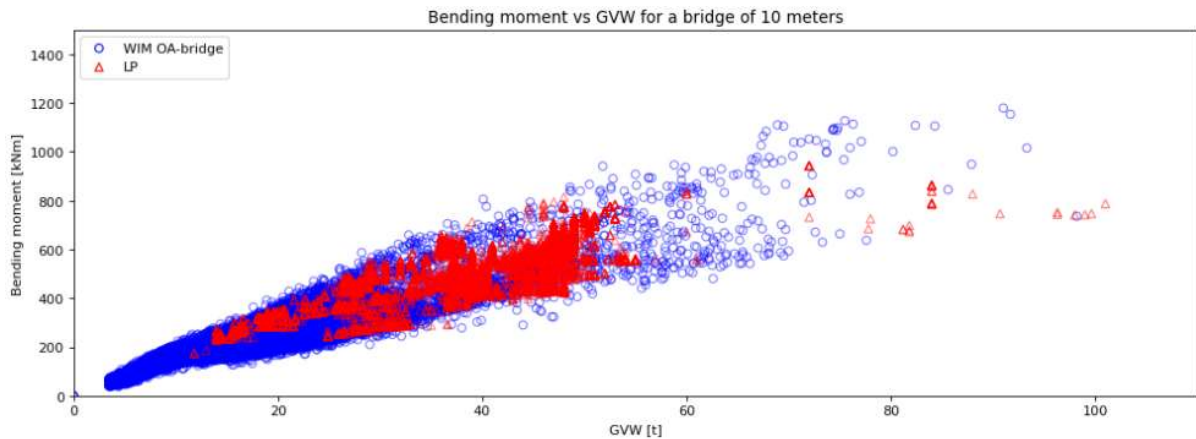


Figure 23 - Comparison between WIM data and LP data for measurement location *Oost-Abtsbrug*.

Both scatter plots clearly show a similar spread of the occurring load effects, especially for vehicles with a GVW up to 60t. For heavier vehicles, the spread becomes larger and the LP data is not representative anymore. Since heavier vehicles generate higher load effects, they are more of interest to compare.

When looked closer at the scatter plot for the *Beukelsebrug*, an interesting occurrence is found. A vehicle with 75t GVW from the WIM data exerts a load effect which is almost 1.5 times as high as the accompanying LP data vehicle of 75t. This is shown in the figure below.

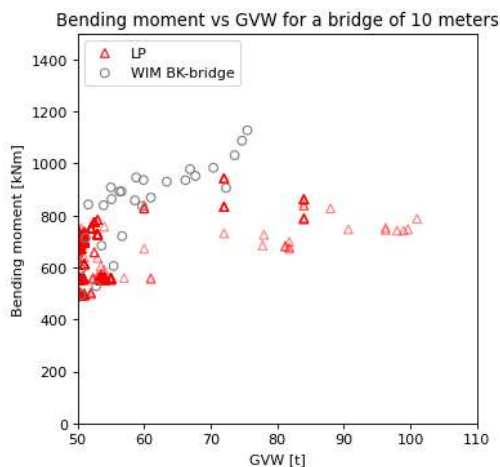


Figure 24 – Caused load effect by vehicles with a GVW higher than 50t

All this depends on the interspatial axle distances and the individual axle loads. The WIM data classifies vehicles in 20 different axle configurations, as shown in appendix B. To gain insight in the individual axle loads and the interspatial axle distances, for all 20 axle configurations a visualisation was made to gain insight in what ‘type’ of vehicle was crossing the bridge.

In the figure shown left below, the vehicle causing the highest load effect is shown schematically. The *first* axle of the vehicle is located at the far right. The legend shows the interspatial axle distances, where the first value is the distance between the first and second axle and so on. Clearly, the latter three axles of this vehicle are exceedingly high loaded, all greater 150 kN.

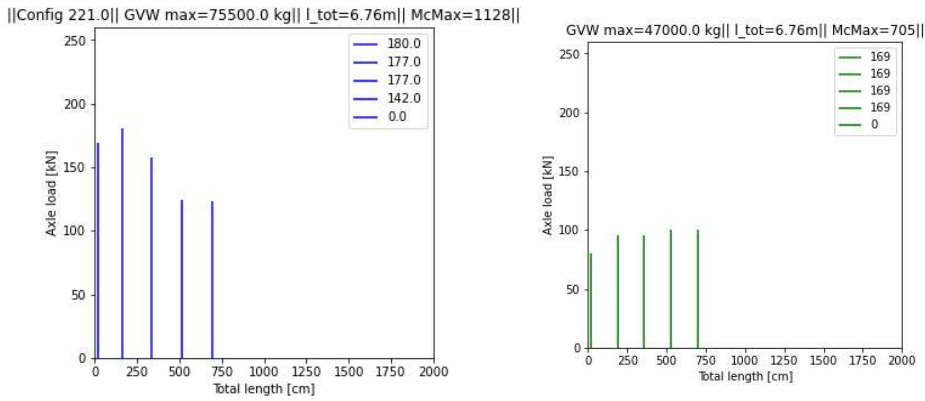


Figure 25 – Comparison between axle loads for tipper trucks in the WIM data (left) and LP data (right).

Based on the interspatial axle distances, this vehicle is most likely a 5-axled tipper truck or a 5-axled mobile crane. Since the WIM data does not clearly distinguish mobile cranes, as shown in appendix B, therefore it is assumed that this is a 5-axled tipper truck.

When compared to tipper trucks in the LP data, shown in the right figure, it becomes clear that the individual axle loads are underestimated, for the latter three axles at least with a factor of 1.5. For the axle spacings it seems that the assumptions made in paragraph 5.1.3 are a pretty good estimate, only the last axle is closely spaced to the fourth than in the assumed load model. In terms of GVW, it is likely that this particular vehicle is overloaded, considering the high individual axle loads for the latter three axles.

However, the figures above consider the highest occurring load effect caused by vehicle class 221. For the same class, now the lowest occurring load effect is shown in the figure below and immediately it shows that the individual axle loads are lower. In fact, the lowest recorded load effect for vehicle class 221 causes very much the same load effect as the assumed load model used in the LP data.

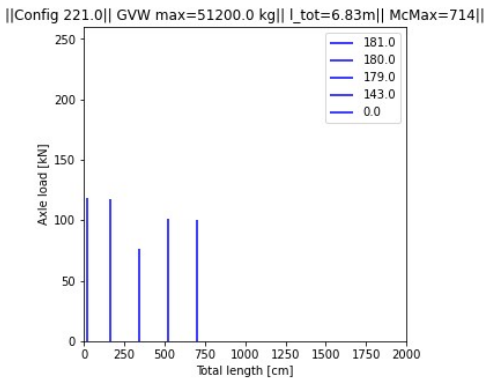


Figure 26 - Configuration of axles for tipper truck from WIM data.

So based on this comparison, it becomes clear that the LP data is a good estimate but needs more tweaking to look more adequately like the WIM data. The calculated load effect caused by a vehicle is constructed by two components: the axle distance and the axle load. Hence, for both components a comparison is made with the WIM data to check whether the initial assumptions were right. This is thoroughly done for each vehicle type in appendix C. At first it must be stated that the considered WIM data registered vehicles with a GVW > 3.5t, where the vehicles in the LP model only have a GVW > 12t. So initially the WIM data will be filtered to fulfil to this condition. Whenever it turns out that either one of the components is wrongly assumed, measures can be taken to correct for this. This can either

be done by changing the initial assumptions and rerunning the process, or by finding a factor to tweak either one, or both, component. The paragraph below shows a summary of the drawn conclusions.

5.3.2 Results of comparison between LP- and WIM data.

In this paragraph all obtained information regarding the comparison between WIM data and LP data will be evaluated and some intermediate conclusions will be drawn. Considering the entire dataset, the adopted strategy to use car license plate registrations as an input to calculate load effects seem to be reasonably close to the WIM data. This is shown in figure 22 and 23. However, these figures show that the WIM data contains outliers, which are not well represented by the LP data. Especially for vehicles with a GVW > 60t. As shown extensively shown in appendix C, this is a combination of overloaded axles and closely spaced axles. Based on this, for both these parameters the following conclusions are drawn.

**Axle loads**

The table below is a summary of the results given in appendix C. The comparison is made from the perspective of the WIM data.

N3-Vehicle class	O4-vehicle class	Conclusion
2-axled semi-trucks	Any semi-trailer	Both axles are overloaded.
3-axled semi-trucks	Any semi-trailer	First two axles are overloaded.
4-axled tipper trucks	None	Primarily underloaded axles.
5-axled tipper trucks	None	Most axles are overloaded in most cases.
4- and 5-axled mobile cranes	3-axled trailer	All axles are overloaded.
2-axled full trucks	None	Both axles are underloaded.
2-axled full trucks	Any non-semi-trailer	First axle is primarily overloaded.
3-axled full trucks	None	First and third axle are overloaded.
3-axled full trucks	Any non-semi-trailer	All axles are primarily overloaded.
4-axled full trucks	None	First axle is overloaded.
4-axled full trucks	Any non-semi-trailer	Most axles are overloaded.
Extreme heavy combination	6+ axles.	No relation is found about axle distances; hence the underestimated load effect must come from overloaded axles.

Table 7 - Summary of conclusions for comparison of axle loads between LP data and WIM data.

As can be seen, for almost all vehicle classes the conclusion can be drawn that at least one axle is overloaded. There where this did not was the case, the majority of the axles were underloaded. This stresses that axle loads are best to be defined as a range of values, rather than a static value, which has been the case so far. Therefore, it is suggested to simulate this under- and overloading behaviour of axles by an axle load factor.



## Interspatial axle distance

N3-Vehicle class	O4-vehicle class	Conclusion
2-axled semi-trucks	Any semi-trailer	Initial assumptions were a good estimate.
3-axled semi-trucks	Any semi-trailer	For 5-axled combinations, distance DT34 will be adjusted.
4-axled tipper trucks	None	Assumed axle configuration made in paragraph 5.1.3 is wrong and needs to be adjusted to approximate a more evenly spaced axle configuration.
5-axled tipper trucks	None	Initial assumptions were a good estimate.
4-axled mobile cranes	3-axled trailer	Distance DT34 is slightly overestimated.
5-axled mobile cranes	3-axled trailer	Distance DT12 is slightly underestimated. Distance DT23 and DT45 are slightly overestimated.
2-axled full trucks	None	Initial assumptions were a good estimate.
2-axled full trucks	Any non-semi-trailer	Distances DT23 and DT34 are unrealistic.
3-axled full trucks	None	Distance DT23 is relatively short and needs to be reviewed.
3-axled full trucks	Any non-semi-trailer	Minimum values for distances DT23, DT34 and DT45 are unrealistically low.
4-axled full trucks	None	Initial assumptions were wrong and need to be adjusted.
4-axled full trucks	Any non-semi-trailer	All values for distances DT45 and DT56 are unrealistic.
Extreme heavy combination	6+ axles.	No comparison was made since no such vehicle type exists in the WIM data. Hence, the conclusion can be drawn that in the WIM data overloaded axles cause maximum load effect. For LP data, the initial assumptions lead to comparable load effects.

Table 8 - Summary of conclusions for comparison of interspatial axle distances between LP data and WIM data.

The table above shows a summary of conclusions obtained for comparing the interspatial axle distances of the LP data with the WIM data. For some vehicle classes, the assumed axle configuration was wrong and will be adjusted accordingly. For other vehicle classes, certain axle distances were either unrealistically low or wrongly assumed. These will be reviewed again and are changed if deemed necessary.

### 5.4 Adjustments of approach

According to the conclusions drawn in the previous paragraph, certain measures have to be taken for both the axle loads and the interspatial axle distances. For both components, certain measures are opted to tweak the LP data to looking more like the WIM data.

#### 5.4.1 Interspatial axle distances

Firstly, the interspatial axle distances that seem to be unrealistic or corrupt will be reviewed upon and are changed accordingly. Furthermore, about 40 entries had unrealistic interspatial distances and are removed from the dataset.



### 5.4.2 Axle load factors $\eta$

Secondly the axle loads will be adjusted. Following from the conclusions in paragraph 5.3, it turned out that for each axle of each vehicle type a range of axle loads exists, rather than the given static value by the RDW. The range of axle loads is likely caused by under- and overloaded vehicles in the WIM data. To simulate this in the LP data, an axle load factor  $\eta_i$  will be introduced. This factor  $\eta_i$  is determined by comparing the axle loads of the vehicle in the LP data with the axle loads of similar vehicles in the WIM data. This process is done for every recorded vehicle and for each  $i$ -th axle of that vehicle. The process below shows how values for  $\eta_i$  are determined.

1. For each recorded vehicle in the LP data, similar vehicle types with the same number of axles are selected in the WIM data.
2. Start with the first axle and calculate all possible values for  $\eta_1$  by  $\frac{AX_{WIM,1}}{AX_{LP,1}}$ . The value of  $AX_{WIM,1}$  follows from the load of the 1<sup>st</sup> axle in the WIM data. The value of  $AX_{LP,1}$  is static and follows directly from the LP data. This results in a range, or distribution, of  $\eta_1$ .
3. Step 2 is repeated for each axle of that vehicle.
4. Finally, for each axle, a range of values for  $\eta_i$  exists. These are stored in separated vectors.
5. When the load effect is calculated, for each axle a random value for  $\eta_i$  is drawn from the separated vectors, as determined in step 4, properly simulating under- and overloaded axles.
6. Steps 2 to 5 are repeated for all recorded vehicles in the LP data.

In the figure below, an example is given for a 2-axled semi-truck carrying a single axle trailer. The red lines indicate the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

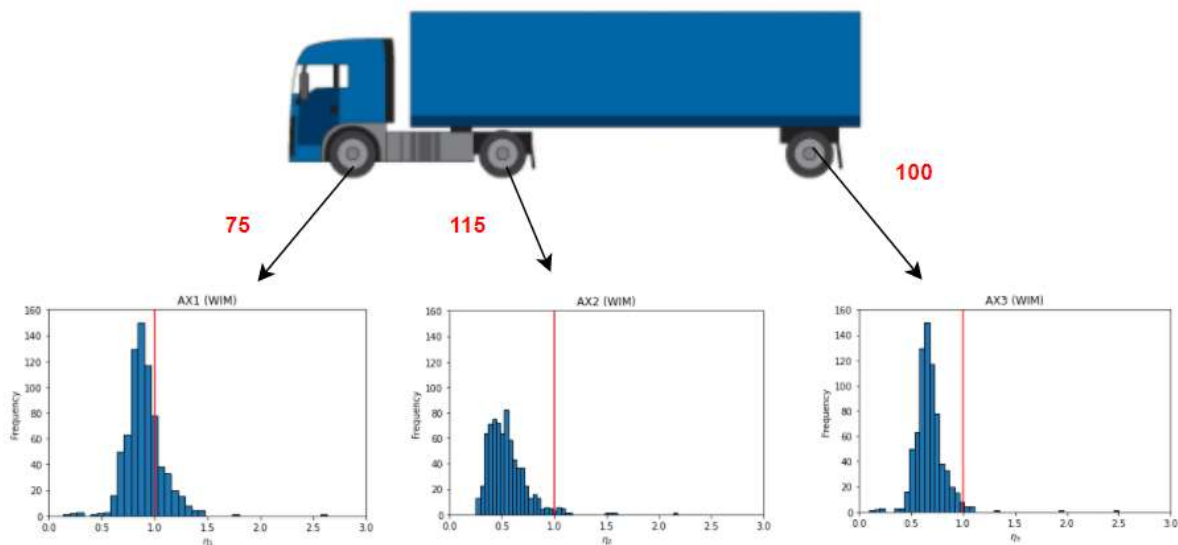


Figure 27 - Distribution of  $\eta_i$  for a 2-axled semi-truck carrying a single axle trailer.

It must be noted that the distribution of  $\eta_i$  for each vehicle and each vehicle type differs, since it is based on the axle loads of the given system. If, for example, the axle load of the first axle was 90 kN instead of 75 kN, the distribution would shift to the left side of the red line. For each vehicle type an example is given in appendix H.

### 5.5 Post-processing

The approach stated in paragraph 5.1.2 is adjusted according to the measures suggested in paragraphs 5.4.1 and 5.4.2. The two figures below show the results for the occurring load effect  $M_{Max}$  based on the initial approach, the adjusted approach, and the WIM data. The top figure compares the initial

model using the initial LP data with the WIM data. The bottom figure compares the modified LP data with the WIM data.

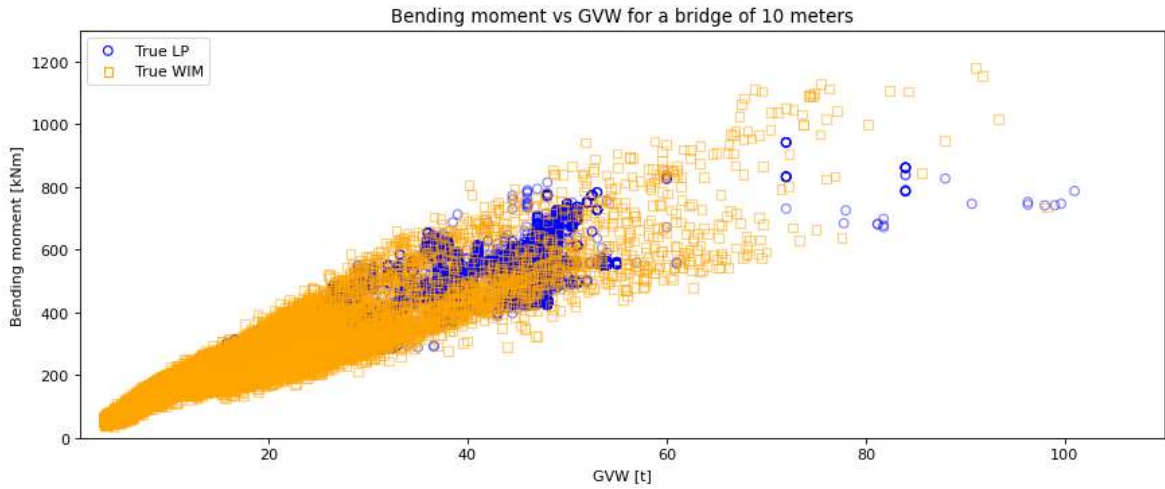


Figure 28 - Bending moments vs GVW for both the LP data and the WIM data.

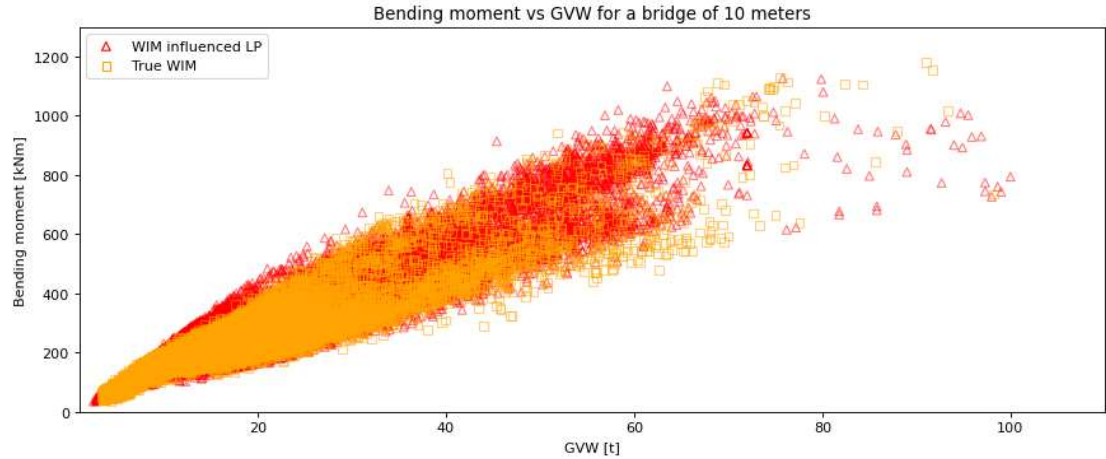


Figure 29 - Bending moments vs GVW for the modified LP data and the WIM data.

The adjusted approach now shows a more spread out behaviour and approximates the load effect caused by the WIM data better. When looked at vehicles with a  $GVW > 70t$  the differences between the two approaches become even more clear. As said before, the two factors for determining load effect are axle loads and interspatial axle distances. Since no excessive outliers are shown in the bottom figure, the conclusion can be drawn that the assumptions made in paragraph 5.1 regarding these distances are valid.

Combined with the axle load factors  $\eta_i$  based on the WIM data, the conclusion can be drawn that the adjusted approach constructs realistic load models. Figure 27 shows the difference between the raw LP data and the modified LP data, properly simulating under- and overloaded vehicles as they are recorded in the WIM data.

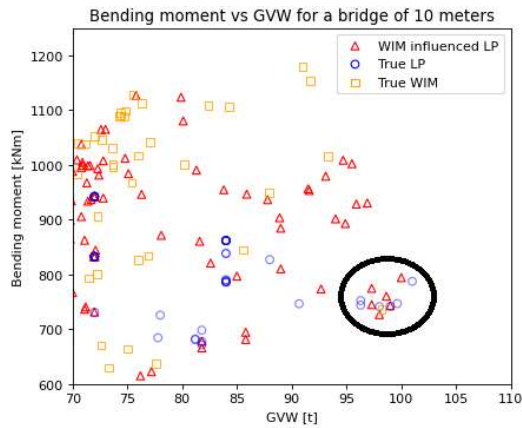


Figure 30 - Comparison between the 3 load models. The black circle indicates vehicles falling in the extreme heavy combinations (EHC).

Since the WIM data is measured in the city of Rotterdam [12], it is likely that these vehicles are driving throughout the main roads of the city of Rotterdam and thus also crossing the bridge reviewed in this thesis.

Note that for the category extreme heavy vehicles no adaptations were made since no comparable vehicles in the WIM data were found, and that they remain relatively the same for LP data and WIM-influenced LP data. This is indicated with the black circle. The differences that are noticeable however, are due to the assumption of the interspatial axle distances as stated in paragraph 5.1

## 5.6 Influence of span length

One of the motivations for this study to be done, was to investigate whether the tandem model from LM1 from the Eurocode was an overconservative load model for relatively short span bridges. A span length of  $1 \text{ m} < l_{\text{span}} < 20 \text{ m}$  can be categorized as a 'short span bridge'. Figure 18 and 86 already show that solely the tandem load from LM1 is rather conservative. However, this can only be said for  $l_{\text{span}} = 10 \text{ m}$ . For other spans, this statement cannot be made, yet. To investigate this, the same procedure for constructing load models is used as before, but now the span length varies from 1 m to 20 m. In the figures below, the results are plotted for span lengths of 5,10,15 and 20 m respectively.

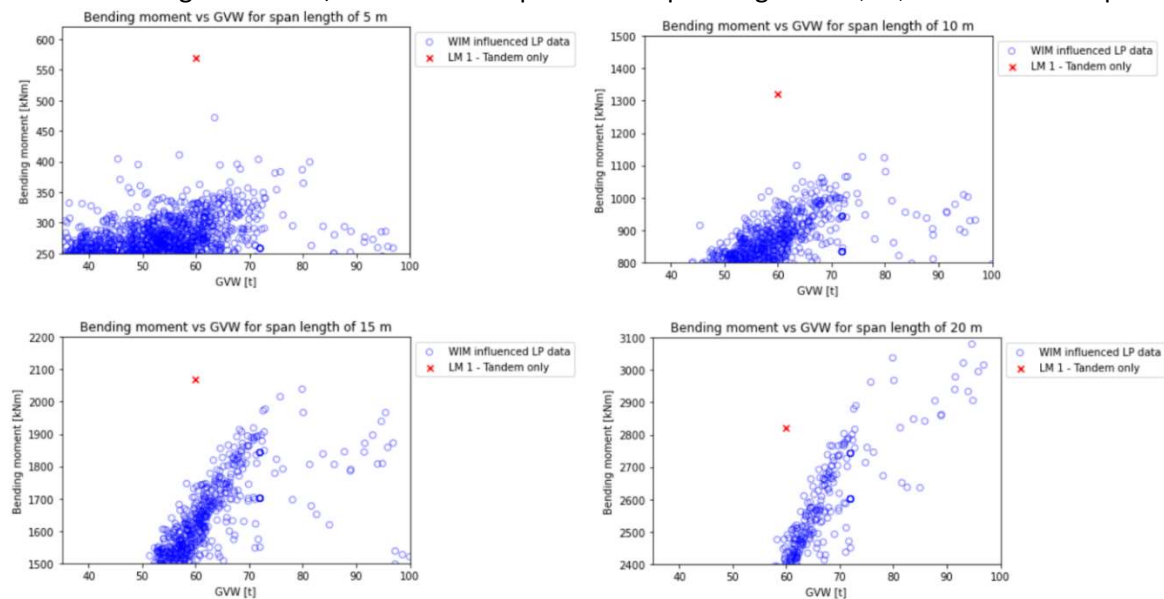


Figure 31 - Figures for bending moment vs GVW for a span length of 10 m for the modified LP data, containing axle load factor  $\eta$ . Results are compared to the tandem load from LM1.

Note that for a span length of 15 m the results of applying the adopted approach almost exceed the load effect caused by the tandem load of LM1. For span lengths of 20 m however, 10 % of the highest load effects exceed the threshold value of LM1. The calculated bending moment by LM1 is only due to the tandem load. Now the question rises what the influence of the uniformly distributed load,  $q_{UDL}$ , is on the bending moment due to traffic loading. Hence, the same plots are made as in figure 31, but the uniformly distributed load is added to the value corresponding to LM1. The value of the uniformly distributed load is  $q_{UDL} = 9 \text{ kN/m}^2$ , according to NEN-EN 1991-2. Only strips with a width of 1 m are considered. The results are shown in the figure below.

The results clearly show that the bending moment caused by LM1 is never exceeded for all span lengths. For span lengths of 5 m and 10 m the highest exerted bending moment does not come close to the value of LM1. For these span lengths, most vehicles with a GVW ranging from 35t to 100t exert the same amount of load effect on the span. Since the span is rather short, this is caused by overloaded axles rather than overloaded vehicles.

For span lengths of 15 m and 20 m a similar pattern is shown. Namely that the bending moment according to LM1 is never exceeded by the simulated vehicles. However, the highest bending moments are caused by the heaviest vehicles as shown in the figure. For longer span lengths, the load effect is caused by overloaded vehicles rather than by overloaded axles.

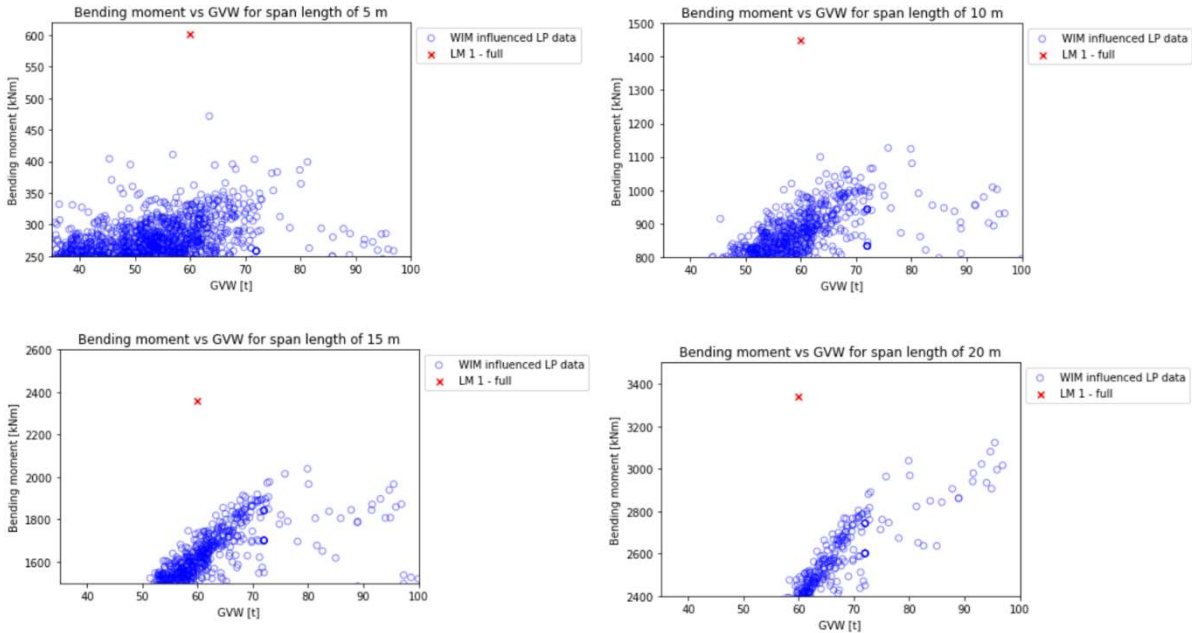


Figure 32 - Figures for bending moment vs GVW for a span length of 10 m for the modified LP data, containing axle load factor  $\eta$ . Results are compared to the full load model LM1.

Based on the obtained results, the conclusion can be drawn that for short spans, i.e. 5 m and 10 m, overloaded axles are causing the highest load effects rather than overloaded vehicles. For longer spans, i.e. 15 m and 20 m, the highest load effects are caused by the heaviest vehicles rather than overloaded axles.

## 6 Monte Carlo Simulation

### 6.1 Introduction

In this chapter, the approach to apply a Monte Carlo simulation to the limit state function is given. Firstly, the definition of the limit state function, in combination with a Monte Carlo simulation is given. Secondly, the necessary components for the limit state function, load distribution  $S(x)$  and resistance distribution  $R(x)$  are explained. The results from the previous chapter can be used as an input for  $S(x)$ . The input for  $R(x)$  has yet to be determined. Finally, a piece of code to run a Monte Carlo simulation is given to simulate the return period of 1000 years and to determine the probability of failure.

#### 6.1.1 Basic principle

The concept of a Monte Carlo simulation is to use a probabilistic approach to solve problems that might be deterministic in nature. Monte Carlo simulations are specifically used when the initial conditions, or the starting values of certain parameters, are not deterministic. Thus, the parameters can have a range of values. So basically, a Monte Carlo is a deterministic process, done an  $N$  amount of times. In this study, the hypothesis is whether the structure fails due to the occurring load or not. These results can be described by making use of the limit state function, shown below.

$$Z(x) = R(x) - S(x)$$

Since an  $N$  amount of simulations is done, also an  $N$  amount of results is obtained. In this case, the parameters resistance  $R$  and the imposed load  $S$  have their own distribution functions. During a Monte Carlo simulation, a random value is generated from these distribution functions, and the limit state  $Z(x)$  is then calculated for that specific set of parameters. These random values are generated from a cumulative distribution function. This distribution function can have any shape. In the picture below, a non-linear distribution function is given.

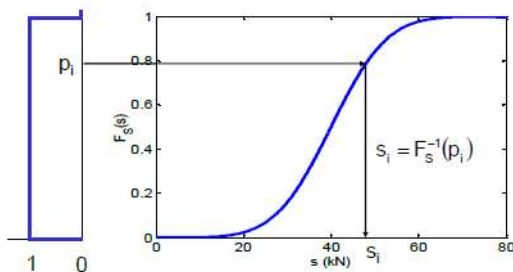


Figure 33 - Example of a non-linear cumulative distribution function<sup>13</sup>.

This process is repeated until all combinations of parameters have been considered. So, the outcome of a Monte Carlo simulation are values for the limit state  $Z(x)$  with a statistical distribution function. To illustrate what that looks like, in the picture below the results of a Monte Carlo simulation is shown with 200 samples, i.e.  $N = 200$ .

<sup>13</sup> Source: Probabilistic Design: Risk and Reliability Analysis in Civil Engineering. Lecture notes CIE4130.

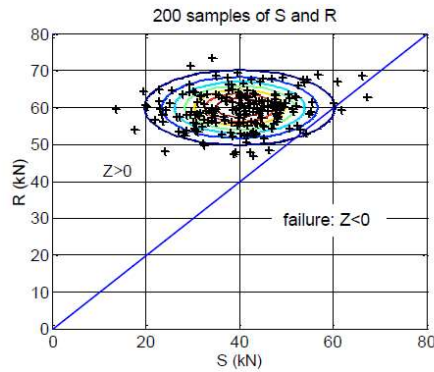


Figure 34 - Example of the limit state function using a Monte Carlo simulation.<sup>14</sup>

### 6.1.2 Reliability index

The main goal of probabilistic design is to design and maintain structures or structural elements with an acceptable risk level in an optimal way. One way to express this risk level is to derive the level of failure, or the relative failure probability  $P_f$ . For  $N \rightarrow \infty$  simulations,  $P_f$  can be described as:

$$P_f = \frac{N_f}{N}$$

Where  $N$  is the number of simulations and  $N_f$  is the number of simulations where  $Z(x) < 0$ .

At first, one must determine what an acceptable level of risk is, often called the target value for reliability. The target value for reliability is often expressed by means of a reliability index  $\beta$ . The reliability index is directly related to the probability of failure, as can be seen from the following equation.

$$\beta = -\Phi^{-1}(P_f)$$

The reliability index also depends on the reference period to which reference period they are applied. In case the yearly maxima are independent, the following relationship can be used to convert  $\beta$  values in relation to different reference periods.

$$\Phi(\beta_n) = [\Phi(\beta_1)]^n$$

## 6.2 Approach

### 6.2.1 Determining load distribution $S(x)$

To determine the load distribution  $S(x)$ , the calculated load effects from the previous chapters are used as a base input for 1 year of load effects. Then using extreme value analysis, several maxima distributions are fitted, resulting in a load distribution  $S(x)$ . The steps are given below:

1. Calculate load effects for 1 year of daily data and store daily maxima
2. Fit a distribution to the daily maxima, called  $f_{day}$
3. Use distribution  $f_{day}$  to simulate a return period of  $t$  years and store the monthly maxima
4. Fit a distribution to the monthly maxima, called  $f_{mon}$
5. Use distribution  $f_{month}$  to simulate the return period 1000 times and store the yearly maxima
6. Finally, fit a distribution called  $f_{yearly}$  to the yearly maxima. This distribution function is the fitted load distribution  $S(x)$ .

<sup>14</sup> Source: Probabilistic Design: Risk and Reliability Analysis in Civil Engineering. Lecture notes CIE4130.



To illustrate the steps stated above, figure 34 shows 3 various load distributions  $S(x)$  for the unmodified LP data, the WIM-influenced LP data and the unmodified WIM data. These 3 models are fitted to a Gumbel-distribution, with parameters accordingly. Chapter 7 will describe this process more in detail.

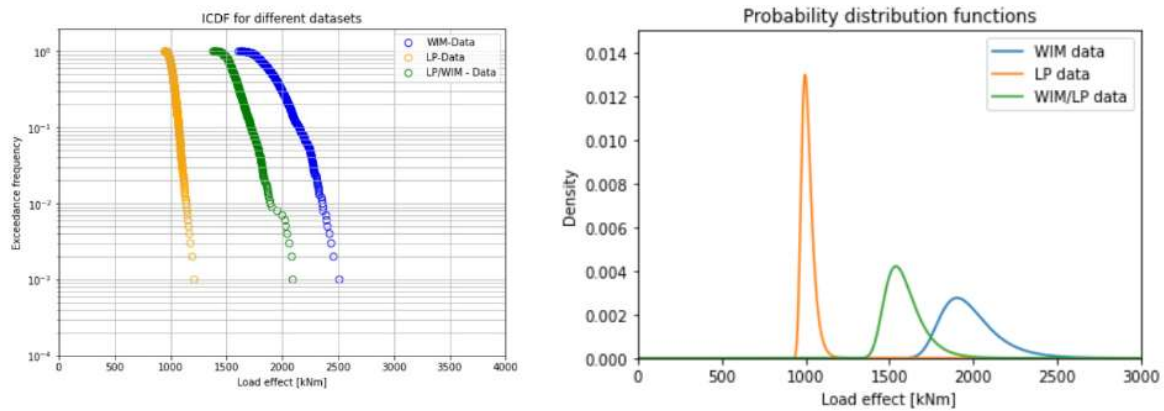


Figure 35 - Example of results obtained by following suggested approach. Left: icdf's for different load models. Right: pfd's for different load models.

#### 6.2.1.1 Curve fitting

Any probability function can be fitted to every dataset. However, how well the fitting is done is highly important. For similar data as used in this thesis, TNO [12, p23] showed that a multimodal distribution was to be preferred over a single Weibull distribution<sup>15</sup>. Hellebrandt [1, p95] showed that for load effect distributions multiple distributions were approximating the distribution, but that a single Gumbel distribution fitted the data best. Figure 20 already shows that the distribution for the load effect is not described by a single distribution, but rather a mixture of multiple distributions, possibly a mixture Gaussian distribution. Hellebrandt [1] and TNO [12] showed that properly fitting distribution to the data is of most importance and thus these steps must be taken care of accordingly.

#### 6.2.2 Determining resistance distribution $R(x)$

A similar approach as used by Hellebrandt [1] will be applied in this thesis, where the JCSS PMC functions as a guideline to determine the resistance distribution  $R(x)$ . A similar procedure was used in [13] and will act like a guideline. This code prescribes distributions for concrete material properties. With these material properties the flexural bending resistance of a concrete cross-section can be determined following these steps:

1. Determine distributions for concrete parameters
2. Determine distributions for reinforcement steel parameters
3. Determine geometrical parameters
4. Calculate resistance  $M_{Rd}$  using concrete mechanics

The following paragraphs describe how to determine the distributions for step 1 and 2. Steps 3 and 4 however are highly depending on parameter input and are not described in further detail. In chapter 7 the necessary parameters are given to complete steps 3 and 4.

<sup>15</sup> This section was about fitting a curve to individual axle loads.

### 6.2.2.1 Concrete properties

#### 6.2.2.1.1 Compressive strength

According to JCSS PMC Part 3, all basic concrete parameters are based on the compressive strength, modified with (distribution) parameters. The general compressive strength of a concrete element can be expressed as:

$$f_{co,ij} = \exp \left( m'' + t_{v''} \cdot s'' \cdot \sqrt{\left(1 + \frac{1}{n''}\right)} \right)$$

Where  $m''$ ,  $s''$  and  $n''$  are all parameters with respect to the initial concrete grade. The initial concrete grade is manually chosen at first. And the parameter  $t_{v''}$  is the Student-distribution for  $v''$  degrees of freedom. The values for these parameters are given in a table, shown below. With these parameters a distribution for  $f_{co,ij}$  can be determined.

Table 3.1.2: Prior parameters for concrete strength distribution ( $f_{co}$  in MPa) [1, 2]

Concrete type	Concrete grade	Parameters			
		$m'$	$n'$	$s'$	$v'$
Ready mixed	C15	3.40	3.0	0.14	10
	C25	3.65	3.0	0.12	10
	C35	3.85	3.0	0.09	10
	C45	3.98	3.0	0.07	10
	C55	-	-	-	-
Pre-cast elements	C15	-	-	-	-
	C25	3.80	3.0	0.09	10
	C35	3.95	3.0	0.08	10
	C45	4.08	4.0	0.07	10
	C55	4.15	4.0	0.05	10

Figure 36 - Figure of different parameters for different concrete grades.<sup>16</sup>

#### 6.2.2.1.2 Tensile strength, modulus of elasticity and ultimate strain

As said before, the other concrete parameters are directly derived from the compressive strength. The tensile strength, modulus of elasticity and the ultimate strain are determined as follows respectively:

$$f_{ct} = 0.3 \cdot f_{c,ij} \left(\frac{2}{3}\right) \cdot Y_{2,j}$$

$$E_{c,ij} = 10.5 \cdot f_{c,ij} \cdot Y_{3,j} \cdot (1 + \beta_d \cdot \varphi(t, \tau))^{-1}$$

$$\varepsilon_{u,ij} = (6 \cdot 10^{-3}) \cdot f_{c,ij}^{-\frac{1}{6}} \cdot Y_{4,j} \cdot (1 + \beta_d \cdot \varphi(t, \tau))$$

The variables  $Y_{i,j}$  contain parameters according to the log-normal distribution with predetermined mean and covariance parameters as shown in the table below.

<sup>16</sup> Source: JCSS Probabilistic Model Code Part 3: Table 3.1.2.



Variable	Distribution type	Mean	Coefficient of variation	Related to
$Y_{1,j}$	LN	1.0	0.06	compression
$Y_{2,j}$	LN	1.0	0.30	tension
$Y_{3,j}$	LN	1.0	0.15	E-modulus
$Y_{4,j}$	LN	1.0	0.15	ultimate strain

Figure 37 - Distribution types and parameters for different concrete parameters.<sup>17</sup>

Parameter  $\beta_d$  is the ratio of the permanent load to the total load and depends on the type of the structure is between  $0.6 \leq \beta_d \leq 1.0$ . [JCSS PMC3 Part 3, p2]. Parameter  $\varphi(t, \tau)$  is the creep coefficient according to the Eurocode and is deterministic [JCSS PMC3 Part 3, p2].

### 6.2.2.2 Reinforcement steel properties

#### 6.2.2.2.1 Steel yield strength

JCSS PMC3 Part 3 also prescribes properties for the reinforcement steel since these properties become probabilistic as well. Paragraph 3.2 prescribes how these properties are determined. Firstly, the yield stress  $X_1$  can be taken as the sum of three independent Gaussian variables [JCSS PMC part 3, p26].

$$X_1(d) = X_{11} + X_{12} + X_{13}$$

Where  $X_{11}$  represent variations in the global mean of different mills,  $X_{12}$  represents the variations in a mill from batch(melt) to bath and  $X_{13}$  represents the variations within a melt. All  $X_{1i}$  variables have a normal distribution with the following parameters:

Property	Distribution
$X_{11}$	$\mathcal{N}(\mu_{11}(d), 19)$
$X_{12}$	$\mathcal{N}(0, 22)$
$X_{13}$	$\mathcal{N}(0, 8)$

Table 9 - Distribution parameters for different steel parameters.

Variable  $X_{11}$  is influenced by the bar diameter  $d$  and initial yield stress  $f_y$  and thus becomes parametric, since parameter  $\mu_{11}(d)$  can be expressed as:

$$\mu_{11} = \mu_1(d) = (f_y + 2 \cdot 30) \cdot (0.87 + 0.13 \cdot e^{(-0.08d)})^{-1}$$

#### 6.2.2.2.2 Bar area

JCSS PMC Part 3 also prescribes the statistical parameters for the total bar area of a cross section, given in table [JCSS PMC Part 3, table 3.1.5] with the following distribution properties:

Property	Distribution
$A_s$	$\mathcal{N}(A_s, 0.06 \cdot A_s)$

Table 10 - Distribution parameters for steel area.

### 6.2.2.3 Concrete and steel force

#### 6.2.2.3.1 Steel force

The force in the steel is expressed as:

$$F_{steel} = A_s \cdot f_{yd}$$

<sup>17</sup> Source: JCSS Probabilistic Model Code Part 3: Table 3.1.1.

Where both  $A_s$  and  $f_{yd}$  are described by their distributions according to the previous paragraphs. The picture below shows how the force in the steel is calculated in the script. The parameters  $S_{xx}$ ,  $n$  and  $d$  are all input based.  $S_{xx}$  is the initial steel yield stress [N/mm<sup>2</sup>],  $n$  is the number of rebars and  $d$  is their respective bar diameter [mm]. The script returns a singular value, expressed in [kN].

```
def F_steel(Sxx,n,d): #F_steel in kN
    X12 = normal_sample(0,22)
    X13 = normal_sample(0,8)
    mu1 = Sxx + 2*30
    mu11 = mu1 * ((0.87 + 0.13*np.exp(-0.08*d))**-1)
    X11 = normal_sample(mu11,19)
    fyd = X11 + X12 + X13

    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)

    return fyd * As_sample / 1e3 , fyd
```

Figure 38 - Excerpt of python code to determine force in steel  $F_{steel}$ .

### 6.2.2.3.2 Concrete force

If all concrete parameters are known, the force in the concrete  $F_{conc}$  can be calculated to the figure below accordingly. The figure below shows the acting forces in a concrete cross-section. The force in the concrete, i.e.  $F_{conc}$ , is calculated as  $F_{conc} = 0.85 \cdot f_{ck} \cdot x_c$ .

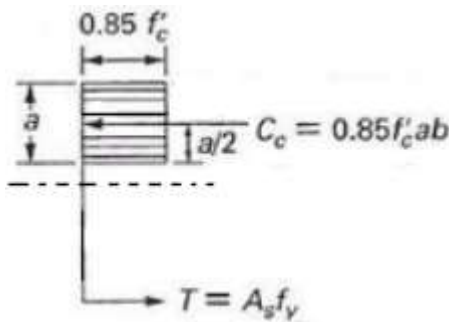


Figure 39 - Schematically representation of acting forces in a concrete cross-section.<sup>18</sup>

However, the compressive height  $x_c$  must be determined first, which can be through:

$$x_u = \frac{A_s \cdot f_y}{0.85 \cdot f_{ck}}$$

Obviously  $F_{conc}$  depends on the steel parameters, and thus also on  $F_{steel}$  which is described in the previous paragraph. The figure below shows the script calculating  $F_{conc}$ . The input parameters are the same and should read the same as for  $F_{steel}$ . The script also returns a single value, expressed in [kN].

```
def F_conc(Sxx,n,d): #F_conc in kN
    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)
    xc = (As_sample)*F_steel(Sxx,n,d)[1] / (0.85*lognorm_sample(np.log(fcp[2]),(fcp[0])))
    Fcd = 0.85*exc * lognorm_sample(np.log(fcp[2]),fcp[0])
    return Fcd/1e3
```

Figure 40 - Excerpt of python code to determine force in the concrete  $F_{conc}$ .

<sup>18</sup> Source: <https://civilengineer.webinfolist.com/design/rcsingle.png>

#### 6.2.2.4 Flexural bending resistance $M_{Rd}$

Now all parameters for both the concrete and the reinforcing steel are known, and thus the resistance can be calculated. To do this, simple concrete mechanics are applied, see the figure below.

Since the  $M_{Rd}$  is highly dependent on steel yield stress, internal lever arm, bar diameter and bar area, concrete grade and much more, a general method is opted where these parameters can be used as an input to determine  $M_{Rd}$ . Using concrete mechanics, the bending resistance can be calculated as  $M_{Rd} = F \cdot z$ , where  $z$  is the internal lever arm and  $F$  is the lowest value of  $F_{steel}$  and  $F_{conc}$ , which are both described in the previous paragraph. The figure of the script below shows how  $M_{Rd}$  is calculated.

```
def M_res(Sxx,n,d,z): #M_res in kNm
    fs = F_steel(Sxx,n,d)[0]
    fc = F_conc(Sxx,n,d)
    focc = np.minimum(fs,fc)
    mres = focc*z
    return mres/1e3
```

Figure 41 - Excerpt of python code to calculate bending moment resistance  $M_{Rd}$ .

The script uses as the same input as the scripts for calculating  $F_{conc}$  and  $F_{steel}$  and now also requires an additional parameter  $z$ . Twice, the forces  $F_{steel}$  and  $F_{conc}$  are calculated and the minimum occurring force is then selected. Ultimately the script returns a bending moment in [kNm].

#### 6.2.2.5 Model uncertainties

According to JCSS PMC part 3 chapter 3.9, two final variables  $\theta_R$  and  $\theta_S$  for resistance and load respectively will be introduced. These variables account for random effects that are neglected in the models and simplifications in mathematical relations<sup>19</sup>. These variables have their recommended probabilistic models which are described in table 3.9.1 in JCSS PMC Part 3 chapter 3.9. Since this thesis focusses on bending moments in plates, modelled as frames, the following distributions are applied.

Property	Distribution
$\theta_S$	LN(1,0.1)
$\theta_R$	LN(1.2,0.15)

Table 11 - Distribution parameters for model uncertainties  $\theta_S$  and  $\theta_R$ .

#### 6.2.3 Method of simulating

The previous paragraphs described all the required parameter inputs needed to be able to determine forces and resistances. The part that is left is how the Monte-Carlo simulation will be performed. Such a simulation however is basically the same calculation done  $N$  times. The figure below shows an excerpt of the script of the simulation. The function  $MCsim$  requires the same input parameters as the function  $M_{res}$ , but also an additional parameter  $e$ . This parameter is simply the number of times the simulation must be done, i.e.  $N$  as used earlier. The script then plots a limit state function like figure 33 and calculates the probability of failure  $P_f$ . This is shown in the figure below.

<sup>19</sup> JCSS Probabilistic Model Code Part 3: paragraph 3.9.

```

1 def MCsim(Sxx,n,d,z,e):
2     print('Running a Monte-Carlo simulation for',e/1e6,'million times.')
3     loadarr = np.zeros(e)
4     resarr = np.zeros(e)
5     zarr = np.zeros(e)
6     zarr2 = np.zeros(e)
7
8     with tqdm(total=e, file=sys.stdout) as pbar:
9         for r in range(e):
10            pbar.set_description('MC-simulation')
11            pbar.update(1)
12            loadarr[r] = (gumbel_sample(mu_yr,beta_yr)+M_G_ED)*lognorm_sample(np.log(1),0.1)
13            resarr[r] = M_res(Sxx,n,d,z)*lognorm_sample(np.log(1.2),0.15)
14            zarr[r] = loadarr[r]/resarr[r]
15            zarr2[r] = resarr[r] - loadarr[r]
16

```

Figure 42 - Excerpt of python code of running a Monte Carlo simulation.

### 6.3 Simulation

The previous paragraphs described all required parameter input and described the general approach needed to simulate traffic. The number of times a simulation has to be done,  $N$  or  $e$  in the code, has still to be determined, however. A study done by Lerche [13] shows that  $N$  is relevant when computation time is governing. Hellebrandt [1, p133] showed that the number of simulations must be within the range of  $10^5 - 10^7$ . For Python, the number of entries a vector can have depends on hardware and not on software. Thus, in theory infinite entries can be added to a vector. This is an advantage of Python over Microsoft Excel, where a worksheet is limited by 1,048,576 rows and 16,384 columns. For a mid-range laptop with a i5-4310 2.00 GHz quadcore CPU and 8,00 GB RAM the calculating time for  $10^5$  simulations is currently averaged at 5.5 minutes at 300 iterations per second.

### 6.4 Post processing

To validate whether the obtained results are reasonable, all intermediate steps such as constructing distributions  $S(x)$  and  $R(x)$  should be researched a posteriori. For distributions  $S(x)$  and  $R(x)$  this mainly done by comparing the results with other results in literature.

## 7 Case study

### 7.1 Introduction

The previous chapters have shown that the LP data, although modified, accurately describes the WIM data. Since LM1 from NEN-EN 1991-2 is based on WIM data [15] and the constructed load model in this thesis is a combination of theoretical axle loads and WIM data, it is of interest to compare both load models and explain any differences. To compare both load models, a theoretical case study will be done. For simplicity reasons, this will be a simply supported RC bridge with a span length of 10 m in an urban area. The bridge will consist of multiple girders, but only one girder will be reviewed and will be modelled as a 1D line girder. This is done because the main reason of this study is to compare the load effect caused by different load models on bridges (girders).

### 7.2 Case

#### 7.2.1 Approach

The bridge girder will initially be designed according to LM1 from NEN-EN 1991-2. Using LM1, the reinforcement for ULS will be determined. This initial design, i.e. concrete – and steel parameters, will act as a starting point for the resistance distribution  $R(x)$ , explained in paragraph 6.2.2. Then in paragraph 7.3 the load distribution  $S(x)$  will be determined and finally a Monte-Carlo simulation will be used to probabilistically verify the initial structure.

#### 7.2.2 Initial design

##### 7.2.2.1 Starting parameters

To properly determine load effects caused by LM1, the following starting parameters are assumed:

- Only one span of 10 m is considered.
- Only one theoretical lane of 3.0 m is considered.
- The number of heavy trucks is assumed to be  $N_{\text{obs}} \geq 2 \cdot 10^6$
- The geometry of the girder is assumed to be considered as a plate, i.e. lane widths of 1.0 m are considered.
- The height of the girder is initially set to be 500 mm and may change during calculating.
- Concrete grade C35/45 is used
- Steel grade FeB500 is used

##### 7.2.2.2 Load effect caused by LM1

This hypothetical bridge girder is simply supported and spans 10 m. According to NEN-EN 1991-2 paragraph 4.3, the structure is loaded by:

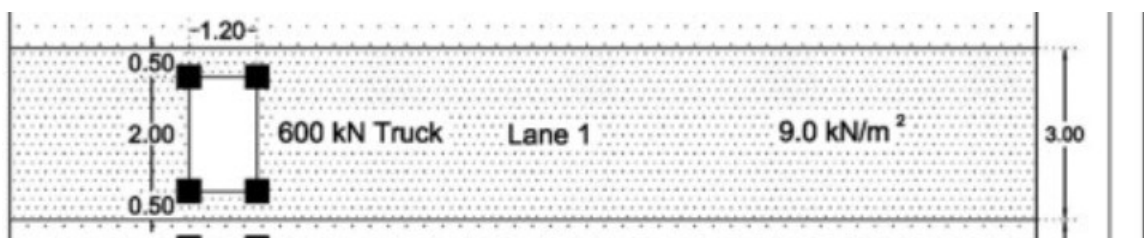


Figure 43 - Load scheme of LM1 from NEN-EN 1991-2<sup>20</sup>.

<sup>20</sup> Source: <https://www.sciencedirect.com/science/article/pii/S1687404814000194>

Where the total width of the lane is 1.0 m and the assumption is made that the entire vehicle is present on a single lane.

$$q_{UDL} = \alpha_{qi} \cdot q_{ik} = 9 \cdot 1,0 = 9 \text{ kN/m}^2$$

$$Q_{1k} = \alpha_{Qi} \cdot Q_{ik} = 1 \cdot 300 = 300 \text{ kN}$$

The design bending moment caused by these two loads is then:

$$M_{udl} = \frac{1}{8} \cdot q_{udl} \cdot l^2 = \frac{1}{8} \cdot 9 \cdot 10^2 = 112.5 \text{ kNm/m}$$

The value for  $M_Q$  follows from the approach to calculate bending moments as stated in paragraph 5.2.

$$M_{Q,Ed} = 1.5 \cdot (M_{udl} + M_Q) = 1.5 \cdot (112.5 + 1320) = 2148.75 \text{ kNm/m}$$

### 7.2.2.3 Load effect caused by own weight

According to NEN-EN 1990 table A2.4, the occurring load effect caused by own weight of the structure is determined by:

$$M_{G,Ed} = 1.35 \cdot M_{G,k}$$

Using the geometry stated in paragraph 7.2.2.1 and a concrete density of  $25 \text{ kN/m}^3$  the following design bending moment occurs:

$$M_{G,Ed} = 1.35 \cdot 156.25 = 210.94 \text{ kNm/m}$$

### 7.2.2.4 Resistance

In the previous paragraphs both the load effect due to own weight and LM1 are calculated. When combined, these result in a total design bending moment of  $M_{Ed} = 2360 \text{ kNm/m}$  in ULS for which the girder will be designed.

#### 7.2.2.4.1 Concrete

Assuming concrete grade C35/45 and according to NEN-EN 1992-1-1 paragraph 3.1.6 the design compressive stress  $f_{cd}$  is set to be:

$$f_{cd} = \alpha_{cc} \cdot \frac{f_{ck}}{\gamma_c} = 19.83 \text{ N/mm}^2$$

#### 7.2.2.4.2 Reinforcement steel

Assuming steel grade FeB500, the yield stress  $f_{yd}$  according to NEN-EN 1992-1-1 paragraph 3.2 the design yield stress  $f_{yd}$  is set to be:

$$f_{yd} = \frac{f_{yk}}{\gamma_s} = 435 \text{ N/mm}^2$$

#### 7.2.2.4.3 Resistance

Assuming the same concrete mechanics shown in paragraph 6.2.2.3.2, the steel in the force and concrete are determined as follows:

$$F_{S,d} = n \cdot A_s \cdot f_{yd}$$

$$F_{C,d} = x_u \cdot 0.85 \cdot f_{cd}$$

Where  $x_u$  can be described as:

$$x_u = \frac{F_{c,d}}{0.85 \cdot f_{cd}} = \frac{n \cdot A_s \cdot f_{yd}}{0.85 \cdot f_{cd}}$$

And the bending moment resistance  $M_{Rd}$  as:

$$M_{Rd} = F_{S,d} \cdot z$$

And the internal lever arm  $z$  as:

$$z = h - c - 0.5 \cdot \emptyset - 0.5 \cdot x_u$$

Where  $z$  is the internal lever arm. As can be seen, the flexural bending moment resistance depends on the internal lever arm. Since this expression depends on multiple variables, the following assumption is made  $z = 0.81 \cdot h$ , such that  $z = 405$  mm. Accordingly, the minimum value for  $F_{S,d,min} = 5826$  kN which is obtained by  $A_{s,min} = 13.4 \cdot 10^2$  mm<sup>2</sup> which corresponds to a tightly spaced reinforcement of rebars 17 –  $\emptyset 32$ , resulting in  $A_{s,tot} = 13.67 \cdot 10^2$  mm<sup>2</sup> and a final resistance of  $M_{Rd} = 2409$  kNm.

This results in a final unity check of  $uc = \frac{2360}{2409} = 0.98$ .

### 7.3 Probabilistic verification modified LP model

The results obtained in the previous paragraph will act as a starting point for the probabilistic verification of this hypothetical girder bridge. For the resistance distribution  $R(x)$ , the proposed reinforcement of 17 $\emptyset 32$  will act as input.

#### 7.3.1 Load distribution $S(x)$

To determine load distribution  $S(x)$  the approach as stated in paragraph 6.2 will be used. The initial source of data, needed for step 1, will be the LP data with the adjustments as described in paragraph 5.4, referred to as the *modified LP model*. This input is given in appendix E. Obviously, the curve fitting done in step 1 is most crucial for other steps. The figure below shows the daily maxima obtained for 1 year of data, i.e. step 1.

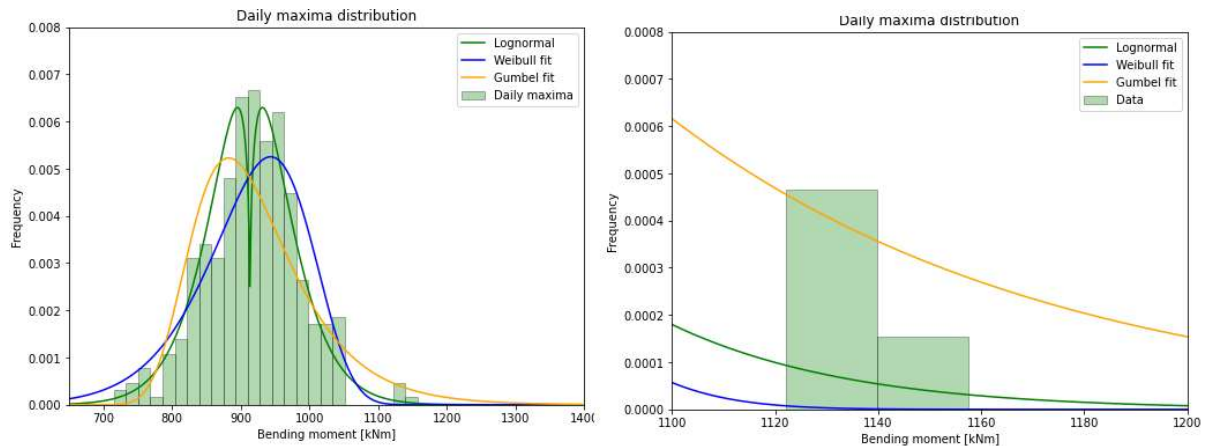


Figure 44 - Relative frequency histograms and fitted distributions for daily maxima for the modified LP load model.

The data does not show clear resemblance of a single GEV-type distribution, nor a unimodal normal distribution. The curve fit shown in figure 43 is done through a Python package called *distfit*. The background to this package, along with an extensive study to curve fitting is shown in Appendix E. For extreme value analysis, the tail of the distribution carries the weight to determine which distribution fits the data properly. Visually, the fitted Gumbel distribution is overestimating the tail and the Weibull



distribution is underestimating the tail. The fitted lognormal distribution using the *distfit* package has a better way of describing the curve. This is strengthened by the figure below.

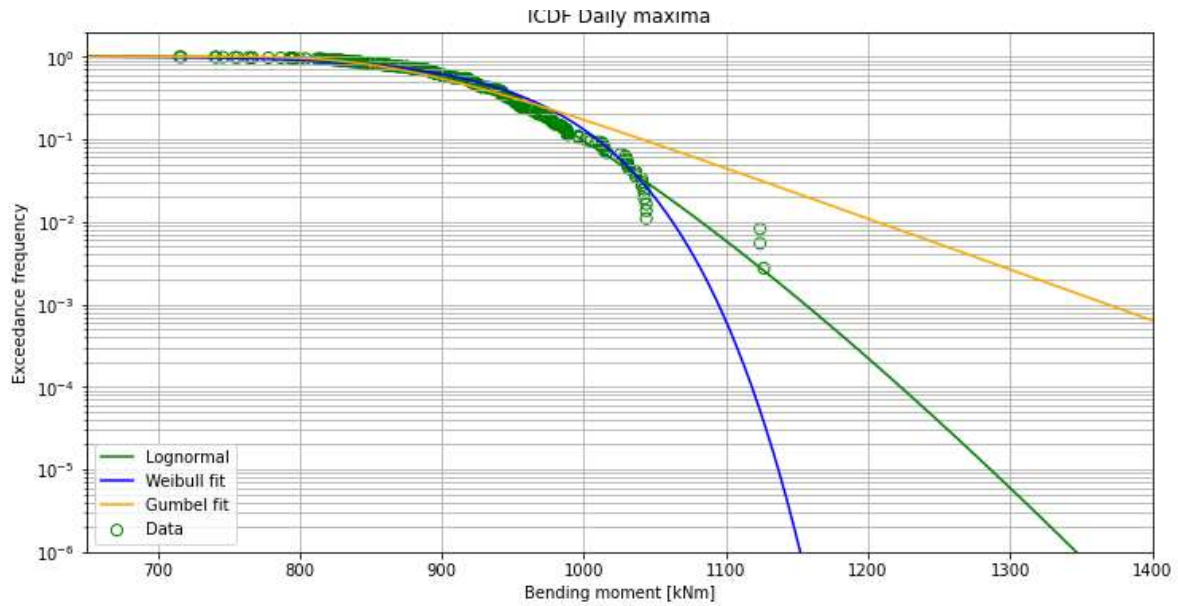


Figure 45 - Fitted inverted cumulative distribution functions for daily maxima for the modified LP load model.

The fitted Gumbel distribution clearly shows an overestimation of the tail of the distribution as opposed to the lognormal distribution. The ‘single’ Weibull distribution shows an underestimation for the tail of the data, especially the 3 values exceeding 1100 kNm. Hence, the fitted lognormal distribution is used as the distribution for the daily maxima. The parameters are shown in Appendix E.

For the monthly maxima, 25 years are simulated by means of drawing samples from the daily maxima distribution. Similarly, as for the daily maxima, several curve fittings have been done. The results are shown below.

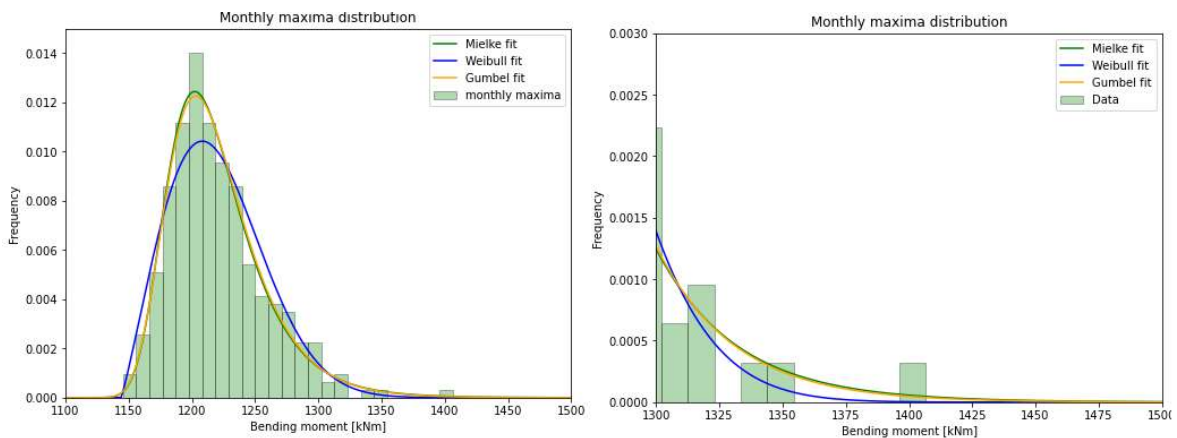


Figure 46 - Relative frequency histograms and fitted distributions for monthly maxima for the modified LP load model.

Visually, the fitted Weibull distribution does not properly fit the data. The *distfit* package determined that a Mielke distribution fitted the data best<sup>21</sup>. However, when inspected, it turned out that the Mielke- and the Gumbel distribution were almost identical. The Mielke distribution however overestimates the tail of the distribution. The Gumbel distribution only slightly overestimates the tail

<sup>21</sup> Extensively described in appendix E.



of the distribution, as shown in figure 45. Hence a Gumbel distribution is used to the fit the data to the monthly maxima. This is further strengthened by the figure below.

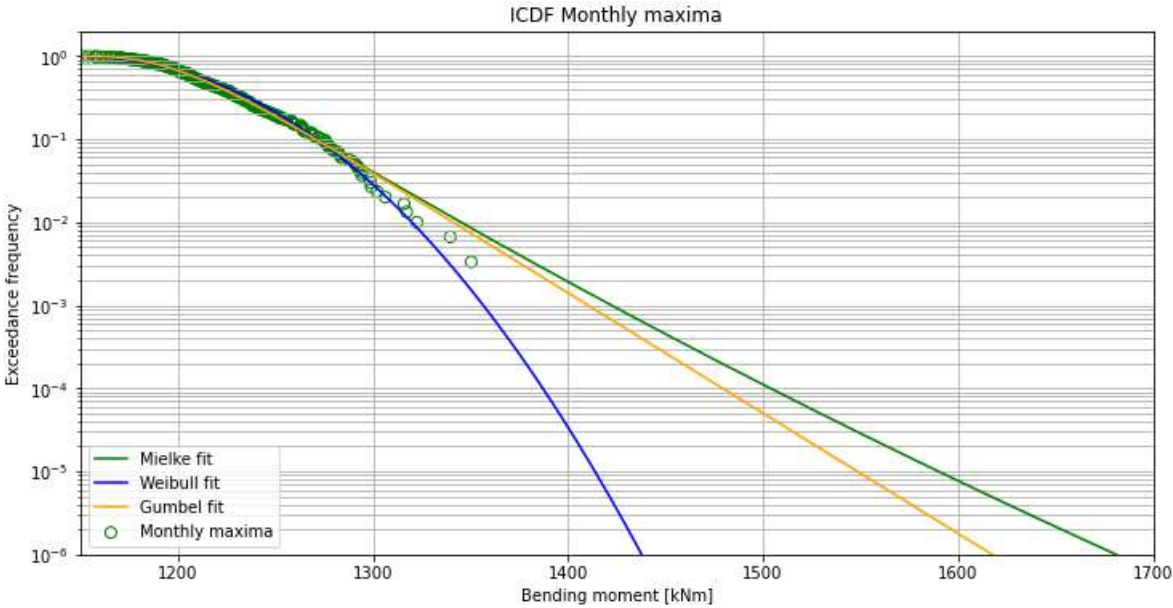


Figure 47 - Fitted inverted cumulative distribution functions for monthly maxima for the modified LP load model.

To determine the yearly maxima distribution, 1000 years of sampling monthly maxima has been done, storing each yearly maximum. Similarly, as for the monthly maxima, the *distfit* package has been used to determine the best fitting distribution to the obtained data. The results are shown below.

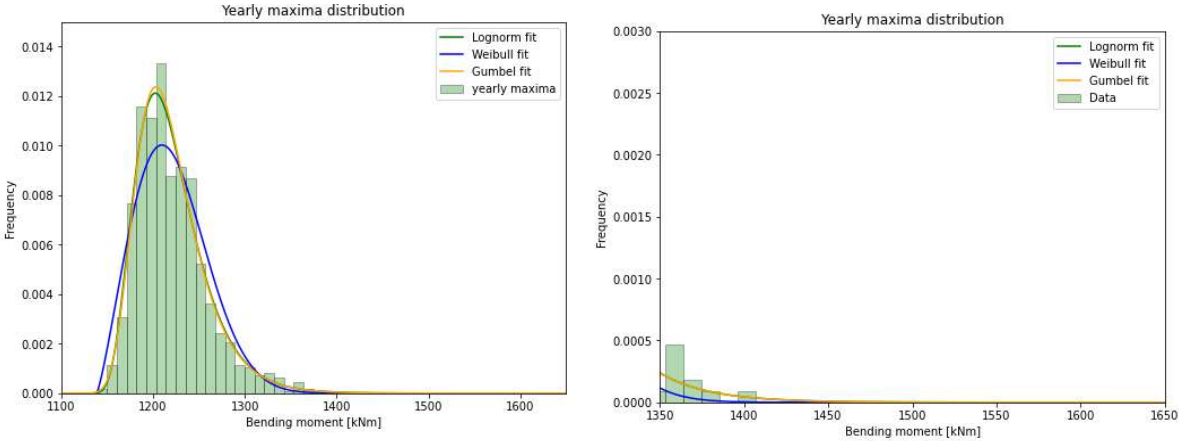


Figure 48 - Relative frequency histograms and fitted distributions for yearly maxima for the modified LP load model.

Visually, the fitted Weibull distribution does not properly fit the data. The *distfit* package determined that a lognormal distribution fitted the data best. However, when inspected, it turned out that the lognormal- and the Gumbel distribution were almost identical. Hence a Gumbel distribution is used to the fit the data to the monthly maxima. This is further strengthened by the figure below.

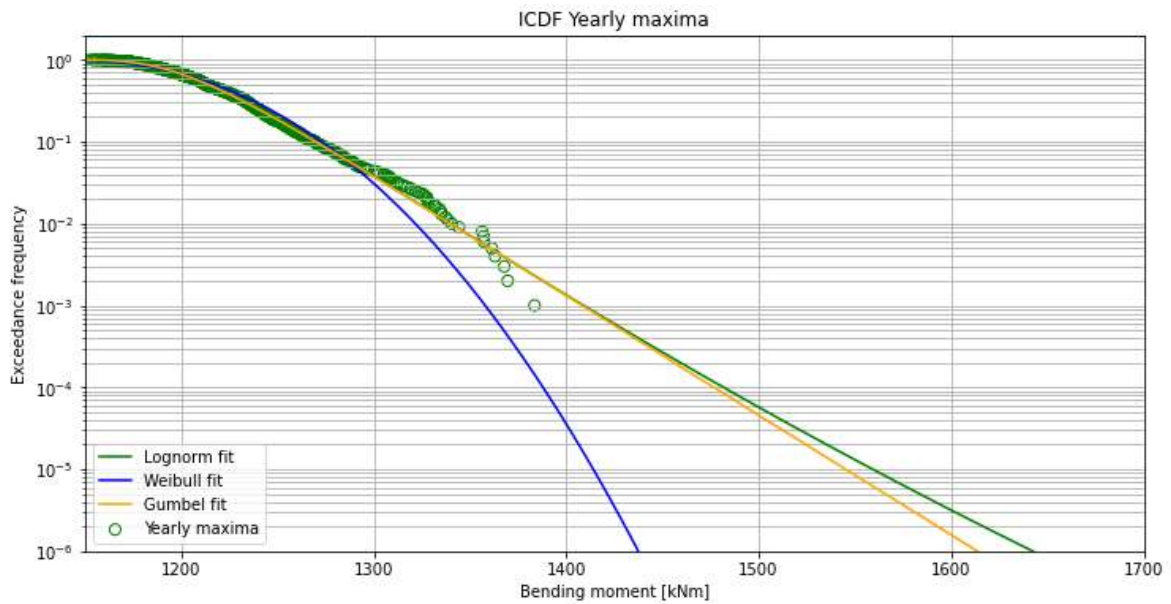


Figure 49 - Fitted inverted cumulative distribution functions for yearly maxima for the modified LP load model.

Based on the facts that the difference between both distributions is negligible and that the monthly maxima distribution is also a Gumbel distribution, the fitted distribution for the yearly maxima is set to be a Gumbel distribution with the following parameters<sup>22</sup>. This function will act as the input for the Monte-Carlo simulation in chapter 7.3.3.

Parameter	Abbreviation (code / math)	Value
Location	<i>loc</i> / $\mu$	1227.9312247878345
Scale	<i>scale</i> $\beta$	28.938224976728158

Table 12 - Parameters for yearly maxima distribution.

### 7.3.2 Resistance distribution R(x)

Paragraph 6.2.2 described how the resistance distribution R(x) was determined. Following from the initial design stated in paragraph 7.2.2, the following input parameters for the function MCsim will be selected. These input parameters result in the following resistance distribution R(x). Note that this distribution does not include the uncertainty parameter  $\theta_R$ .

Parameter	Code	Value	Units
$f_{yd}$	Sxx	500	N/mm <sup>2</sup>
n	n	17	—
$\varnothing_s$	d	32	Mm
z	z	405	N/mm <sup>2</sup>
N	e	$1 \cdot 10^6$	—

Table 13 - Input parameters for MCsim function

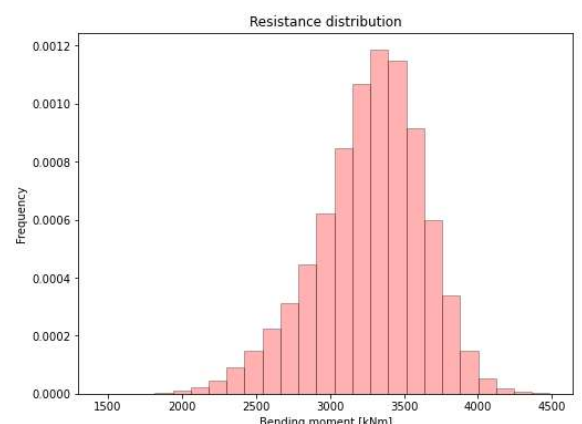


Figure 50 - Relative frequency histogram of distribution of resistance  $M_{Rd}$

<sup>22</sup> This probability density function does not include the model uncertainty  $\theta_S$ .

### 7.3.3 Simulation

The Monte-Carlo simulation will be done by drawing random samples from the load distribution  $S(x)$  and the resistance distribution  $R(x)$ . This will be done  $10^6$  times, which takes about 1 hour of calculation time. During the simulation, the model uncertainty factors  $\theta_S$  and  $\theta_R$  are considered.

### 7.3.4 Results

The figures below show the result of a Monte Carlo simulation for a bridge with a span length of 10 m.

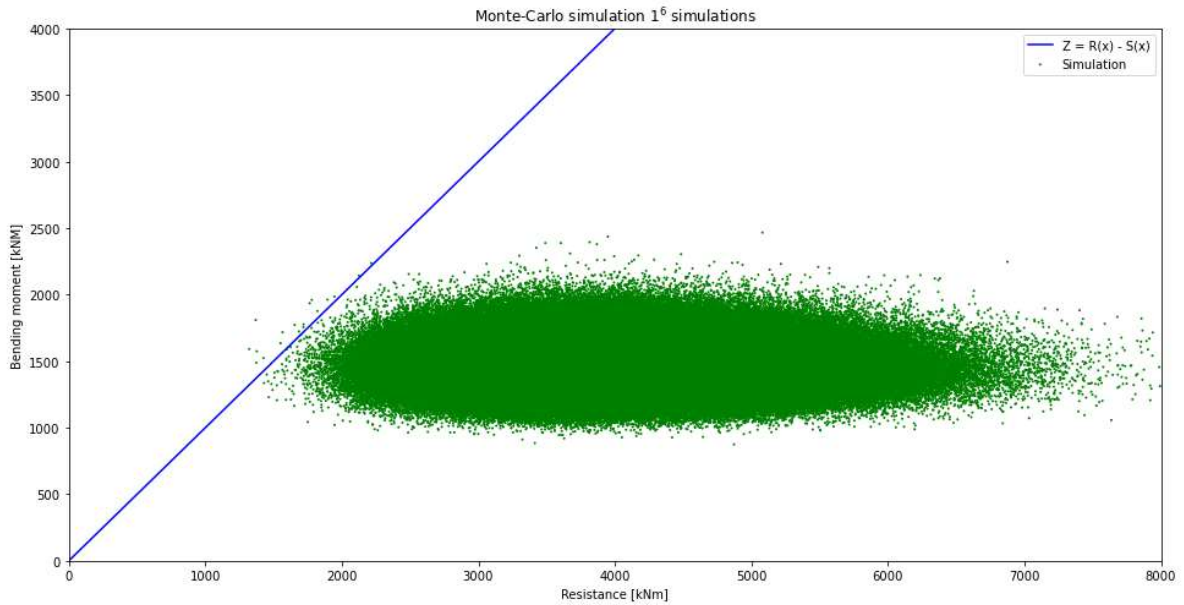


Figure 51 – Results obtained from Monte Carlo simulation using the modified LP model as a load model input.

In total one million simulations were done. As can be seen, the spread of the occurring bending moments is rather small since the distribution of  $S(x)$  is also rather small.  $R(x)$  however, has a rather wide spread, which is mainly due to the model uncertainty factor  $\theta_R$ . This is shown in the figure below.

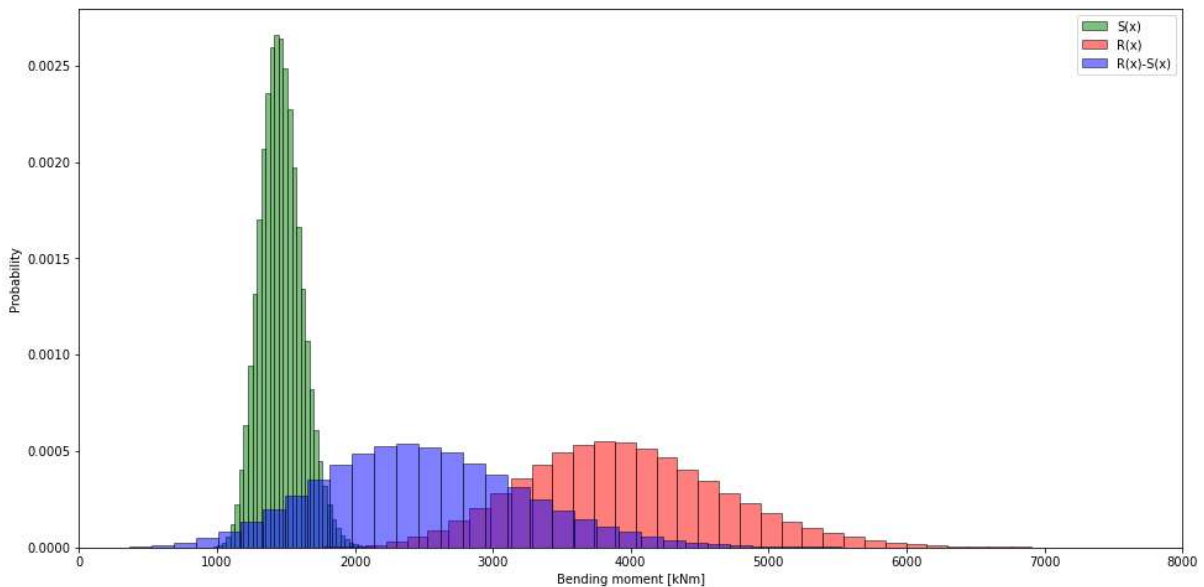


Figure 52 - Distributions for  $S(x)$  and  $R(x)$  with uncertainty factors  $\theta_S$  and  $\theta_R$  incorporated.

In total, 18 failures were recorded. Hence the probability of failure is  $P_f = 1.8 \cdot 10^{-5}$  corresponding to a reliability index  $\beta = 4.13$ .

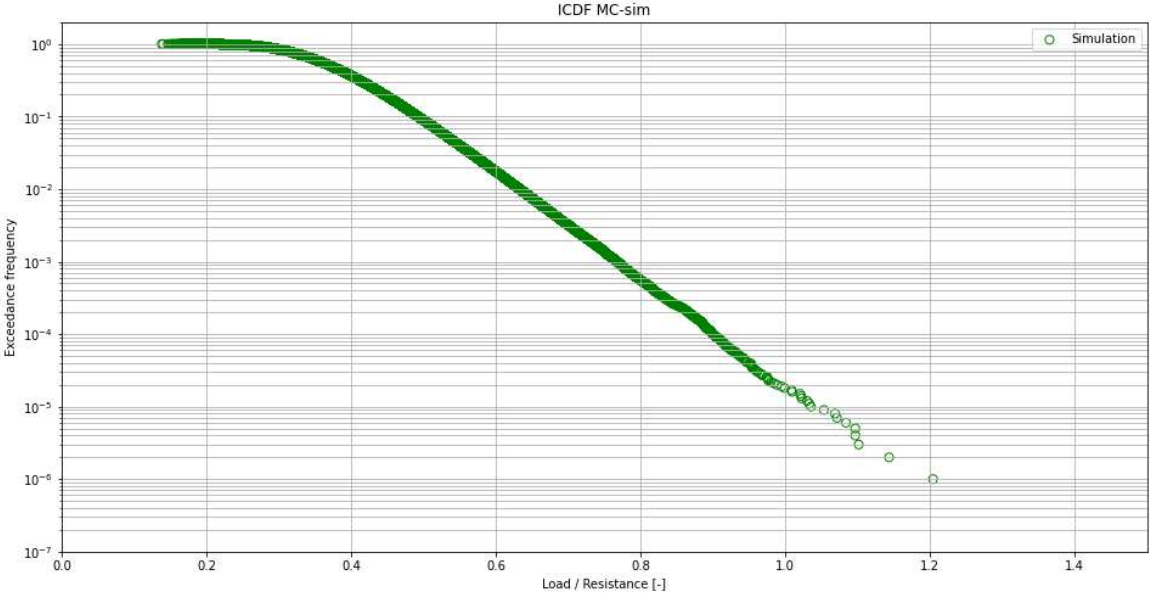


Figure 53 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using modified LP model as a load model input.

### 7.4 Probabilistic verification of various load models

#### 7.4.1 LP data

The dataset used in the previous paragraph comes from the modified LP data model, which incorporated the load factor  $\eta_i$  to simulate under- and overloaded trucks. It is interesting to see what the impact of this load factor is on the probabilistic verification of a similar case study suggested in paragraph 7.3. The same initial design values and the same procedure are used. Since no overloaded axles are simulated, the expectation is that the probability of failure will decrease. The procedure to determine the load distribution  $S(x)$  is elaborated in appendix F. The figure below shows the final load- and resistance distributions  $S(x)$  and  $R(x)$ .

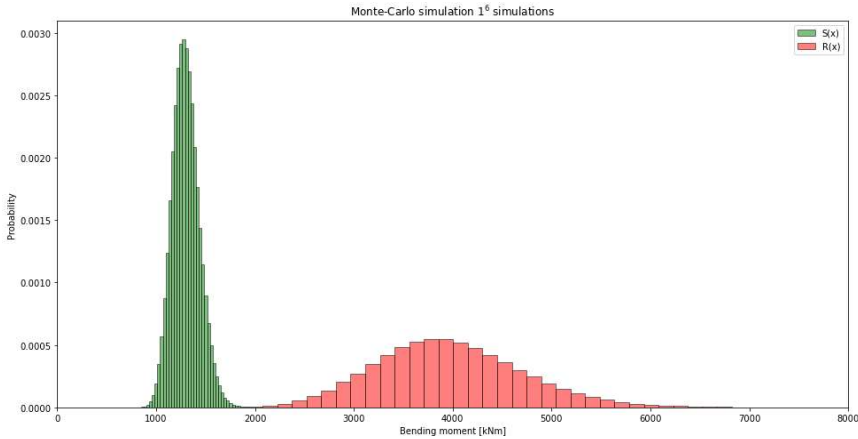


Figure 54 - Probability density functions for  $S(x)$  and  $R(x)$  used in the Monte Carlo simulation.

A Monte-Carlo simulation is done using these two distributions as an input. In total,  $10^6$  simulations were done. The figure below shows the results of the Monte-Carlo simulation. As can be seen, the spread of the resistance is large, and the spread of the load is rather tight. In total, the system only

failed once, i.e.  $P_f = 10^{-6}$ . According to NEN-EN 1990 Annex C, this results in a reliability index  $\beta = 4.75$ .

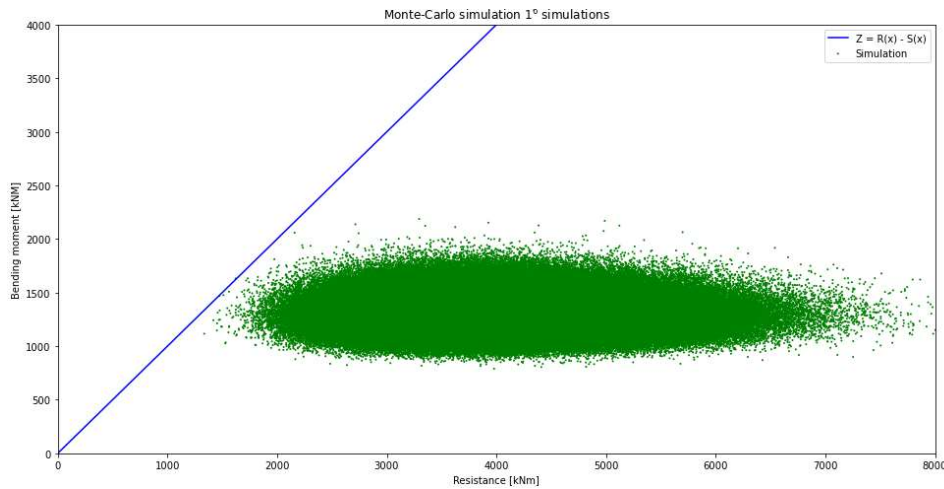


Figure 55 - Results obtained from Monte Carlo simulation using the LP model as a load model input.

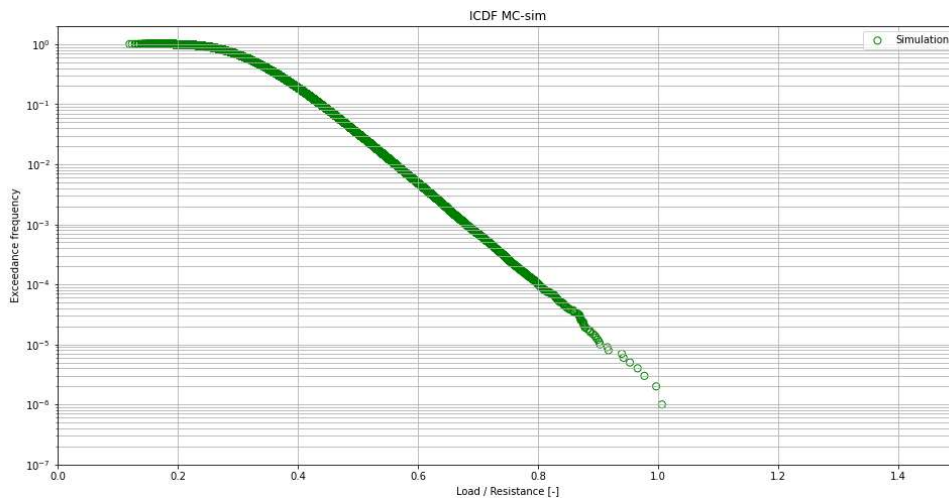


Figure 56 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using LP model as a load model input.

The bottom figure shows the course of the inverted cumulative distribution for the obtained Monte Carlo results. It is possible to fit a curve to this distribution similarly as done in paragraph 7.3 for bridges with the same span and same traffic load to skip calculation time.

#### 7.4.2 WIM data

The modified LP data model is a combination of both the LP data and the WIM data. It is therefore of interest see whether different load models lead to different probability of failures, and thus different reliability indices. In this paragraph, the WIM data is used as a load model input for the probabilistic verification of the same case study as stated in paragraph 7.3. Only the load distribution  $S(x)$  will change accordingly. This process is captured and explained in appendix G. The resulting distributions  $S(x)$  and  $R(x)$  are shown below.

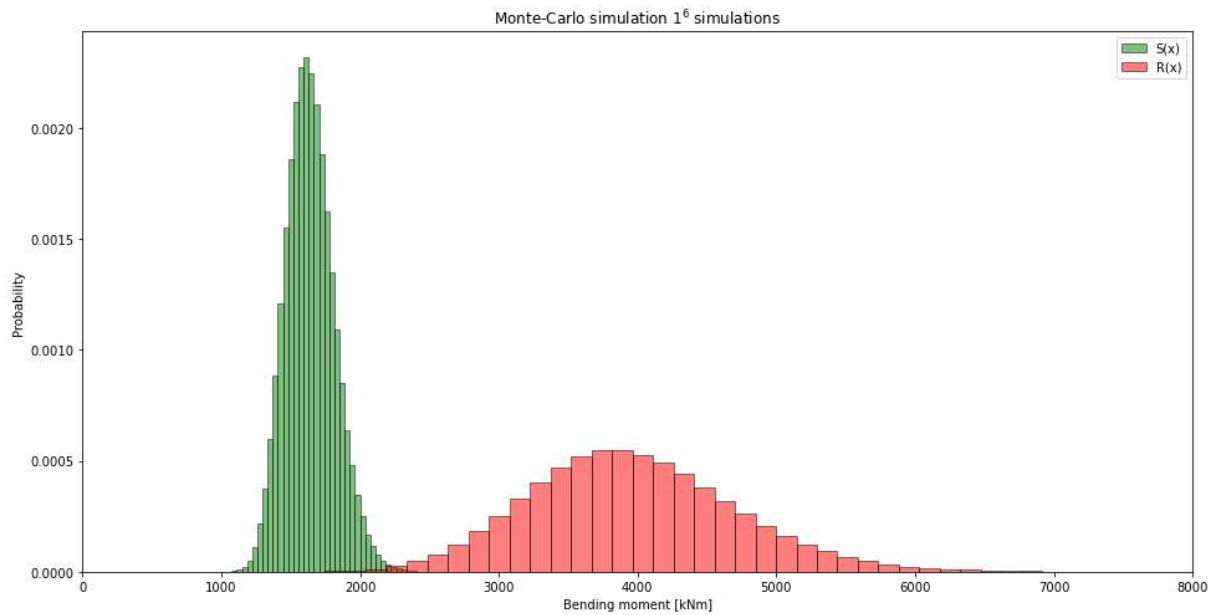


Figure 57 - Probability density functions for  $S(x)$  and  $R(x)$  using the WIM model as a load model input.

A Monte Carlo simulation is done using these two distributions as an input. In total,  $10^6$  simulations were done. The figure below shows the results of the Monte-Carlo simulation. As can be seen, the spread of the resistance is large, and the spread of the load is rather tight, but less tight than for the LP-model. In total, the system failed 129 times, i.e.  $P_f = 1.29 \cdot 10^{-4}$ . According to NEN-EN 1990 Annex C, this results in a reliability index of  $\beta = 3.65$ .

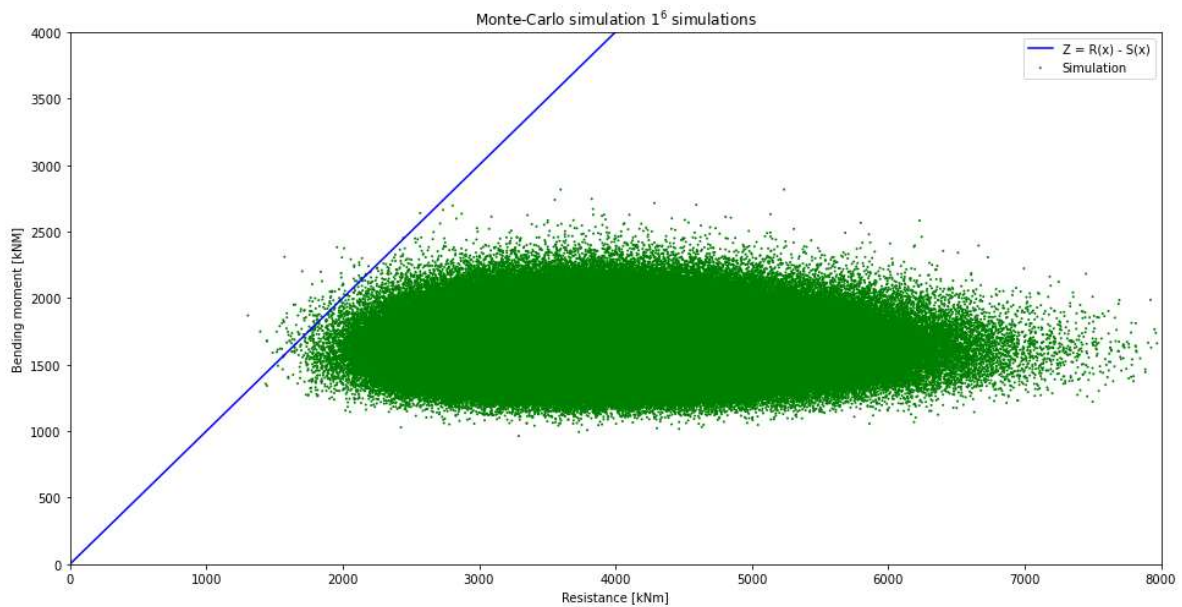


Figure 58 - Results obtained from Monte Carlo simulation using the WIM model as a load model input.

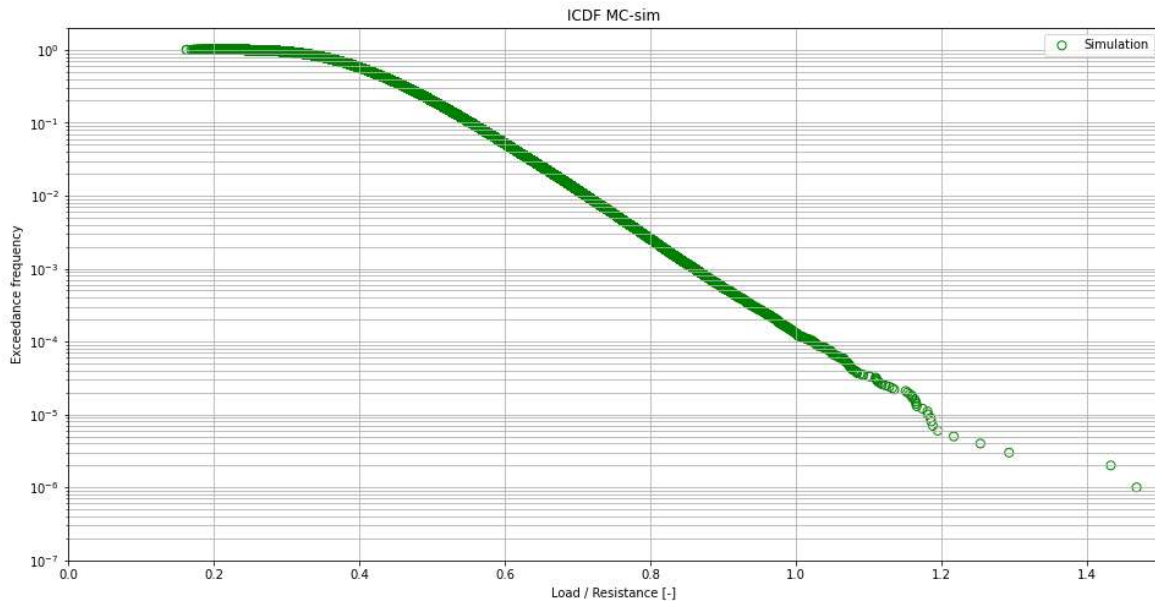


Figure 59 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using WIM model as a load model input.

## 7.5 Dutch National Annex – Reliability index

Whenever a probabilistic verification of an existing structure is not possible for whatever reason, there exists a deterministic way to determine whether an existing structure still fulfils the safety required for a predetermined remaining design working life. This approach is stated in structural codes NEN8700 and NEN8701. In this approach, additional partial factors are applied to increase the design resistance values and to decrease the design load values. The values of these partial factors are based on a certain reliability level. For structures falling in the CC3 category with a reference period of 15 years, a threshold value for the reliability index must be  $\beta = 3.3^{23}$ , according to NEN8700 B3.2 table B.2. When structures fulfil this requirement, structural measurement do not have to be taken during the remaining working life.

## 7.6 Different span lengths

Until now, the only considered span length was 10 m. Paragraph 5.6 has shown that the cause for the highest load effect shifts from overloaded axles to heavier vehicles. The question rises whether there is a relation between span length and reliability index. Therefore, different single span lengths will be considered in this paragraph. Only the span length changes. The same recorded vehicles, including their respective  $\eta$  factors, are applied to the structure. The same procedure as stated in paragraph 7.3 will be used.

### 7.6.1 Initial design

In this paragraph different span lengths will be evaluated using the same approach as in paragraph 7.3. The initial design for the multiple span lengths will be kept the same as much as possible. Table 14 shows the starting parameter input. All other parameters are kept the same as described in paragraph 7.2.2.

<sup>23</sup> For existing structures built before 2003, where wind loading is not dominant.

Span Length [m]	Rebars	Thickness $h$ [mm]	concrete	Lever arm $z$ [mm]	$M_{Ed}$ [kNm]	$M_{Rd}$ [kNm]
L = 5	7Ø32	500		405	950	991
L = 10	17Ø32	500		405	2360	2409
L = 15	21Ø32	700		567	4150	4165
L = 20	32Ø32	700		567	6069	6347

Table 14 - Parameter input for different span lengths

### 7.6.2 Load distributions $S(x)$

The same procedure described in paragraphs 7.3.1 and 7.3.2 will be adopted, although not as thoroughly. The same *modified LP data* model is used to calculate load effects for the different span lengths, also briefly shown in paragraph 5.6. For each span length, to determine  $S(x)$ , the suggested best fitted distribution by the *distfit* package will be selected and the parameters will be determined accordingly. For  $R(x)$ , the exact same procedure will be applied, however with the parameter input shown in table 15. The table below shows the relevant parameter input for  $S(x)$ .

$S(x)$		
Span length	Distribution	Parameters
5 m	Gumbel	$\mu = 523.200$ $\beta = 26.708$
15 m	Generalised logistic	$shape = 6.386453$ $loc = 2129.676299$ $scale = 34.0627$
20 m	Gumbel	$\mu = 3365.841449$ $\beta = 63.268484$

Table 15 - Distribution parameters for different span lengths

The figure below shows all load distributions for the different span lengths. As the span length increases, so does the spread of the load. The distribution for span length  $L = 5$  m is rather tight, which is obvious since the span is likely to be loaded by only one or two (overloaded) axles. As the span length increases, so does the number of axles that load the structure. Hence, more spread in the load effect is observed.



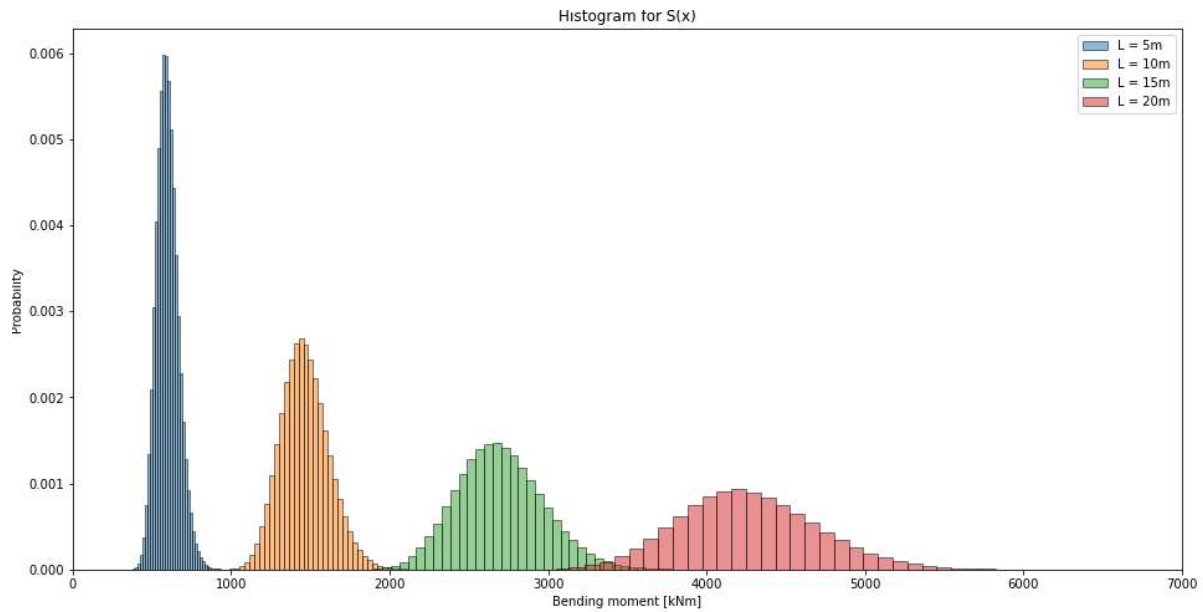


Figure 60 - Overview of probability density functions for  $S(x)$  for different span lengths.

### 7.6.3 Simulation

The Monte-Carlo simulation will be done by drawing random samples from the load distribution  $S(x)$  and the resistance distribution  $R(x)$ . This will be done  $10^6$  times, which takes about 1 hour of calculation time for a mid-range laptop. During the simulation, the model uncertainty factors  $\theta_S$  and  $\theta_R$  are considered.

### 7.6.4 Results

The figure below shows the Monte-Carlo simulation results for all four span lengths. In total,  $10^6$  simulations were done per span length. The results are shown in the figure below.

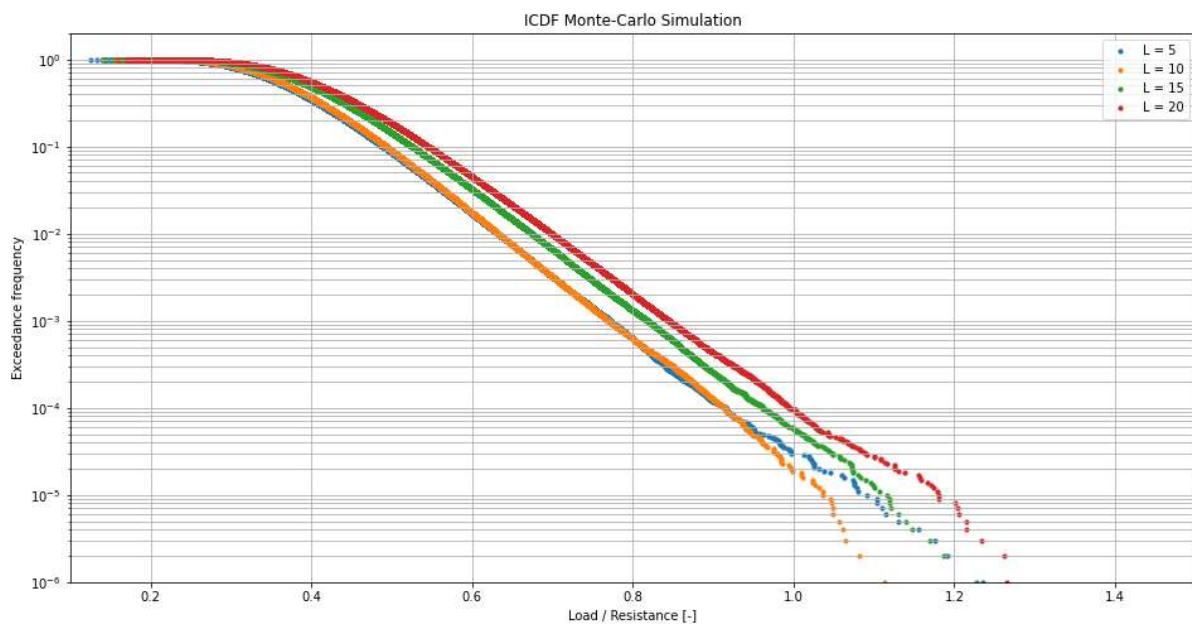


Figure 61 - Inverted cumulative distribution function for results obtained from Monte Carlo simulation using modified LP model as a load model input for different span lengths.

The figure shows the exceedance frequency for the probability of the different span lengths. Remarkably, the span length of 10 m has a lower probability of failure than span length of 5 m. Each span shows a similar pattern for the exceedance frequency up to failure i.e., ( $Z(x) = 0$ ). Based on this figure, one can expect that intermediate span lengths follow the same course as shown in the figure.

The table below shows for each span length  $P_f$  and  $\beta$ . All reviewed span lengths fulfil the requirement regarding the reliability index.

Span length	$P_f$	$\beta$
L = 5m	$2.90 \cdot 10^{-5}$	4.0
L = 10m	$1.90 \cdot 10^{-5}$	4.1
L = 15m	$5.80 \cdot 10^{-5}$	3.8
L = 20m	$9.60 \cdot 10^{-5}$	3.7

Table 16 - Probability of failure and reliability index for different span lengths.

## 7.7 Summary of results

This paragraph captures the most essential obtained results from the different studies. In total, six different studies have been done, ranging from different load model input to different span lengths. The resistance distribution is based on the JCSS Probabilistic Model Code with parameters based on an initial design based on the NEN-EN 1992. The key outcome of these studies is the obtained reliability index  $\beta$ , which is used to determine the safety of a structure for the (remaining) design working life. The table below shows the results obtained for the different studies.

Span length	Load input	$P_f [10^{-5}]$	$\beta$
10 m	LP model	0.1	4.8
10 m	WIM model	12.9	3.7
10 m	Modified LP model	1.90	4.1
5 m	Modified LP model	2.90	4.0
15 m	Modified LP model	5.80	3.8
20 m	Modified LP model	9.60	3.7

Table 17 – Overview of results

All different studies show that a reliability index of at least  $\beta = 3.7$  is obtained. As the span increases, so does the distribution of the load input, as shown in figure 57. When applying the LP model, the highest reliability index is achieved. The modified LP-model and the WIM-model generate the same reliability index. When considering different span lengths, a similar course for the probability of failure is shown.

## 7.8 Intermediate conclusion

The results in this chapter have shown that the modified LP model, which is based on WIM data, result in a slightly more favourable reliability index than the WIM model. The difference however is 11%. This difference can be explained by not accurately describing the WIM model well enough through the modified LP model. This can be obtained by either adjusting the load factor  $\eta_{F,i}$  or the interspatial axle distances  $w_i$ .

The unmodified LP model, based on solely the legally allowed axle loads, results in an unrealistically high reliability index. This is due to not incorporating overloaded axles. Also, the relative static value of the axle loads results in a very tight spread of generated load effect, thus decreasing the probability of an exceedingly high occurring load effect. These two factors massively influence the load distribution and as the resistance distribution remains the same, the probability of failure drastically decreases, resulting in an unrealistically high reliability index.

Although briefly shown in chapter 5, but profoundly proved in this chapter, the addition of the load factor  $\eta_{F,i}$  in the regular LP model results in a realistic spread of the generated load effect. Altogether, the adopted strategy resulted in comparable results with the WIM model.

## 8 Conclusion

In this thesis the applicability of using a load model based on license plates as a load model input for probabilistic analysis was investigated. The results obtained throughout this thesis are used to draw conclusions for the stated objectives and goals. These will be answered in their stated chronology.

*Objective 1 – Understand mechanism of current load models.*

The current prescribed load models in the Eurocode are deterministic in nature. The Eurocode does prescribe the use of a probabilistic load model, but requires relevant site-specific data, such as WIM data. If the structural safety of an existing bridge needs to be verified, additional documents such as the NEN8700 and the NEN8701 to the Eurocode are available. An example of the difference between the results using the different prescribed load models are shown in paragraph 7.7

*Objective 2 – Achieve a detailed insight in current traffic loadings on bridges throughout the inner city of Rotterdam.*

Firstly, only one camera location was considered in this thesis. However, a detailed approach is given which can be easily be applied to investigate multiple camera locations. Secondly, the system that initially provides legal and technical information about passing vehicles for this thesis cannot directly be used as an input for constructing load models, since actual license plates are not registered as shown in paragraphs 3.1.2 and 4.2. However, by making use of reversed engineering it was possible to couple license plates based on the following set of criteria: vehicle length, gross vehicle weight, wheelbase, and kerb mass. Accompanying axle loads were found. For interspatial axle distances no information was found and thus several assumptions were made. This resulted in the draft load model, called LP model.

When comparing to available WIM data from nearby locations, it turned out that static values of the axle loads as given by the RDW were not properly describing the observed range of values as given by the WIM data. Adaptations were made as shown in paragraph 5.3 and 5.4. Moreover, an axle load factor  $\eta$ , as shown in paragraph 5.4, was used to introduce under- and overloading in the draft load model, resulting in the modified LP model. This model was ultimately used in a probabilistic analysis of a hypothetical single slab span bridge as shown in chapter 7. The other load models, LP model and WIM model were also used as a load model input. The results were 4.1, 4.8 and 3.7 respectively.

*Objective 3 - Case study – Application of the defined load model to an existing bridge in Rotterdam and compare the results*

This objective was achieved in chapter 7, where the modified LP model was successfully used as a load model input for a reliability assessment of a hypothetical single span bridge. Resulting in a reliability index of 4.1. The WIM load model and the LP load model were also used as a load model input, resulting in reliability indexes of 4.8 and 3.7 respectively. When compared to the required reliability index of 3.7 complying with reference period of 25 years, all three load models can be used as a load model input. The modified LP model results in a slightly more favourable reliability index than the WIM load model.

As discussed, all three objectives are successfully researched and are answered accordingly. The initial load model, the LP model, turned out to result in far too optimistic reliability results. When adapted the load model using the axle load factor  $\eta_i$ , it turned out that a less optimistic result was obtained regarding the structural reliability and drifted more towards results obtained solely by using the load model based on WIM data. Hence, for camera location 4400-C it was possible to determine a load model based on camera registrations and available WIM data. It must be stated that this was only possible by applying WIM data. Therefore, the strategy shown in this thesis can be applied to construct a load model based on LP- and WIM data for this specific location. For other locations however, this conclusion cannot be drawn since there was no adequate comparison done of the qualitative effect of the used WIM data for other (nearby) locations. This will be further elaborated in the chapter *Discussion* and *Recommendations*. Furthermore, there are few points of discussion regarding this strategy, which will also be discussed in the chapter *Discussion*.

Furthermore, the results have also led to the following conclusion, which was initially not part of the research goal or objectives. For short bridges, overloaded axles influence the load effect greatly as opposed to the GVW for longer span bridges. When solely using the LP load model as a load model input, overloaded axles are not properly accounted for and thus the presumed load effects is underestimated. This prompts the need of applying the axle load factor  $\eta_i$  for short span bridges. This effect is already stated in the Eurocode by applying LM2, where a single axle with a total of 400 kN can be applied anywhere on the bridge deck<sup>24</sup>.

---

<sup>24</sup> Source: <http://bridgedesign.org.uk/tutorial/eu-load-models.php>

## 9 Discussion

During this research, several assumptions were made that can be regarded as controversial if not addressed properly. For each subsection, several points of discussion are made.

### 9.1 Source of data

In this subsection points of discussion are made regarding the source of the data. At first the camera data will be discussed. It must be noted that over in 30% , the camera system could not recognize the license plate of the vehicle, but it did register passing vehicle. Two main reasons for this are foreign license plates, which are not provided in the RDW database, and bad weather conditions. Paragraph 4.2.3 shows that more trailers than tractors were recorded, which can be explained by the fact that Dutch trailers were towed by foreign tractors.

Furthermore, due to privacy reasons, the actual recorded license plates were not stored and therefore were not used. Recorded information coupled with license plates were obtained through a bypass-system, which is thoroughly explained in 4.2.3. A consequence of this is that perhaps falsely registered axle loads are put in the LP load model. However, the range of axle loads is rather narrow, as shown in paragraph 7.4.1 and appendix F.

For both the WIM- and LP data it must be noted that in reality sometimes trucks have one of their axles lifted to prevent unnecessary wearing of the tires if the vehicle is not fully loaded. For the WIM this might result in wrongly classified vehicle types<sup>25</sup>. For the LP data this results in an overconservative assumption that a vehicle always has fully loaded axles.

As mentioned in the conclusion, the source of the WIM data especially is governing for the applicability of the suggested approach mentioned in this thesis. In this particular case, the load effects caused by the WIM data as well as the LP data minorly differ. Hence, the axle load factor  $\eta_i$  was introduced. However, at this point it is unknown whether similar results were obtained when using a different set of WIM data. The comparison between WIM data and LP data therefore is qualitative and the considered camera location should be comparable to the location where the WIM data was recorded and vice versa. This is a flaw at adopting the approach in this thesis to multiple locations throughout the city of Rotterdam. The results obtained in this thesis might be coincidentally optimistic and therefore multiple locations should be considered. This is further described in the chapter *Recommendations*.

### 9.2 Coupling license plates and transforming data to load models

In this subsection points of discussions are considered regarding the step of coupling license plates and transforming the data to load models. As already stated in paragraph 4.2 and 5.1, the actual recorded license plates were not available and thus a bypass system was used, shown in paragraph 4.2.3. A minor flaw of this bypass system is that the criteria used in paragraph 4.2.3 sometimes resulted in multiple hits for one registered vehicle, hence also resulting in a possible range of values for the individual axle loads. Anecdotally these ranges were checked, and in no case very different results were found. The impact of this on the calculated load effect and furthermore on the results shown in paragraph 7.7 is assumed to be minor.

The next point regards the interspatial axle distances for 2+ axled vehicles. As already mentioned in paragraph 4.2.3, no interspatial axle distances were given by the RDW database and thus an assumption for the axle configuration, and thus the interspatial axle distances, was made. These initial assumptions were based on intuition and common sense and were not based on a scientific way of

---

<sup>25</sup>For example, if a 4-axled vehicle has one axle lifted, it is registered as a 3-axled vehicle.

determining axle distances. A more scientific way to determine a range of possible axle distances is to compare similar vehicle types within the WIM data, then compute the distribution for all axle distances and then use this distribution to randomly draw axle distances for the registered vehicles accordingly. This would probably result in more realistic distribution for the axle distances; however the increase of practical significance is assumed to be low.

The final point regards the values of the used axle loads in the WIM-, LP- and modified LP model. The values of the applied axle loads are assumed to be static and therefore do not deviate whilst a vehicle travels across a bridge span. In NEN-EN 1992-1 the use of a deterministic dynamic amplification factor, DAF, is suggested to account for this scenario. This DAF factor was not applied throughout this thesis. When do considered, the exerted load effect will logically increase, and therefore the assumed load distributions in the Monte-Carlo simulations will change as well. For the resistance distributions  $R(x)$  in the Monte-Carlo simulations nothing changes, hence the prediction will be that the reliability index will decrease. In case the reliability index does not reach the threshold value  $\beta = 3.7$  as stated in paragraph 7.5, a new qualitative comparison must be done whether the use of the WIM data is applicable in this case.

### 9.3 Generated load model and application

This subsection handles points of discussion regarding the load model and the application of the obtained load model. The most important part of this thesis is the practicality of the suggested method. As it turned out, for location 4400-C, the suggested method was applicable. However, this was mainly due to the use of the available WIM data and the assumption that the vehicles registered for location 4400-C and the vehicles registered in the WIM data were comparable.

If applying the method suggested in this thesis, which implies incorporating  $\eta_i$  based on WIM data, the case should always be that the WIM data is representative for the considered camera location. In this case, location 4400-C is close to the highway A13. One could imagine that if WIM data from a suburban area is used, a totally different result will be obtained. This results in that this strategy is only applicable if WIM data from comparable locations, i.e. roads are available.

Furthermore, general remarks must be made about the generated load model. As can be seen throughout chapters 5 and 6, all generated bending moments are caused by only one vehicle. During the Monte-Carlo simulations only one value is extracted from the load distributions  $S(x)$  derived in paragraph 7.3. This is a minor flaw in the suggested approach, since in reality multiple vehicles can simultaneously cross a bridge, driving in the same or opposite directions. Also, traffic congestions are not taken into account. However, as stated in TNO [12], traffic congestion would only be normative larger span bridges. This thesis focusses more on the load effect on smaller span bridges.

## 10 Recommendations

This final chapter of this thesis will be about recommendations for further studies regarding this subject. In this study, a load model was constructed based on data coupled to license plates and on a combination of license plate data and WIM data. The study that this study is based on<sup>26</sup>, did not contain that information and so more knowledge is gained about this subject.

As mentioned before, the approach suggested in this study could use some improvements regarding the applicability to multiple bridge locations. As already mentioned, only the data of one camera location was used to construct the LP model and the WIM data of two other locations were used to construct the modified LP model. The drawn conclusion shows that qualitatively the results between the LP model and the WIM model were comparable, as illustrated in figure 27, and thus the modified LP could be constructed for this specific location, i.e. location 4400-C.

However, to construct a load model that is applicable to multiple bridges, further research must be done. At least the following parts of this further research should be covered. Firstly, compare generated load effects for multiple camera locations to have a better understanding of traffic loads within different areas of the city of Rotterdam. Secondly, a qualitative comparison between each camera location and the WIM data should be done, i.e. make similar figures like figure 27 for all considered camera locations and verify whether the results are comparable, i.e. verify whether the use of WIM data as input is reasonable.

Based on the outcome of this further research, possibly certain areas or roads in the city of Rotterdam could be categorized based on load effects. The WIM data can then be used accordingly as a parameter input to determine axle load factors  $\eta_i$  for certain areas or roads.

Another point of improvement can be made in the Monte-Carlo simulation regarding the resistance distribution  $R(x)$ . In this study,  $R(x)$  is constructed by following the suggestions stated in the JCSS Probabilistic Model Code. This indicates that  $R(x)$  is a theoretical distribution that is not representative for any considered location, in this case location 4400-C. As the reliability index  $\beta$  is based on the input for  $R(x)$ , whenever a different input is used,  $\beta$  will change as well. Hence, if concrete and reinforcement parameters for any specific bridge is known, the results for  $\beta$  are more accurately described and site-specific load distribution  $R_i(x)$  can be constructed. Previously done research has already been done to the actual strength of the concrete, and it has been shown that  $f_{ck}$  is higher than expected<sup>27</sup> which strengthens hypothesis that this is a viable strategy. This can be further researched by taking samples of existing bridges to determine concrete and reinforcement steel parameters and verify whether site-specific load distributions  $R_i(x)$  can be constructed.

A third recommendation is to consider other generated load effects by traffic loads. This study only considered load effects in the form of bending moments. However, some bridges might be prone to shear failure and thus need to be considered separately. This is especially the case for existing bridges in highways in the Netherlands<sup>28</sup>. With minor modifications, the method to calculate bending moments can be adapted to calculate shear forces generated by traffic loads, thus resulting in load distribution  $S(x)$  for shear force, indicated with  $S_v(x)$ .

---

<sup>26</sup> Study done by Hellebrandt [1].

<sup>27</sup> <https://www.betoniek.nl/betonsterkte-bestaande-bruggen-blijkt-veel-hoger>

<sup>28</sup> <https://www.cementonline.nl/artikel/aanpak-dwarskrachtproblematiek>



To finalize, a study to the use of the suggested load model regarding asset management could be done. As mentioned in paragraph 2.2 and 2.3, where certain bridges in the city of Rotterdam might need structural reinforcement, others do not. If solely following the suggested approach in the Eurocode, a structure does either comply or not. This will sometimes result in unnecessary structural reinforcements, that might also have been able to be postponed. However, the load model suggested in this thesis, at least for location 4400-C, gives slightly more realistic results, thus the need of structural reinforcement could be postponed. This is especially viable for municipalities with many structures in their asset management, where the need for economical decisions regarding optimal timing of applying structural reinforcements is dependent on a cost effective analysis.

## 11 References

1. L. Hellebrandt, 2014. Structural Reliability of Existing City Bridges: Analysis with Monte Carlo simulation including a load model based on weigh-in-motion measurements. Technical University of Delft. Available at: <http://resolver.tudelft.nl/uuid:95880857-329a-4beb-8619-4724da800a7a>
2. JCSS, 2001. *Probabilistic Model Code*, Available at: <https://www.jcss-lc.org/jcss-probabilistic-model-code/>
3. S.N. Jonkman, R.D.J.M. Steenbergen, O. Morales-Nápoles, A.C.W.M. Vrouwenvelder & J.K. Vrijling, 2017. *Lecture Notes CIE5140: Probabilistic Design: Risk and Reliability Analysis in Civil Engineering*. Technical University of Delft
4. N. Manoj, 2016. *First-Order Reliability Method: Concepts and Application*. Technical University of Delft. Available at: <http://resolver.tudelft.nl/uuid:c4c941fa-a9c1-4bd4-a418-afc54bb6d475>
5. NEN-EN 1990, NEN-EN 1991-2, NEN-EN 1992. <https://www.nen.nl> (02-10-2019)
6. Y. Bouassida et al, 2010. *Bridge Design to Eurocodes Worked examples*. Vienna, Austria.
7. H. Gulvanessian, J.-A. Calgaro, M. Holický, 2002. *Designers' guide to EN 1990 Eurocode: Basisstructural design*. Thomas Telford, London.
8. S. Coles, 2004. *An introduction to statistical modeling of extreme values*. Springer-Verlag, London.
9. <https://www.statisticshowto.datasciencecentral.com/extreme-value-distribution/> (10-10-2019)
10. <https://www.cbs.nl/nl-nl/maatschappij/verkeer-en-vervoer/transport-en-mobiliteit/infra-en-vervoermiddelen/vervoermiddelen/categorie-vervoermiddelen/bestelauto-s-vrachtwagens-en-bussen> (24-3-2020)
11. E. Kuiper, Dr. N.E. Ligterink, 2013. *Voertuigcategorieën en gewichten van voertuigcombinaties op de Nederlandse snelweg op basis van assen-combinaties en as-lasten*. (TNO 2013 R12138).TNO.
12. Vervuurt, A. H. J. M., Pruiksma, J. P., Steenbergen, R. D. J. M., Courage, W. M. G., Miraglia, S., & Morales Napoles, O. (2015, april). *Tussenrapportage herijking verkeersbelastingen voor brugconstructies op basis van WIM analyses van april 2013 (IQ-2014-33c)* (TNO-2014-R11653). TNO.
13. Favier, P., Bertrand, D., Eckert, N., and Naaim, M.: *A reliability assessment of physical vulnerability of reinforced concrete walls loaded by snow avalanches.*, Nat. Hazards Earth Syst. Sci., 14, 689–704, (<https://doi.org/10.5194/nhess-14-689-2014>), 2014.
14. Lerche, I., & Mudford, B. S. (2005). *How Many Monte Carlo Simulations Does One Need to Do?* Energy Exploration & Exploitation, 23(6), 405–427. <https://doi.org/10.1260/014459805776986876>
15. Sedlacek, G. et al., 2008. *Background document to EN 1991- Part 2 - Traffic loads for road bridges -and consequences for the design*.
16. Picture used at title page: <https://www.ad.nl/rotterdam/jarenlang-werden-ze-gedoogd-maar-nu-zijn-vrachtwagens-verbannen-van-de-olympiaweg~a4df053f/>, taken by Frank de Roo©.

## Appendix A – RDW excerpt

In the figures below, information of a 5-axled tipper truck is given. The tipper truck is a GINAF X6 5250 CSE. For privacy reasons, the license plate is crossed out. Only relevant information will be discussed.

GINAF  
X6 5250 CSE

Vul hieronder uw kenteken in:

Gegevens opvragen

**Basis** Motor & Milieu Technisch Fiscaal

- Algemeen
- Vervaldata en historie
- Gewichten
- Tellerstanden
- Status van het voertuig
- Terugroepacties

Figure 62 - Excerpt of available information through RDW

The sheets *Basis* (General) and *Technisch* (Technical) contain relevant information. For sheet *Basis*, the paragraphs *Algemeen* (General) contains general information about the vehicle itself (brand name, vehicle type etc.) and *Gewichten* (Weights) contains all information about masses. The figures below show the paragraphs *Algemeen* and *Gewichten*. The page is automatically translated by Google Translate.

Algemeen			
J	Vehicle category	Company car (N3G)	
	Carrosserietype	Truck (BA)	
	Carrosseriecode	Kipper (10)	
	Design	kipper	
D.1	Brand	GINAF	
D.2	Type	X6 5250 CSE	
D.2	Variant	not registered	
D.2	Performance	not registered	
D.3	Trade name	X6 5250 CSE	
K	Typegoedkeuringsnummer	not registered	
	Place chassis number	Right, right bar at front axle	
	Number of private / business owners	0 / 1	

Figure 63 - Excerpt of section "Algemeen". General information of a 5-axled tipper truck

Gewichten			
G	Mass in running order	19150 kg	
	Mass empty vehicle	19050 kg	
F.1	Technical max. Mass vehicle	52000 kg	
F.2	Permissible max. Vehicle mass	50000 kg	
F.3	Maximum mass assembly	0 kg	
0.1	Trailer braked	0 kg	
	Center axle trailer braked	0 kg	
	Trailer autonomously braked	0 kg	
	Trailer braked	0 kg	
0.2	Trailer unbraked	0 kg	

Figure 64 - Excerpt of section "Gewichten". Information about masses of a 5-axled tipper truck

The sheet *Technisch* contains technical information about the vehicle. The paragraph *Eigenschaften* (Properties) gives general technical information about length, wheelbase and other technical properties. The paragraph *Assen* (Assen) gives technical information about the maximum axle load, stated by law. The figures below show these two paragraphs.



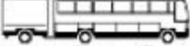

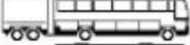

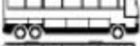

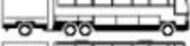

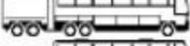
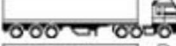
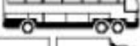
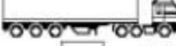
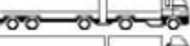

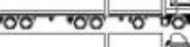

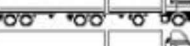

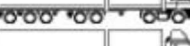

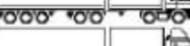

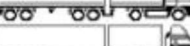

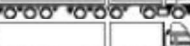
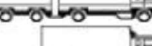
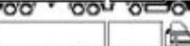

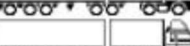

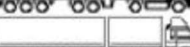
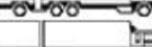

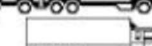
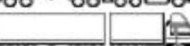

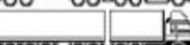
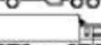
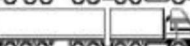


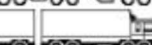



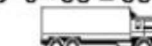

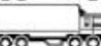









Properties		
	Length	1000 cm
	Width	250 cm
	number of doors	not registered
<b>S.1</b>	number of seats	2
	Number of wheelchair spaces	0
<b>S.2</b>	Number of pitches	0
<b>L</b>	Number of axes	5
	The number of wheels	10
<b>M</b>	Wheelbase	725 cm
	Deviating maximum speed	not registered
	Distance front vehicle to heart link	0 cm
	Load capacity	30950 kg

Figure 65 - Excerpt of section "Technisch". Technical information of a 5-axled tipper truck.

Assen						
No.		1	2	3	4	5
	Driven axle	No	No	No	And	And
	Hefa	No	No	And	No	No
	Shaft code	V	V	A	A	A
	Track gauge	0 cm	0 cm	0 cm	0 cm	0 cm
	Handling code	Not a dish. Not a dish. Not a dish.			G	G
	Technically permissible maximum axle load	10000 kg	10000 kg	8000 kg	12000 kg	12000 kg
	Legally permitted maximum axle load	10000 kg	10000 kg	7000 kg	11500 kg	11500 kg

Figure 66 - Excerpt of section "Assen". Technical information of a 5-axled tipper truck.

## Appendix B - Subclassification of vehicles

WIM-NL vehicle sub-classes		WIM-NL vehicle sub-classes	
sub-class	symbol	sub-class	symbol
B11		T12O1	
B11A1		T12O2	
B11A2		T12O3	
B12		T12O4	
B12A1		T12O11	
B12A2		T12O21	
B21		T12O111	
R11111		T12O1111	
R111111		T21O11	
R11112		V11	
R111121		V11A1	
R11113		V11A2	
R11211		V11A11	
R112121		V11A12	
R1122		V12	
R11221		V12A2	
R1123		V12A11	
R121221		V12A12	
R12211		V13	
R1222		V21	
R12221		V22	
R1223		V22A2	
T11O1		V22A11	
T11O2		V22A12	
T11O3		V111	
T11O4		V112	
T11O11		V211	
T11O21		V1111	
T11O111		O (*others*)	
T11O1111			

Modified: 24-08-2006

Figure 67 -Subclassification of vehicles in the WIM data<sup>29</sup>.

<sup>29</sup> Source: Vervuurt, AHJM., Pruiksmas, JP., Steenberghe, RDJM., Courage, WMG., Miraglia, S., & Morales Napoles, O. (2015). Statische belastingen Herijking verkeersbelastingen voor brugconstructies op basis van WIM analyses van april 2013. InfraQuest.

# Appendix C – Comparison between different vehicle types between LP data and WIM data

## Semi-trucks

In this paragraph, both the 2- and the 3-axled semi-trucks carrying semi-trailers will be discussed. The 2- and 3-axled semi-trucks carrying semi-trailers are categorised in subclasses TT11Oxx, TT12Oxx and TT21Oxx. In a study done by TNO [12, p.24], it was shown that different axle configurations such as tandems, tridems or quads usually one axle is overloaded, and the accompanying axle(s) is underloaded. In the LP data, the values of axle loads are deterministic, whilst in reality they variate in value. This is shown in the figures below.

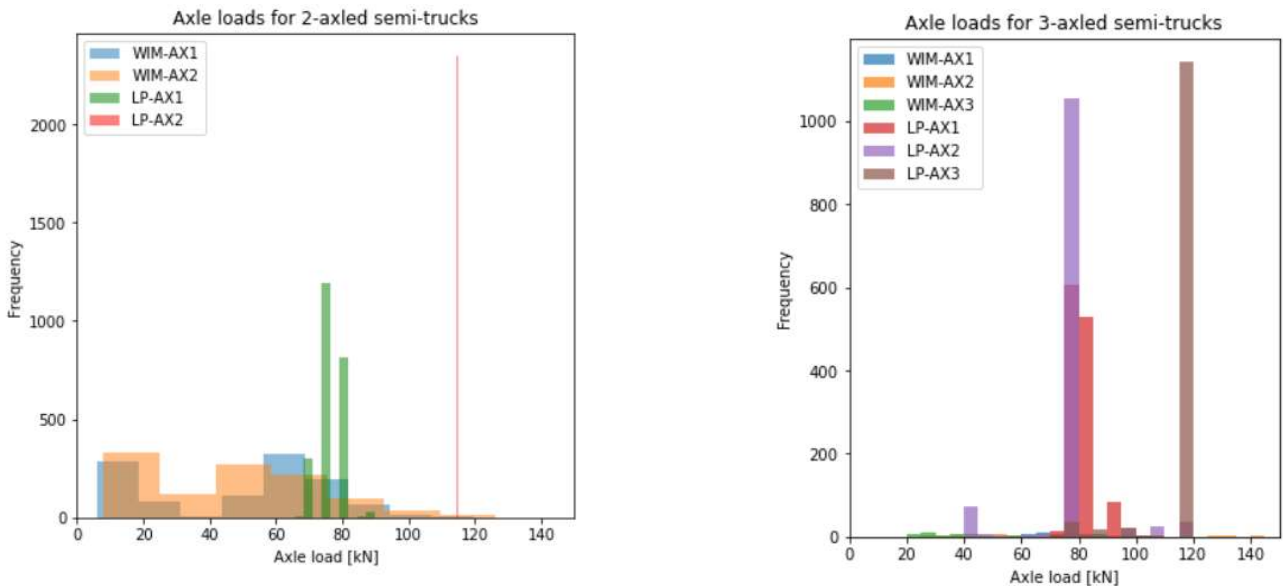


Figure 68 - Frequency distribution of axle loads for 2- and 3-axled semi-trucks for LP and WIM data.

## 2-axled semi-trucks

For 2-axled semi-trucks, the first axle load theoretically can span from 60 kN to 90 kN. The second axle load is theoretically always 115 kN. As can be seen from the graph however, the WIM- axle load distribution is much more comprehensive than the LP-distribution. With most of the axles being underloaded, and only a small portion of both axles being overloaded (not simultaneously). The table below shows a comparison for both the LP- and WIM data regarding 2-axled semi-trucks carrying trailers.

N3 vehicle	O4 axles	LP	WIM		Threshold [kN]	Underloaded	Overloaded
2-axled	1	1137	66	Axle 1	76,15	87,77%	12,23%
				Axle 2	114,92	89,32%	10,68%
				Axle 3	100	95,6%	4,4%
2-axled	2	1045	389	Axle 1	75,71	70,96%	29,04%
				Axle 2	115	88,43%	11,57%
				Axle 3	92,84	98,2%	1,8%
				Axle 4	92,84	97,94%	2,06%
2-axled	3	1236	1980	Axle 1	77,03	68,84%	31,16%
				Axle 2	115	85,45%	14,55%
				Axle 3	89,38	93,39%	6,61%
				Axle 4	89,38	95%	5%
				Axle 5	89,38	95,51%	4,49%

Table 18 - Overview of axle loads for 2-axled semi-trucks carrying 1-,2-, or 3-axled trailers for both LP data and WIM data.

Most of the axles are not overloaded, with a few exceptions for 4- and 5-axled combinations. Noticeably in these cases, the first axle is overloaded most of the time as opposed to the other axles.

The WIM data also recorded 2-axled semi-trucks carrying 4-axled trailers, classified as vehicle class T1104 or T1101111. Although the LP data did not record any vehicles of this kind, it is still useful to review the data. The data is shown in the table below. Instead of the *Treshold [kN]* value, the average weight of each axle is calculated and based on that the ratio under- overloaded is determined.

N3 vehicle	O4 axles	Number		AVG [kN]	Underloaded	Overloaded
2-axled	4	13	Axle 1	68,85	38,46%	61,54%
			Axle 2	52,15	53,85%	46,15%
			Axle 3	32,85	69,23%	30,77%
			Axle 4	23,69	61,54%	38,46%
			Axle 5	23,62	69,23%	30,77%
			Axle 6	23,23	61,54%	38,46%

Table 19 - Overview of axle loads for 2-axled semi-trucks carrying 4-axled trailers for WIM data only.

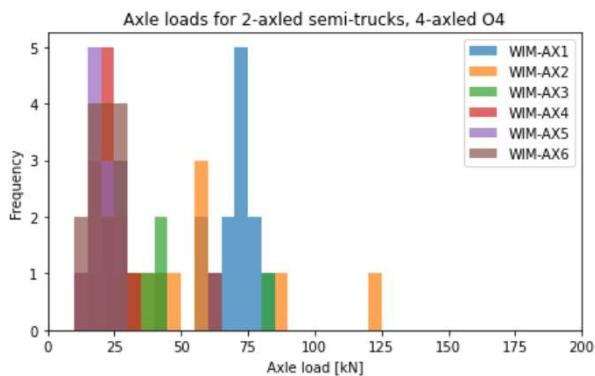


Figure 69 - Frequency distribution of axle loads for 2-axled semi-trucks carrying 4-axled trailers.

In this vehicle category, a lot of axles seem to be overloaded, especially the first two axles. The histogram shows that the 2<sup>nd</sup> axle once is relatively high loaded as opposed to the other five axles.

The table and graph below show the distribution for interspatial axle distances for 2-axled semi-trucks carrying any kind of trailers. Since table 20 shows that 4- and 5-axled combinations are often occurring, only those vehicle combinations will be evaluated and especially the interspatial axle distances that differ the most.

N3	O4 AX	IAD	LP			WIM			LP/WIM		
			AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
2-axled	1	DT12	3,77	3,53	3,9	3,58	1,3	12,77	105%	272%	31%
		DT23	7,1	4,8	9	5,7	1,02	15,69	125%	471%	57%
2-axled	2	DT12	3,79	3,5	4,12	4,19	2,11	6,98	90%	166%	59%
		DT23	6,76	3,07	8,93	5,62	2,01	12,15	120%	153%	73%
		DT34	1,3	1,25	1,35	3,62	1,3	7,41	36%	96%	18%
2-axled	3	DT12	3,78	3,5	5,6	3,8	2	7,51	99%	175%	75%
		DT23	6,35	3,39	8,43	5,56	0,99	9,71	114%	342%	87%
		DT34	1,3	1,25	1,35	1,46	0,75	7,15	89%	167%	19%
		DT45	1,3	1,25	1,35	1,44	0,75	7,1	90%	167%	19%
2-axled	4	DT12	NAN			3,88	3,12	4,1	NAN		
		DT23	NAN			5,83	5,42	6,86	NAN		
		DT34	NAN			1,31	1,27	1,35	NAN		
		DT45	NAN			1,32	1,28	1,36	NAN		
		DT56	NAN			1,32	1,28	1,36	NAN		

Table 20 - Overview of interspatial axle distances for 2-axled semi-trucks carrying trailers for both LP data and WIM data.

- For 3-axled LP combinations: the minimum and maximum values for *DT12* and *DT23* differ quite a lot. However, it is quite likely that different vehicle types are compared.
- For 4-axled LP combinations: the mean value for *DT34* differs quite a lot. This is also likely due to the comparison between wrong vehicle classes.
- For 5-axled LP combinations: the minimum value for *DT23* seem to be overestimated quite a lot. However, this is also likely due to the comparison between wrong vehicle classes.

### 3-axled semi-trucks

For 3-axled semi-trucks a similar pattern is shown. However, because the LP data considering 3-axled semi-trucks greatly outnumbers the WIM data for 3-axled semi-trucks, a very skewed distribution is obtained. Hence in the figure below, the y-axis is cut-off at 100, which clearly shows how the ratio between all axles for the WIM- and LP data differ. Just as for the 2-axled semi-trucks, most axles are underloaded, and only a small portion of the axles is overloaded. However, the overloaded axles can be as high as 193 kN, greatly exceeding the legally allowed axle load.



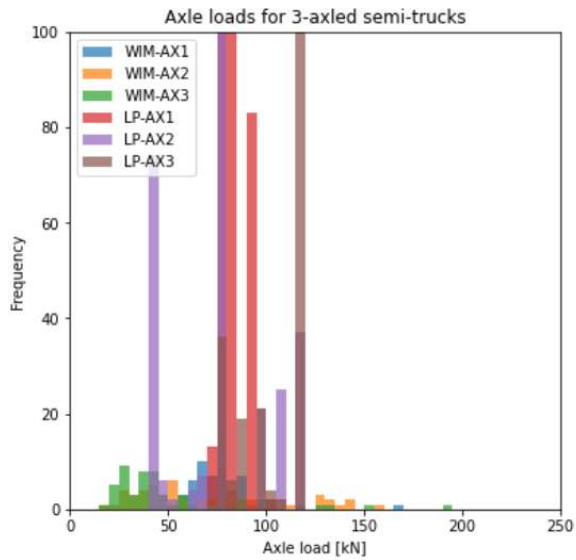


Figure 70 - Frequency distribution of axle loads for 3-axled semi-trucks for both LP data and WIM data.

N3 vehicle	O4 axles	LP	WIM		Threshold [kN]	Underloaded	Overloaded
3-axled	1	37	22	Axle 1	79,46	59,09%	40,91%
				Axle 2	77,32	77,27%	22,73%
				Axle 3	110,134	90,9%	9,1%
				Axle 4	100	81,81%	18,19%
3-axled	2	455	25	Axle 1	78,63	72%	28%
				Axle 2	76,54	52%	48%
				Axle 3	111,34	100%	0%
				Axle 4	92,79	96%	4%
				Axle 5	92,79	100%	0%
3-axled	3	739	6	Axle 1	77,79	83,33%	16,67%
				Axle 2	73,93	66,66%	33,34%
				Axle 3	113,79	50%	50%
				Axle 4	89,34	50%	50%
				Axle 5	89,37	50%	50%
				Axle 6	89,35	50%	50%

Table 21 - Overview of axle loads for 3-axled semi-trucks carrying 1-, 2-, or 3-axled trailers for both LP data and WIM data.

The table and graph below show the distribution for interspatial axle distances for 3-axled semi-trucks carrying any kind of trailers.

N3	O4 AX	LP				WIM			LP/WIM		
		IAD	AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
3- axled	1	DT12	2,75	2,3	3,97	3,46	2,19	5,59	79%	105%	71%
		DT23	1,31	1,25	1,35	3,1	1,29	8,07	42%	97%	17%
		DT34	6,49	3,04	8,5	3,61	1,28	11,9	180%	238%	71%
3- axled	2	DT12	2,79	2,4	3,97	3,81	3,55	4,32	73%	68%	92%
		DT23	1,3	1,25	1,35	5,05	4,34	8,08	26%	29%	17%
		DT34	6,73	3,04	8,5	1,59	1,28	1,99	423%	238%	427%
		DT45	1,3	1,25	1,35	2,27	2	7,77	57%	63%	17%
3- axled	3	DT12	2,73	2,51	3,55	2,53	2,19	2,82	108%	115%	126%
		DT23	1,3	1,25	1,35	1,42	1,29	1,58	92%	97%	85%
		DT34	6,3	3,4	8,29	3,97	1,53	5,81	159%	222%	143%
		DT45	1,3	1,25	1,35	1,81	1,59	1,99	72%	79%	68%
		DT56	1,3	1,25	1,35	2,22	2	2,48	59%	63%	54%

Table 22 - Overview of interspatial axle distances for 3-axled semi-trucks carrying 1-, 2- or 3-axled trailers for both LP data and WIM data.

- For 4-axled LP combinations: the mean values for *DT23* and *DT34* differ quite a lot. This is likely due to the comparison between wrong vehicle classes.
- For 5-axled LP combinations: the mean value for *DT23* and *DT34* also differ quite a lot. This is also likely due to the comparison between wrong vehicle classes. The minimum value for *DT34* however can be the result of wrongly interpreting the axle configuration for a semi-trailer.

A few entries seem to deviate entirely from the WIM data. In particular *DT34* for a 4-axled combination, *DT34* for a 5-axled combination and *DT34* for a 6-axled combination. Property *DT34* describes the length between the last axle of the N3-vehicle and the first axle of the O4-vehicle, thus explaining at least for the LP data, why *DT34* is rather high.

For the WIM data however, the data shown in this table is based on vehicle class solely. So quite likely, the shown data also contains vehicles that aren't semi-trucks carrying semi-trailers, but rather a relatively 'long' tipper truck or a 4-axled N3-vehicle carrying a 2-axled trailer. This difference is shown by the figure below, where it becomes clear that the vehicle type of which the comparison is made, is likely not the same for the WIM data and the LP data.

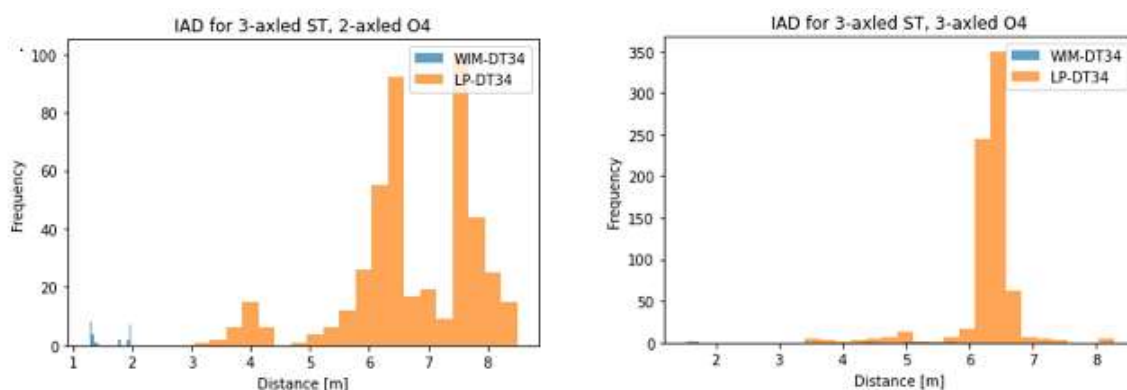


Figure 71 - Frequency distribution of interspatial axle distance *DT34*. Left: 3-axled semi-truck carrying 2-axled semi-trailer. Right: 3-axled semi-truck carrying 3-axled semi-trailer.

### Tipper trucks

In this paragraph the vehicle category tipper trucks will be evaluated. To distinguish tipper trucks from the WIM data, similar assumptions as stated in previous paragraph are used.

- Only 4- and 5-axled tipper trucks are considered
- All interspatial axle distances  $w_i < 200$  cm
- For LP data, the vehicle type must read 'tipper truck' (*kipper* in the code).

The WIM data recorded 88 vehicles matching these criteria. 68 were 5-axled tipper trucks, and 28 were 4-axled tipper trucks. The LP data recorded 942 4-axled tipper trucks and 1209 5-axled tipper trucks.

### 4-axled tipper trucks

For the LP data, none of the 4-axled tipper trucks apply to these criteria. This is a direct influence of the assumptions made in paragraph 5.1.3, where it is assumed that 4-axled tipper trucks have two groups of closely spaced axles as opposed to four more-or-less equally spaced axles. In total 1209 5-axled tipper trucks, of which 1176 match the criteria stated above. Both tipper trucks categories greatly outclass the registered WIM data. The figure below shows the results for 4-axled tipper trucks.

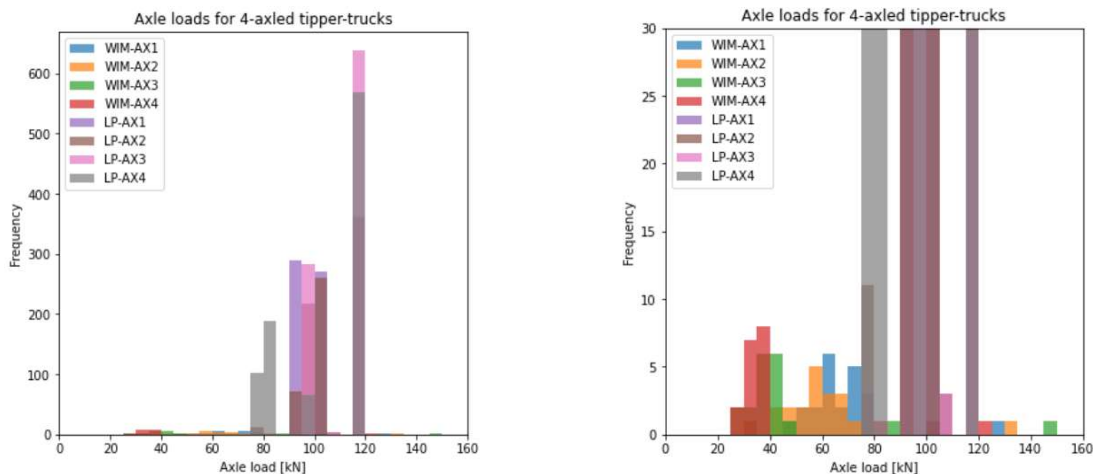


Figure 72 - Frequency distribution for axle loads for 4-axled tipper trucks for both LP data and WIM data. Right figure is zoomed in to properly display WIM data.

The left figure shows the true distribution of all four axle loads for both the WIM- and LP data. The right figure shows a less skewed distribution. Once again, the majority of the recorded WIM data seem to be underloaded than the allowed maximum allowed axle loads, whereas a small portion is exceeding the maximum allowed axle loads. The table below show numerical results regarding under- and overloaded axles. Note that in this case only 4-axled tipper trucks carrying no trailers are considered.

N3 vehicle	LP	WIM		Threshold [kN]	Underloaded	Overloaded
4-axled	924	20	Axle 1	102,71	95%	5%
			Axle 2	103,7	95%	5%
			Axle 3	108,86	95%	5%
			Axle 4	102,04	95%	5%

Table 23 - Overview of axle loads for 4-axled tipper trucks for both LP data and WIM data.

As shown above, a relatively small portion of the recorded axles are overloaded. All four axles are equally overloaded. The approach used in the LP data is a viable approach.

The table below shows the interspatial axle distances for 4-axled tipper trucks. As can be seen, the assumed axle configuration in the LP data is wrong, as the mean of DT23 differ by a factor of 2,22. When looking at the WIM data, all axles of the 4-axled tipper truck are approximately evenly distributed. When adjusting the assumptions for the LP data, this will be taken into account.

N3	LP			WIM			LP/WIM			
	IAD	AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
4-axled	DT12	1,12	0,9	1,35	1,91	1,63	1,94	59%	55%	70%
	DT23	3,97	2,77	5,7	1,79	1,77	1,81	222%	156%	315%
	DT34	1,13	0,9	1,35	1,82	1,8	1,85	62%	50%	73%

Table 24 - Overview of interspatial axle distances for 4-axled tipper trucks for both LP data and WIM data.

5-axled tipper trucks

In the figure below the results are shown for 5-axled tipper trucks.

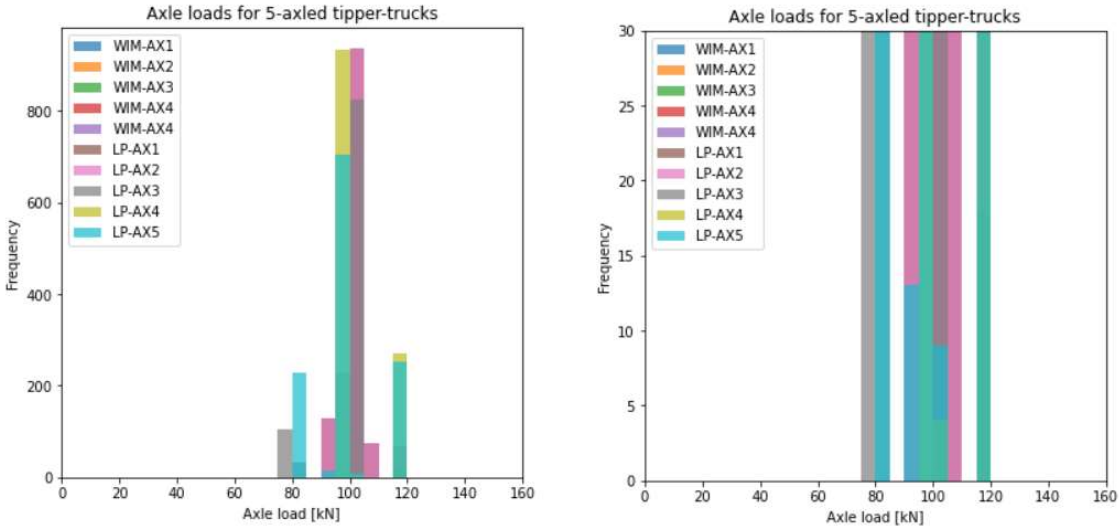


Figure 73 - Frequency distribution for axle loads for 5-axled tipper trucks for both LP data and WIM data. Right figure is zoomed in to properly display WIM data.

The left figure shows the distribution for all fixe axle loads for the WIM data. Once more, most of the recorded WIM data seems to be underloaded. A small portion is overloaded, primarily axles 4 and 5. The table below show the numerical results.

N3 vehicle	LP	WIM	Threshold [kN]	Underloaded	Overloaded	
5-axled	1209	68	Axle 1	100,08	54,41%	45,59%
			Axle 2	100,08	61,76%	38,24%
			Axle 3	96,57	45,59%	54,41%
			Axle 4	99,49	30,89%	69,11%
			Axle 5	96,31	30,89%	69,11%

Table 25 - Overview of axle loads for 5-axled tipper trucks for both LP data and WIM data.

Immediately it shows that the 5-axled tipper trucks are quite likely to be overloaded. Especially the latter two axles seem to be overloaded many times. However, when looking to the histograms, it seems that the amount of overloading is not that high. The numerical results shown in the table might therefore give a skewed representation of the data.

The table below shows the interspatial axle distance distribution for 5-axled tipper trucks. As can be seen, the assumed axle configuration for the LP data is approaching the axle configuration in the WIM data very well. Thus, for 5-axled tipper trucks the assumed axle configuration is correct.

N3	LP				WIM			LP/WIM		
	IAD	AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
5-axled	DT12	1,76	1,68	2,2	1,79	1,4	1,99	98%	120%	111%
	DT23	1,76	1,68	2,2	1,81	1,76	1,98	97%	95%	111%
	DT34	1,76	1,68	2,2	1,55	1,29	1,86	114%	130%	118%
	DT45	1,76	1,68	2,2	1,59	1,29	1,88	111%	130%	117%

Table 26 - Overview of interspatial axle distances for 4-axled tipper trucks for both LP data and WIM data.

### Mobile cranes

In the LP data, the vehicle category mobile cranes is explicitly stated in the LP data, whereas it is not directly recognisable in the WIM data. Moreover, due to the axle positions, number of axles and legally allowed axle loads, the mobile crane category shows high similarities with tipper trucks. To account for this, now a comparison is made between the length of the mobile cranes and the tipper trucks. This is shown in the figure below.

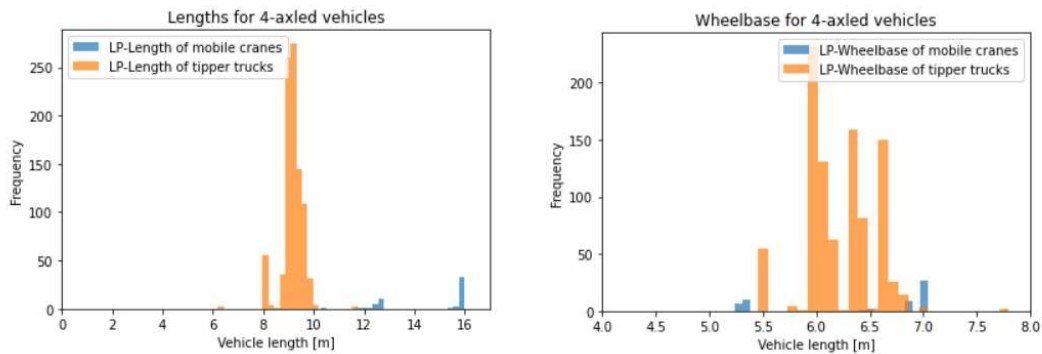


Figure 74 - Frequency distributions for 4-axled mobile cranes and tipper trucks for both LP data and WIM data. Left: total vehicle length. Right: total wheelbase.

This graph shows a clear distribution of the separate vehicle lengths and wheelbases for both vehicle classes. From these graphs, the following conclusion can be drawn:

- The total length of a tipper truck is limited to  $l_{tip} \leq 12$  m
- The total length of a mobile crane is for most part  $l_{crane} \geq 12$  m
- The total wheelbase of a tipper truck ranges from  $5.5 \text{ m} \leq w_{tip,tot} < 7$  m, except for an incidental outlier.
- The total wheelbase of a mobile cranes is either  $w_{crane,tot} \leq 5.5$  m or  $w_{crane,tot} \sim 7$  m.

Based on these interim conclusions, the following assumptions can be made to compare WIM data and LP data for mobile cranes.

- Only 4- and 5-axled mobile cranes are considered
- All interspatial axle distances  $w_i < 3$  m. Slightly increased to widen the bandwidth.
- The total length  $l_{tot} > 12$  m
- The sum of the intermediate axle distances must be  $w_{tot} > 6.5$  m

Using the criteria above, the following results are obtained for 4-axled and 5-axled mobile cranes.

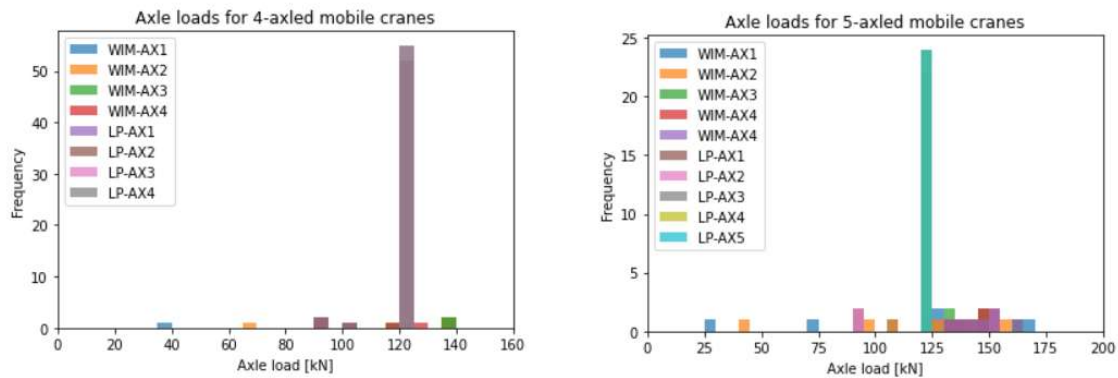


Figure 75 - Frequency distribution for axle loads of 4- and 5-axled mobile cranes for both LP data and WIM data.

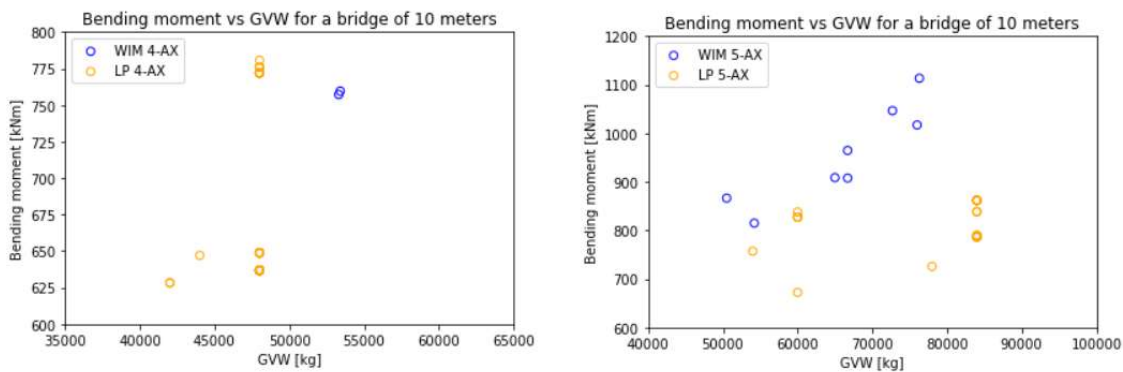


Figure 76 - Bending moment vs GVW for 4- and 5-axled mobile cranes acting on a span length of 10 m.

N3 vehicle	LP	WIM		Threshold [kN]	Underloaded	Overloaded
4-axled	3	55	Axle 1	118,55	33,33%	66,67%
			Axle 2	118,55	33,33%	66,67%
			Axle 3	120	33,33%	66,67%
			Axle 4	120	33,33%	66,67%
5-axled	8	24	Axle 1	117,5	37,5%	62,5%
			Axle 2	117,5	37,5%	62,5%
			Axle 3	120	0%	100%
			Axle 4	120	0%	100%
			Axle 5	120	0%	100%

Table 27 - Overview of axle loads for 4- and 5-axled mobile cranes for both LP data and WIM data.

As can be seen from the graphs and the table, a lot of axles that are recorded are overloaded. When considering  $M_{Max}$  it seems that for 4-axled mobile cranes the highest occurring bending moment seems to be reasonably approximated by the LP data. For 5-axled mobile cranes however, it seems that the LP data is underestimating the true occurring  $M_{Max}$ . A likely explanation for this is that 5-axled mobile cranes seem to have a higher GVW and likely thus a higher axle load.

The table below shows the interspatial axle distances for 4- and 5-axled mobile cranes. For both vehicle types, all axles are assumed to be evenly distributed along the total wheelbase. For 4-axled mobile cranes, both distances  $DT_{12}$  and  $DT_{23}$  are well assumed. Distance  $DT_{34}$  however is underestimated. This shows the initial assumption made in paragraph 5.1.3 is false and needs to be adjusted.

For 5 -axled mobile cranes, distance *DT12* is underestimated and distances *DT23* and *DT45* are overestimated. For these all these distances the deviation is small. When adjusting the model for the LP data, this will be taken into account by shifting the axle configuration a bit.

N3	LP				WIM			LP/WIM		
	IAD	AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
4-axled	DT12	2,16	1,75	2,35	2,12	2,11	2,13	102%	83%	110%
	DT23	2,16	1,75	2,35	2,02	2,01	2,04	107%	87%	115%
	DT34	2,16	1,75	2,35	2,72	2,69	2,75	79%	65%	85%
5 -axled	DT12	1,88	1,77	2,29	2,37	2,05	2,59	79%	86%	88%
	DT23	1,88	1,77	2,29	1,68	1,62	1,87	112%	109%	122%
	DT34	1,88	1,77	2,29	1,79	1,57	1,98	105%	113%	116%
	DT45	1,88	1,77	2,29	1,68	1,31	1,93	112%	135%	119%

Table 28 - Overview of interspatial axle distances for 4- and 5-axled mobile cranes.

### Full Trucks

The final vehicle group that has to be evaluated are the full trucks, possibly carrying trailers. In the WIM data, they are represented by vehicle the classes as shown in appendix B. In the LP data, the full trucks carrying trailers are depicted with a vehicle type other than *tipper truck*, *mobile crane* or *semi-truck*. These vehicle types are (in Dutch) *gesloten opbouw*, *geconditioneerd met temperatuurregeling*, *geconditioneerd voertuig*, *afneembare bovenbouw*, *open wagen*, *huifopbouw*, *voor vervoer voertuigen*.

### 2-axled full trucks

First, only the 2-axled full trucks are considered. A distinction is made whether the trucks carry a trailer or not. In the WIM data, the vehicle class 11 represents this data. In the LP data, the vehicle type must be one of the vehicle types shown above. The graphs below show the recorded axle loads for 2-axled full trucks, carrying no trailers.

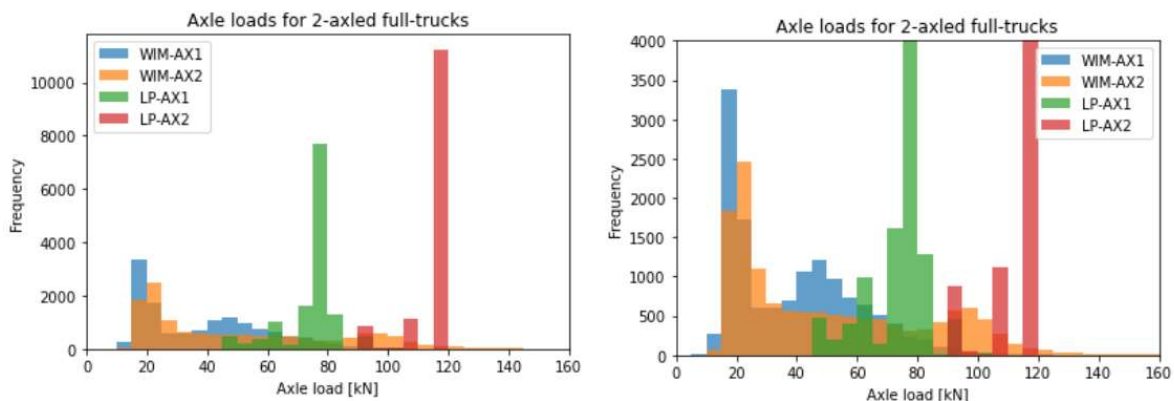


Figure 77 - Frequency distribution for axle loads for 2-axled full trucks for both LP data and WIM data. Right figure is zoomed in to properly distribution WIM data.

The graphs show the ongoing trend with other vehicle classes as well, namely the fact that the majority of the recorded axle loads are underloaded and only a small portion is overloaded. The table below shows the percentage of overloaded and underloaded axles. This threshold value is calculated as the mean value of recorded axle loads, based on the LP data, for both axles individually.



N3 vehicle	LP	WIM	O4 axles	Threshold [kN]	Underloaded (%)	Overloaded (%)
2-axled full-truck	13302	13475	0	Axle 1	72,52	93,74
				Axle 2	112,7	97,31

Table 29 - Overview of axle loads for 2-axled full trucks for both LP data and WIM data.

Now the 2-axled full trucks carrying trailers will be considered. As shown in appendix B, 2-axled full trucks carrying trailers are classified as 111,211,1111,2111<sup>30</sup> which refers to 1-, 2- and 3-axled trailers respectively. The graphs and tables below show the results.

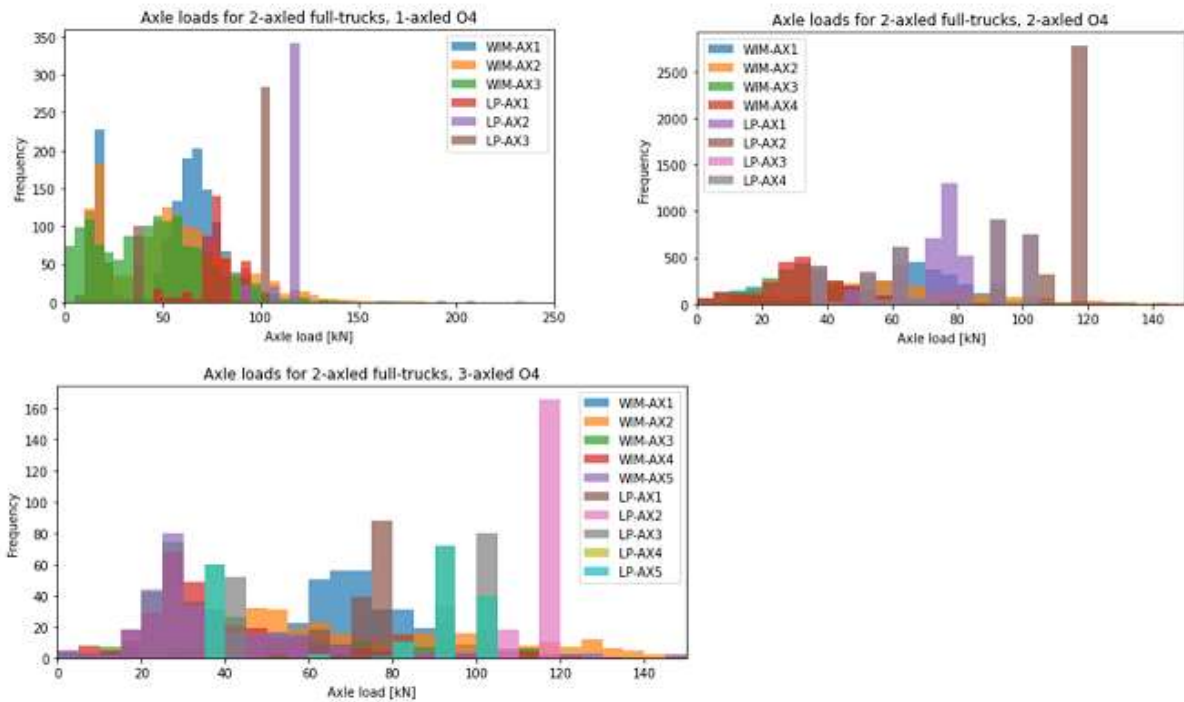


Figure 78 - Frequency distributions of axle loads for 2-axled full trucks carrying 1-, 2-, or 3-axled trailers.

<sup>30</sup> Vehicle categories 111, 211 and 1111 are analysed before in paragraph 5.3.1.1, corresponding to 2- and 3-axled semi-trucks. At this point it is assumed that within the WIM data, also 2-axled full trucks carrying 1- and 2-axled trailers are registered.



N3 vehicle	O4 axles	LP	WIM		Threshold [kN]	Underloaded	Overloaded
2-axled	0	13302	13475	Axle 1	72,52	93,74%	6,26%
				Axle 2	112,7	97,31%	2,69%
2-axled	1	387	1137	Axle 1	74,43	85,4%	14,6%
				Axle 2	113,18	98,32%	1,68%
				Axle 3	84,23	88,13%	11,87%
2-axled	2	3264	2956	Axle 1	73,15	69,48%	30,52%
				Axle 2	112,94	94,21%	5,79%
				Axle 3	74,55	95,08%	4,92%
				Axle 4	74,55	95,6%	4,4%
2-axled	3	337	186	Axle 1	74,63	64,98%	35,02%
				Axle 2	113,8	84,57%	15,43%
				Axle 3	77,58	89,31%	10,69%
				Axle 4	74,23	88,72%	11,28%
				Axle 5	73,68	88,72%	11,28%

Table 30 - Overview of axle loads for 2-axled full trucks carrying 1-, 2,- or 3-axled trailers.

As can be seen, most of the recorded vehicles in the WIM data fall in the category of 2-axled carrying no trailers. For this category, the number of overloaded axles is little. Trucks carrying trailers however show some significant overloading, especially those carrying 2-axled trailers. It is remarkable that for both the 2- and 3-axled truck-trailer combination, the number of overloaded axles is quite high, i.e. over 30 percent.

The table below shows the distribution of the interspatial axle distances for both the LP- and the WIM data. A few odd entries are found, especially for the LP data. These are summed and explained below:

N3	O4 AX	LP				WIM			LP/WIM		
		IAD	AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
2-axled	0	DT12	5,41	3,55	7	4,55	1,73	13,15	119%	205%	53%
2-axled	1	DT12	5,24	3,3	7	3,58	1,3	12,77	146%	254%	55%
		DT23	1,13	0,9	1,35	5,7	1,02	15,69	20%	88%	9%
2-axled	2	DT12	5,28	3	7	3,81	2	7,86	139%	150%	89%
		DT23	2,5	0,11	7,46	5,7	1,52	12,15	44%	7%	61%
		DT34	4,95	0	9,79	1,75	0,75	7,41	283%	0%	132%
2-axled	3	DT12	5,15	3,55	6,7	3,72	2,02	5,19	138%	176%	129%
		DT23	1,29	0,9	5,19	5,04	2,01	10,88	26%	45%	48%
		DT34	4,59	2,52	7,96	1,82	0,75	7,16	252%	336%	111%
		DT45	1,17	0,9	2,64	1,73	0,75	1,98	68%	120%	133%

Table 31 - Overview of interspatial axle distances for 2-axled full trucks carrying 1-, 2-, or 3-axled trailers.

- 3-axled LP combinations: the mean value for *DT23* seem to unreasonably small. When looking at the script, it seems that a small error was made. This will be patched.
- 4-axled LP combinations: the mean value for *DT34* seem to be unreasonably high. However, this is likely caused by the same error made when comparing semi-trucks; a comparison is made between different vehicle classes. However, the minimum value of 0 is obviously unrealistic and needs to be reviewed upon.

- 5-axled LP combinations: the mean values for DT23 and DT34 differ quite a lot from the WIM data and need to be reviewed upon.

Apparently, a small error was made when writing the script to construct the load models. This will be reviewed and adjusted accordingly.

### 3-axled full trucks

This paragraph contains 3-axled full trucks carrying both trailers of any kind and no trailers at all. First, only 3-axled full trucks will be considered. They are represented in the WIM data by vehicle class 12 and 21.

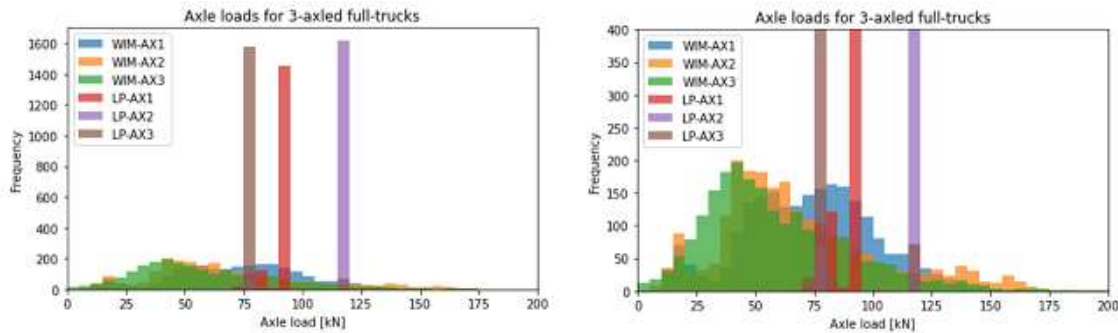


Figure 79 - Frequency distributions of axle loads for 3-axled full trucks for both LP data and WIM data. Right figure is zoomed in to properly display WIM data.

N3 vehicle	O4 axles	LP	WIM	Threshold [kN]	Underloaded	Overloaded	
3-axled	0	1692	2208	Axle 1	88,29	71,88%	28,12%
				Axle 2	113,3	87,64%	12,36%
				Axle 3	76,77	74,78%	25,22%

Table 32 - Overview of axle loads for 3-axled full trucks for both LP data and WIM data.

The 3-axled full trucks carrying no trailers show a significant number of overloaded axles, and in particular the first and last axle. Although the vast majority is underloaded, the portion that is overload, is usually exceedingly overloaded as can be seen in the histograms.

Secondly, the 3-axled full trucks carrying trailers of any kind will be shown. These vehicles are represented in the WIM data by vehicle classes 221, 1121, 2121<sup>31</sup>. In the WIM data, no single axled trailers are recorded. In the LP data however, 594 entries were recorded. Since there is nothing to comparable material, the single axled trailers in the LP data are not considered. The graphs and table below show the result.

<sup>31</sup> Vehicle categories 221, 1121 and 2121 are analysed in paragraph 5.3.1.1, corresponding to 2- and 3-axled semi-trucks. At this point, it is assumed that within these categories both full-trucks and semi-trucks are recorded.

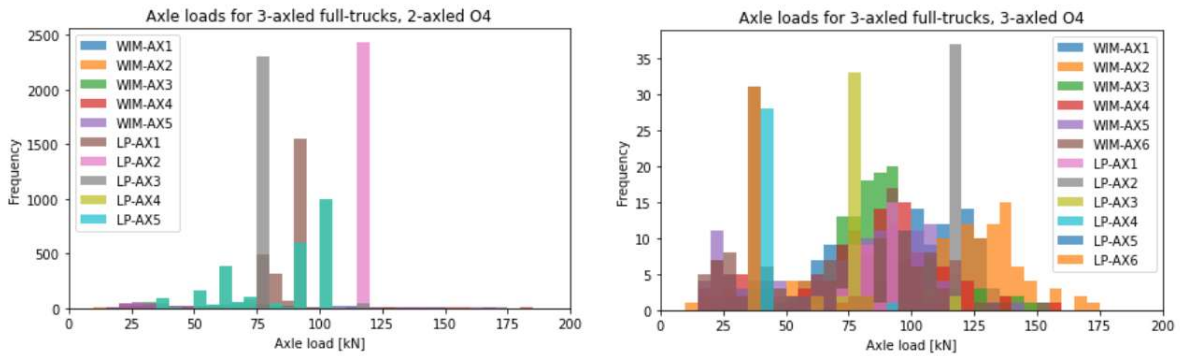


Figure 80 - Frequency distributions of axle loads for 3-axled full trucks carrying 2- or 3-axled trailers for both LP data and WIM data.

N3 vehicle	O4 axles	LP	WIM	Threshold [kN]			
				Underloaded	Overloaded		
3-axled	2	367	2483	Axle 1	85,2	64,57%	35,43%
				Axle 2	11,54	87,47%	12,53%
				Axle 3	75,52	71,39%	28,61%
				Axle 4	83,04	73,57%	26,43%
				Axle 5	83,04	75,48%	24,52%
3-axled	3	38	136	Axle 1	81,74	31,62%	68,38%
				Axle 2	113,84	44,12%	55,88%
				Axle 3	76,66	28,68%	71,32%
				Axle 4	47,45	19,11%	80,89%
				Axle 5	43,61	22,05%	77,95%
				Axle 6	43,61	22,05%	77,95%

Table 33 - Overview of axle loads for 3-axled full trucks carrying 2- or 3-axled trailers for both LP data and WIM data.

It becomes immediately visible that within these vehicle classes a lot of overloaded axles are noted. Especially for 6-axled combinations, where for the majority of registered vehicles overloaded axles are seen. Once again, for the 5-axled combinations the first axle seems to be overloaded most of the times.

The table below show the interspatial axle distances for 3-axled full trucks carrying a trailer of any kind. Just like their 2-axled predecessors a lot of odd results are shown. These are summed and explained accordingly.

N3	O4 AX	LP				WIM			LP/WIM		
		IAD	AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
3- axled	0	DT12	5,53	2,96	6,6	4,15	1,32	10,49	133%	224%	63%
		DT23	1,12	0,9	1,35	1,39	0,75	4,01	81%	120%	34%
3- axled	2	DT12	5,22	2,97	6,58	3,37	1,4	5,82	155%	212%	113%
		DT23	1,12	0,9	1,35	1,44	0,82	1,99	78%	110%	68%
		DT34	2,66	0,06	6,02	4,66	1,29	12,55	57%	5%	48%
		DT45	4,49	0	9,3	2,08	0,99	7,1	216%	0%	131%
3- axled	3	DT12	5,07	2,95	6,43	3,83	2,52	5,57	132%	117%	115%
		DT23	1,14	0,9	1,35	1,35	1,25	1,46	84%	72%	92%
		DT34	1,45	0,9	5,17	4,75	2,92	6,51	31%	31%	79%
		DT45	5,04	1,26	8,46	3,32	1,27	4,85	152%	99%	174%
		DT56	1,14	0,92	1,33	1,44	1,27	1,98	79%	72%	67%

Table 34 - Overview of interspatial axle distances for 3-axled full trucks carrying no, 2- or 3-axled trailers for both LP data and WIM data.

- For all LP vehicle combinations: *DT23* seem to be slightly underestimated. This will be adjusted in the new model by slightly increasing the value.
- For 5-axled LP vehicle combinations: the minimum value for *DT23*, *DT34* and *DT45* are unrealistic and need to be reviewed upon.
- For 6-axled LP vehicle combinations; the minimum value for *DT23*, *DT34* and *DT56* are somewhat low and need to be reviewed upon.

#### 4-axled full trucks

In this paragraph, 4-axled full trucks will be evaluated. 4-axled full trucks carrying no trailers are classified in vehicle class 31, 22, 211 and 112, as shown in appendix B. However, classes 112 and 211 are already analyzed, which leave classes 31 and 22 yet to be analyzed. The graphs and table below show the obtained result.

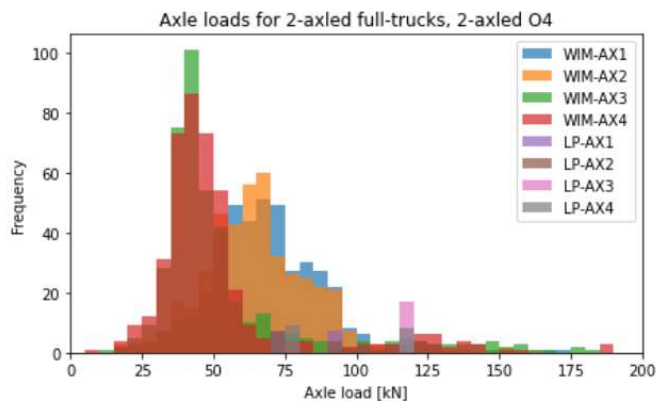


Figure 81 - Frequency distribution of axle loads for 4-axled full trucks for both LP data and WIM data.

N3 vehicle	O4 axles	LP	WIM	Threshold [kN]	Underloaded	Overloaded	
4-axled	0	17	427	Axle 1	86,59	83,61%	16,39%
				Axle 2	88,35	89,46%	10,54%
				Axle 3	115	91,57%	8,43%
				Axle 4	93,83	90,63%	9,37%

Table 35 - Overview of axle loads for 4-axled full trucks for both LP data and WIM data.

4-axled full-trucks carrying no trailers are not well represented by the LP data, as only 17 vehicles were recorded. Only the first two axles show relatively overloading as opposed to the latter axles. However, the graph shows that for primarily the latter axles, the ratio of overloading is quite high.

Secondly, all 4-axled full trucks carrying 2- and 3-axled trailers will be analyzed. These are classified in classes 222, 1122, 2122 as shown in appendix B. The graphs and table below show the result.

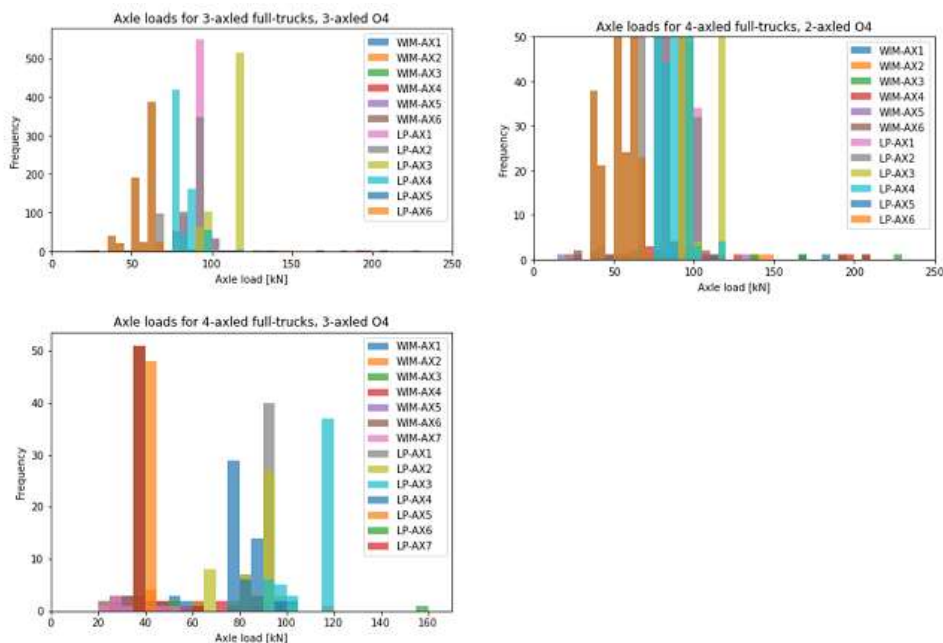


Figure 82 - Frequency distributions for axle loads of 4-axled full trucks. Top left: No trailer. Top right: 2-axled trailer. Bottom left: 3-axled trailer.

N3 vehicle	O4 axles	LP	WIM		Threshold [kN]	Underloaded	Overloaded
4-axled	0	427	17	Axle 1	86,59	83,61%	16,39%
				Axle 2	88,35	89,46%	10,54%
				Axle 3	115	91,57%	8,43%
				Axle 4	93,83	90,63%	9,37%
4-axled	2	13	686	Axle 1	89,02	30,77%	69,23%
				Axle 2	85,32	23,08%	76,92%
				Axle 3	109,77	61,54%	38,46%
				Axle 4	79,93	53,85%	46,15%
				Axle 5	56,64	53,85%	46,15%
				Axle 6	56,64	53,85%	46,15%
4-axled	3	51	12	Axle 1	88,53	66,6%	33,4%
				Axle 2	85,18	66,6%	33,4%
				Axle 3	109,45	91,6%	8,4%
				Axle 4	79,47	91,6%	8,4%
				Axle 5	41,59	58,3%	41,7%
				Axle 6	37,35	66,6%	33,4%
				Axle 7	37,35	66,6%	33,4%

Table 36 - Overview of axle loads for 4-axled full trucks carrying no, 2- or 3-axled trailers for both LP data and WIM data.

In this category, it seems that only 6- and 7-axled combinations have an exceedingly number of overloaded axles. Especially 6-axled combinations, where a significant number of axles are also exceedingly overloaded (> 150 kN). Especially the first two axles are overloaded in most cases.

The table below show the differences between interspatial axle distances for LP- and WIM data. Irregularities or odd results are discussed below.

N3	O4 AX	IAD	LP			WIM			LP/WIM		
			AVG [m]	Min. [m]	Max. [m]	AVG [m]	Min. [m]	Max. [m]	AVG [%]	Min. [%]	Max. [%]
4-axled	0	DT12	1,11	0,93	1,34	2,02	1,44	7,54	55%	65%	18%
		DT23	4,87	3,69	5,99	2,97	0,93	4,45	164%	397%	135%
		DT34	1,11	0,93	1,34	1,51	0,82	1,85	74%	113%	72%
4-axled	2	DT12	2,29	1,9	2,5	1,75	1,64	1,94	131%	116%	129%
		DT23	2,28	1,9	2,6	3,11	2,13	4,46	73%	89%	58%
		DT34	2,29	1,9	2,6	1,44	1,33	1,66	159%	143%	157%
		DT45	0,33	0	1,35	5,75	2,2	7,88	6%	0%	17%
		DT56	2,79	1,02	9,1	2,22	1,57	5,02	126%	65%	181%
4-axled	3	DT12	2,32	2,02	2,5	1,87	1,79	1,99	124%	113%	126%
		DT23	2,32	2,02	2,5	2,71	2,16	4,23	86%	94%	59%
		DT34	2,32	2,02	2,5	1,37	1,32	1,44	169%	153%	174%
		DT45	1,1	0,9	1,34	4,82	3,73	7,47	23%	24%	18%
		DT56	0	0	0	3,44	1,77	4,54	0%	0%	0%
		DT67	1,15	0,9	1,35	1,52	0,32	1,86	76%	281%	73%

Table 37 - Overview of interspatial axle distances for 4-axled full trucks carrying no, 2- or 3-axled trailers for both LP data and WIM data.

- For all 4-axled N3-vehicles: it seems that the assumed axle configuration used in paragraph 5.1.3 is not true, since the WIM data shows that the axles are more likely to be evenly distributed along the vehicle length.
- For 6-axled LP combinations: all values for *DT45* seem to be false and need to be reviewed.
- For 7-axled LP combinations: all values for *DT45* and *DT56* need to be reviewed, as they are unrealistically low.

#### Extremely Heavy Combinations (EHC)

Due to the approach stated in paragraph 5.1.3, the remaining category EHC should contain all vehicles that exceed a certain threshold value for either the number of axles or the GVW. As of now, for the WIM data, a maximum of 7 axles have been evaluated. For the LP data, a maximum GVW of 84t has been evaluated, leaving a small number of vehicles yet to be analyzed. Hence, to make a comparison for this vehicle category, the following criteria will be adopted:

- For WIM data, vehicles with 7+ axles **or** with a GVW > 84t will be evaluated
- For LP data, vehicles classified as *Extreme* will be evaluated.

For the WIM data, a total of 21 vehicles were recorded falling in a variety of vehicle classes, ranging from 5 axles to 10 axles and from 82,4t to 98,2t. The LP data contains 14 recorded vehicles, with the properties stated in paragraph 5.1.3. Since there is no criterium based on number of axles or interspatial axle distances, the occurring bending moments  $M_{Max}$  are evaluated.

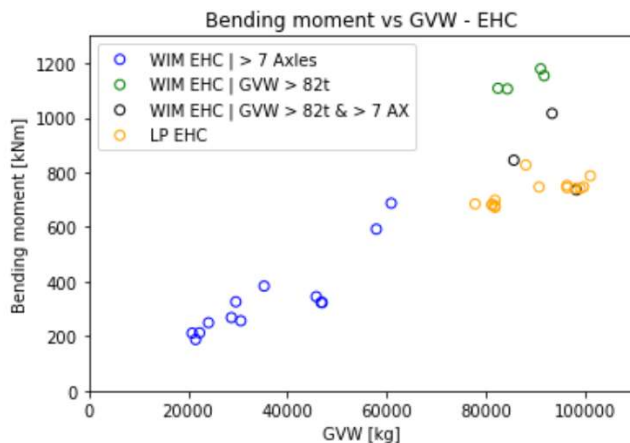


Figure 83 - Bending moment vs GVW for vehicle category EHC. WIM results grouped by denoted criteria. LP data not grouped.

EHC	#	$M_{Max}$ [kNm]			GVW [t]		
		Mean	Maximum	Minimum	Mean	Maximum	Minimum
WIM >7 axles	13	336	687	188	36	61	21
WIM >82t	4	1137	1179	1106	87	92	82
WIM (>82t & >7 ax)	3	866	1016	736	92	98	86
LP	14	727	827	672	90	101	78

Table 38 - Overview of statistical values for bending moments for vehicle category EHC.

The figure above shows a great spread in occurring bending moments for vehicles in the WIM data.

- Blue entries denote WIM data with more than 7 axles
- Green entries denote WIM data with GVW > 82t
- Black entries denote WIM data that meets both criteria
- Orange entries denote the LP data

The way this data is presented, shows that also for WIM data, the more axles does not necessarily result in higher load effect. This is shown by the blue and green entries in figure 80. Moreover, this strengthens the presumption that the occurring load effect is rather a result of closely spaced, overloaded axles rather than solely a function of maximum GVW.

When comparing the LP data to the WIM data, it seems that the LP data is somewhat of a category on its own and does not accurately represent the EHC category with the assumed criteria stated before. However, the adopted strategy for EHC in LP data seem to give a range of threshold values for vehicles with  $> 7$  axles. For WIM-vehicles with a GVW  $> 82t$ , the LP data is underestimating the occurring mean load effect by more than 50%. For adaptations regarding the approach, this must be taken into account, since simply adding more axles to the EHC apparently does not accurately describe EHC in the WIM data.

### Intermediate conclusions

#### Axle loads

- For 2-axled semi-trucks carrying semi-trailers: primarily the first two axles of semi-trucks are overloaded. The ratio of overloading is not quite high however, so no adjustments will be made.
- For 3-axled semi-trucks carrying semi-trailers: primarily the first two axles of semi-trucks are overloaded. Figure 68 shows that the ratio of overloading can be quite high, and thus an adjustment will be made to simulate overloaded axles.
- For 4-axled tipper trucks carrying no trailers: all four axles are primarily not overloaded. However, the ratio of underloaded axles is quite high. Accordingly, a factor will be introduced to simulate underloaded axles.
- For 5-axled tipper trucks carrying no trailers: a fair share of all axles seems to be overloaded. Accordingly, a factor will be introduced to simulate overloaded axles.
- For 4- and 5-axled mobile cranes: all axle loads are primarily underestimated by the approach used in 5.1.3. To simulate the overloaded axles as observed in the WIM data, a factor will be introduced to simulate overloaded axles.
- For 2-axled full trucks carrying no trailers: both axles are primarily not overloaded, and the ratio of overloading is not high as shown in figure 74. Both axles can be quite underloaded however, and thus a factor will be introduced to simulate under- and overloading.
- For 2-axled full trucks carrying 1-axled trailers: the first and third axle seem to be overloaded primarily. As shown in figure 75, the ratio of overloading can be as high as 2 times. Thus accordingly, a factor will be introduced to simulate under- and overloading.
- For 2-axled full trucks carrying 2-axled trailers: the first axle is overloaded primarily and accordingly a factor will be introduced to simulate under- and overloading.
- For 2-axled full trucks carrying 3-axled trailers: the first axle is primarily overloaded and accordingly a factor will be introduced to simulate under- and overloading.
- For 3-axled full trucks carrying no trailers: the first and third axles are primarily overloaded and accordingly a factor will be introduced to simulate under- and overloading.
- For 3-axled full trucks carrying 2-axled trailers: all axles seem to be overloaded and accordingly a factor will be introduced to simulate under- and overloading.
- For 3-axled full trucks carrying 3-axled trailers: all axles seem to be primarily overloaded and accordingly a factor will be introduced to simulate under- and overloading.
- For 4-axled full trucks carrying no trailers: primarily the first axle seems to be overloaded and accordingly a factor will be introduced to simulate under- and overloading.



- For 4-axled full trucks carrying 2-axled trailers: primarily all axles seem to be overloaded and accordingly a factor will be introduced to simulate under-and overloading.
- For 4-axled full trucks carrying 3-axled trailers: primarily the 5<sup>th</sup> axle seems to be overloaded and accordingly a factor will be introduced to simulate overloading.
- For EHC: As shown by figures 28 and 80 the combination of overloaded axles and closely spaced axles are determining for causing the maximum load effect. Since in this category no real relation is found between axle distances, overloaded axles must be the driving factor behind these load effects. Hence, for all axles a factor will be introduced to simulate overloading.

### Interspatial axle distances

- For 2-axled semi-trucks carrying semi-trailers: the assumptions made in paragraph 5.1.3 are validated and no adjustments have to be made.
- For 3-axled semi-trucks carrying semi-trailers: the assumptions made in paragraph 5.1.3 are roughly validated. However, for 5-axled combinations the distance *DT34* will be reviewed.
- For 4-axled tipper trucks carrying no trailers: the assumed axle configuration made in paragraph 5.1.3 is wrong and needs to be adjusted to approximate a more evenly spaced axle configuration.
- For 5-axled tipper trucks carrying no trailers: the assumed axle configuration is validated.
- For 4-axled mobile cranes carrying no trailers: the distance *DT34* is slightly overestimated and will be adjusted accordingly.
- For 5-axled mobile cranes carrying no trailers: the distance *DT12* is slightly underestimated and distances *DT23* and *DT45* are slightly overestimated. Both will be adjusted accordingly.
- For 2-axled full trucks carrying no trailers: the distance *DT12* is properly assumed. However, due to the nature the WIM data recorded vehicles ( *GVW > 35t*), a skewed distribution is shown.
- For 2-axled full trucks carrying 1-axled trailers: the assumed distance *DT23* is wrong and needs to be adjusted.
- For 2-axled full trucks carrying 2-axled trailers: distances *DT23* and *DT34* are unrealistic and need to be reviewed.
- For 2-axled full trucks carrying 3-axled trailers: distances *DT23* and *DT34* need to be reviewed.
- For all 3-axled full trucks: the distance *DT23* is relatively low and needs to be reviewed.
- For 3-axled full trucks carrying 2-axled trailers: the minimum values for *DT23*, *DT34* and *DT45* are unrealistically low and need to be reviewed.
- For 3-axled full trucks carrying 3-axled trailers: the minimum values for *DT23*, *DT34* and *DT56* are unrealistically low and need to be reviewed.
- For 4-axled full trucks: the assumed axle configuration in paragraph 5.1.3 is wrong and needs to be adjusted.
- For 4-axled full trucks carrying 2-axled trailers: all values for *DT45* seem unrealistic and need to be reviewed.
- For 4-axled full trucks carrying 3-axled trailers: all values for *DT45* and *DT56* seem unrealistic and need to be reviewed.
- For EHC: no comparison is made since the vehicle types in the WIM data are not of one type. Combined with the results for axle loads, the conclusion can be drawn that EHC in the WIM data are caused by (an) overloaded axle(s) and are not a result of very closely spaced axles. For the LP data however, the conclusion can be drawn that the assumed approach approximates the calculated load effect caused by overloaded vehicle properly.

### Contribution of groups of axles to maximum load effect

Paragraph 5.2.3.3 showed that the maximum load effect generated by a vehicle is usually due to 2,3 or 4 axles. The approach stated in this paragraph is also used for the WIM data to look at possible differences. The results are shown below.

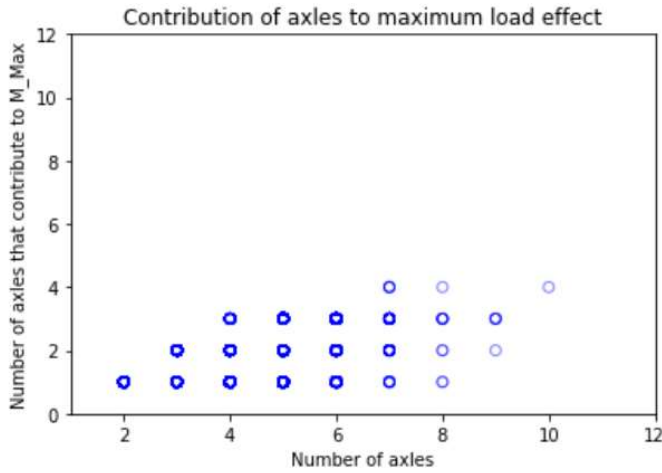


Figure 84 - Relation between number of axles contributing to the maximum load effect and the total number of axles

Contribution of axles to maximum load	5	0	0	0	0	0	0	4	1	0	0
	4	0	0	0	0	0	0	4	1	0	1
	3	0	0	0	30	657	303	49	5	4	0
	2	0	0	805	925	343	141	19	2	1	0
	1	0	13501	2541	2452	1693	116	7	2	0	0
	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	
<b>Number of axles</b>											

Table 39 - Numerical overview of number of axles contribution to maximum load effect.

Remarkably, for a bridge length of 10 m , only four axles simultaneously exert load effect on the span for a 10-axled vehicle combination. Even more striking is that in this case, the maximum load effect generated by a vehicle is in 57% of the cases due to a single axle, which stresses even more the influence of overloaded axles for relatively short span bridges. The expectation is that for shorter span bridges, the number of axles that contribute to the maximum load effect will go down. Likewise, for longer span bridges this number will increase. In appendix C this is described in detail.

### Location of maximum load effect

Also, for the WIM data the location of maximum occurring load effect has been determined. This is shown in the graphs and tables below.

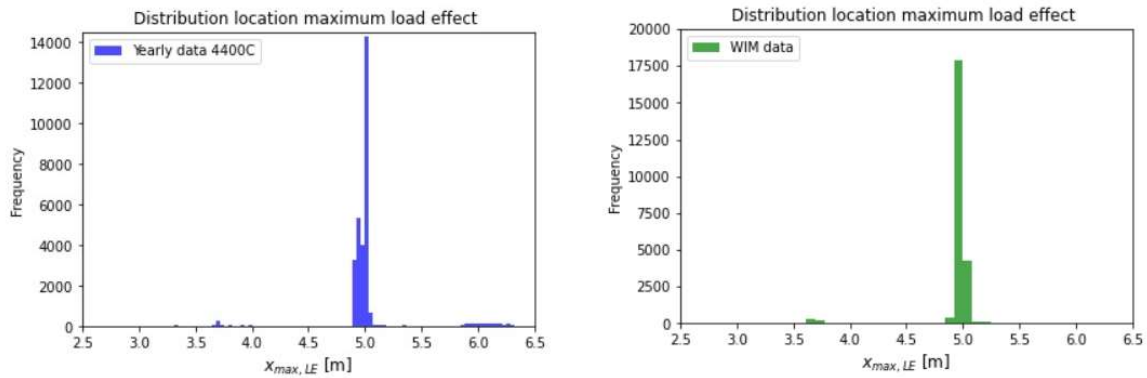


Figure 85 - Frequency distributions for location of maximum load effect. Left: LP data. Right: WIM data.

	LP	WIM
Unity	Value	Value
Mean	5,01	4,96
Median	5	5
Minimum	3,24	2,79
Maximum	6,91	8,2

Table 40 - Statistical values for locations of maximum load effects for both LP data and WIM data.

As can be seen, both the LP data and the WIM data give similar results with respect to the location of the maximum load effect as both are centred around midspan, i.e.  $x_{max,LE} = 5$  m.

### LM1

Also, figure 83 shows that in general a linear relation can be seen between the occurring bending moment and the gross vehicle weight. This is true especially for the majority of vehicles with a GVW smaller than 60t. For vehicles with a GVW greater than 60t, such a correlation is not clearly visible. The vehicles belonging to category EHC, do not exert a high load effect despite their high gross vehicle weight. This is since the entire combination exerts its weight only partially in time, which is due to the interspatial axle distances. When the bridge length exceeds a certain threshold, logically the category EHC will cause the highest load effect. This is shown in the figure below.

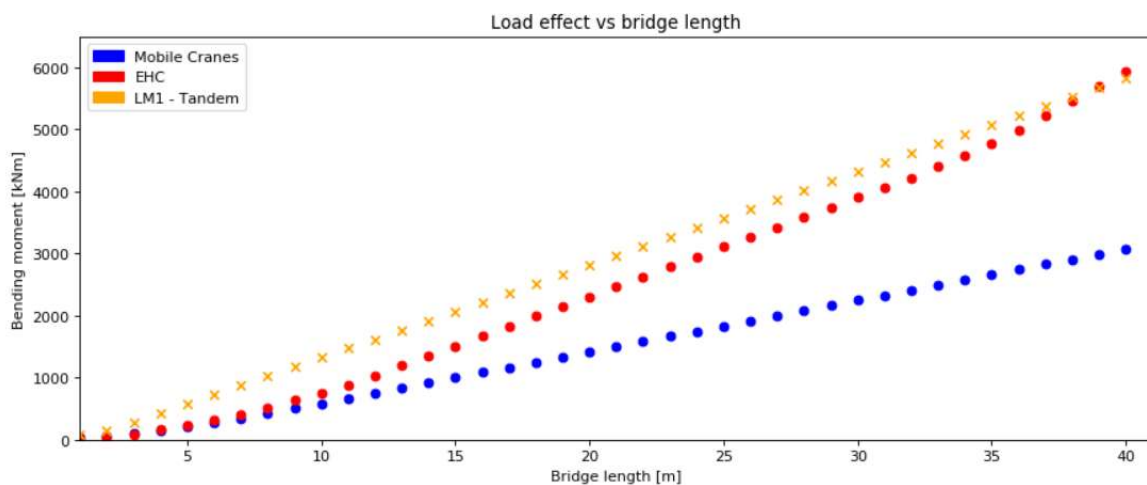


Figure 86 - Bending moment vs span length for 3 load models: mobile cranes, EHC, tandem load of LM1.

In the figure above, an arbitrary vehicle in both categories has been chosen to demonstrate. For mobile cranes, for bridges with a span larger than 16 m the increment in load effect suddenly increases. For EHC this is the case at 18 m bridge span. For single bridge spans greater than 40 m the tandem load model from LM1 represents the registered live load. This is further evidence that load model LM1 is overly conservative for small span bridges.

## Appendix D – Overloaded axles in WIM data

This appendix is a continuation appendix C where the influence of number of axles contributing to the maximum load effect is investigated. The figure below shows an example of a 3-axled vehicle. Columns *DT12* and *DT23* are the interspatial axle distances, respectively. Column *nF1* is the indicated ratio  $n_{F,1}$  and column *AXLOAD* describes the number of axles responsible for the highest occurring load effect. Immediately it shows that although three axles are present, only the first axle has the highest contribution, despite the relatively large axle distance to the last two axles.

CONFIG	GVW	AXW1	AXW2	AXW3	LENGTH	DT12	DT23	McMax [kNm]	nF1	AXLOAD	
5	21.0	201.0	113.0	53.0	35.0	1257.0	593.0	136.0	282.5	1.0	1.0

Figure 87 - Excerpt of WIM data showing contribution of axles to maximum load effect for 3-axled vehicle.

The figure below shows an example of an 8-axled vehicle with a GVW of 85,6t. A first assumption would be that due to the high GVW and the high number of axles, the contribution of a group of axles would be relatively the same. However, the 4<sup>th</sup> and 5<sup>th</sup> axle are both relatively high loaded, the 5<sup>th</sup> axle. The ratio  $\eta$  for axles 4,5,6 and 7 show that this 8-axled vehicle loads the structure as a 4-axled vehicle, with primarily only the 5<sup>th</sup> axle exerting load effect.

CONFIG	GVW	AXW1	AXW2	AXW3	AXW4	AXW5	AXW6	AXW7	AXW8	LENGTH	DT12	DT23	DT34	DT45	DT56	DT67	DT78	McMax [kNm]	
8219	222211.0	856.0	98.0	88.0	92.0	144.0	250.0	54.0	67.0	63.0	1962.0	420.0	132.0	180.0	366.0	129.0	294.0	129.0	844.715
	nF4	nF5	nF6	nF7	AXLOAD														
	0.112511	0.736935	0.119224	0.03133	1.0														

Figure 88 – Excerpt of WIM data showing contribution of axles to maximum load effect for 8-axled vehicle.

The figure below shows that the maximum occurring load effect in the WIM data is generated by either 2 or 3 axles. Assuming that the interspatial axle distances are reasonable, and not very small, this indicates that overloaded axles contribute most to maximum occurring load effects.

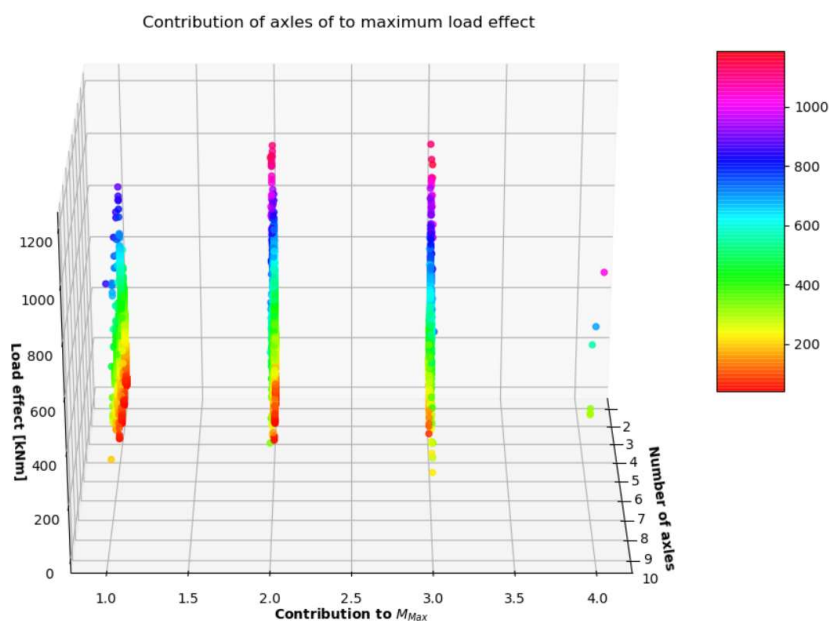


Figure 89 - 3D representation of contribution of axles to maximum load effect by adding colour scale.

## Appendix E – Modified LP data: Distribution fitting to maxima

### Daily maxima

In theory, any function can be fitted to any distribution. However, in which way the distribution fits best is debatable. The final distribution shown in figure 46 is determined by fitting 89 univariate distributions through the residual sum of squares method and hypothesis testing. For each distribution, the location, shape, and possibly other parameters are determined by maximum likelihood estimation. This is done by a Python package called *distfit*<sup>32</sup>, freely available online. All 89 distributions are explained and elaborated in the Scipy library<sup>33</sup>. It must be noted that some of these distributions are a modified version of the normal distribution, such as the hyper gaussian function.

Initially to fit a distribution to the daily maxima, the parameters for Weibull-, Gumbel-, Normal- and Beta distribution are calculated. Then, using the *distfit* package, the theoretically best fitting distribution is calculated. The results are shown below.

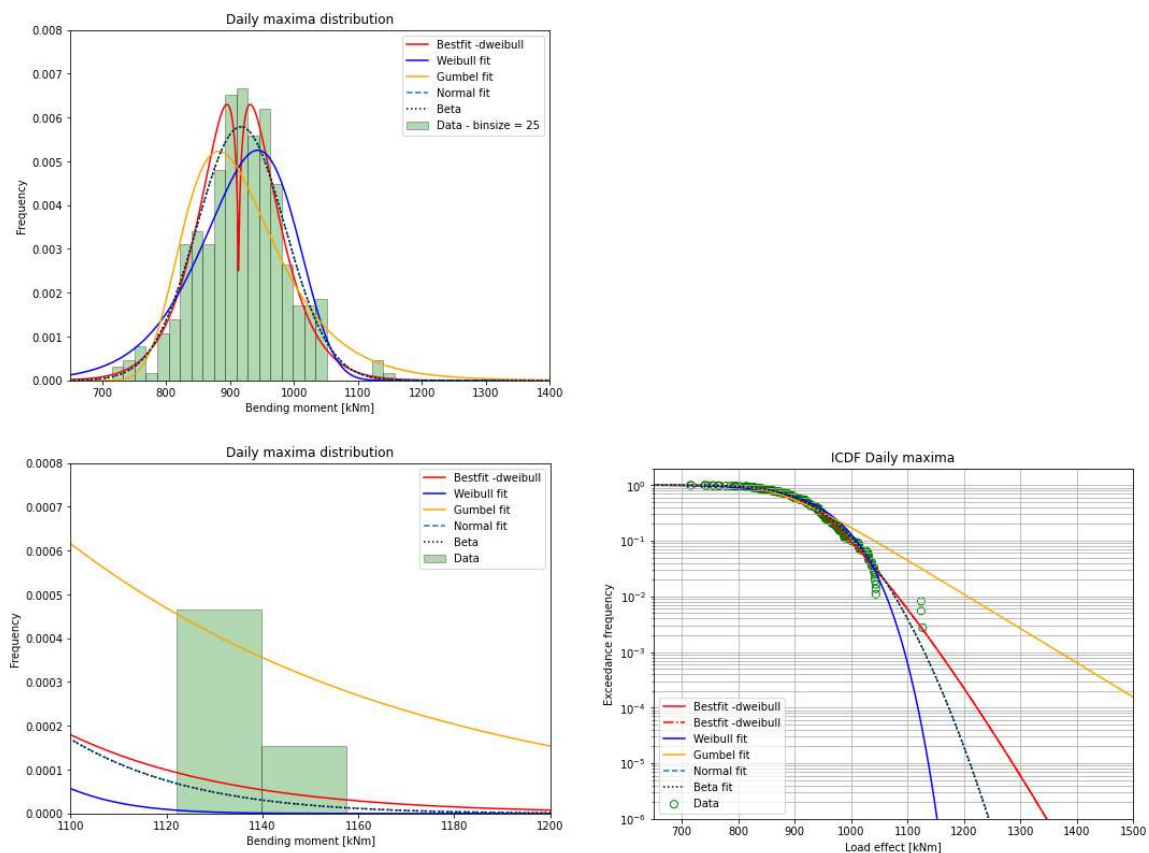


Figure 90 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for daily maxima. Bottom left: Enhanced vision of tail of relative frequency. Bottom right: icdf of daily maxima and fitted distributions.

The figure shows that most of the fitted distributions approximate the histogram well, only showing some minor differences between the peaks and tail. The best fitted distribution using the *distfit* package is the *dweibull* distribution, short for Double Weibull distribution. Since the tail is of most importance in extreme value analysis, this part must be fitted well. From figure 87 the following observations are done:

<sup>32</sup> <https://pypi.org/project/distfit/>

<sup>33</sup> <https://docs.scipy.org/doc/scipy/reference/stats.html>

- The Weibull distribution is underestimating the exceedance frequency.
- The Gumbel distribution is overestimating the exceedance frequency.
- The Normal distribution although accurately describes the highest value of the daily maxima, underestimates the exceedance frequency of the tail.
- The Beta distribution also underestimates the exceedance frequency, slightly more than the Normal distribution.
- The Double Weibull distribution seems to fit the data best and although slightly overestimating the highest daily maxima, the rest of the tail is properly described.

Based on these observations, the most logical choice is to use a Double Weibull distribution to fit the daily maxima, with the following parameters<sup>34</sup>:

Parameter	Abbreviation (code)	Value
Shape	<i>c</i>	1.2859689606414004
Location	<i>loc</i>	913.3432857362216
Scale	<i>scale</i>	58.52499587622529

Table 41 - Distribution parameters for fitted daily maxima distribution.

### Monthly maxima

The same procedure as for the daily maxima is now applied to determine the monthly maxima for a reference period of  $t$  years. In this example, a return period of 25 years is applied. Thus 25 years of monthly maxima is generated. With making use of the *distfit* package, several probability distributions are fitted. The following results are obtained.

The figure shows a mixture of fitted distributions to the monthly maxima of 25 years. The *distfit* package determined that the *mielke*<sup>35</sup>, short for Mielke Beta-Kappa / Dagum, fits the data best. The following observations can be done:

- The Weibull distribution is underestimating the exceedance frequency.
- The Gumbel distribution is slightly overestimating the exceedance frequency.
- The Normal distribution although accurately describes the middle values (1250 – 1350 kNm) of the monthly maxima, underestimates the exceedance frequency of the tail.
- The Beta distribution, although accurately describing the tail, underestimates the peak of the distribution at 1255 kNm.
- The Mielke distribution is slightly overestimating the exceedance frequency, but accurately describes the peak of the distribution.

<sup>34</sup> <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.dweibull.html#scipy.stats.dweibull>

<sup>35</sup> <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mielke.html>

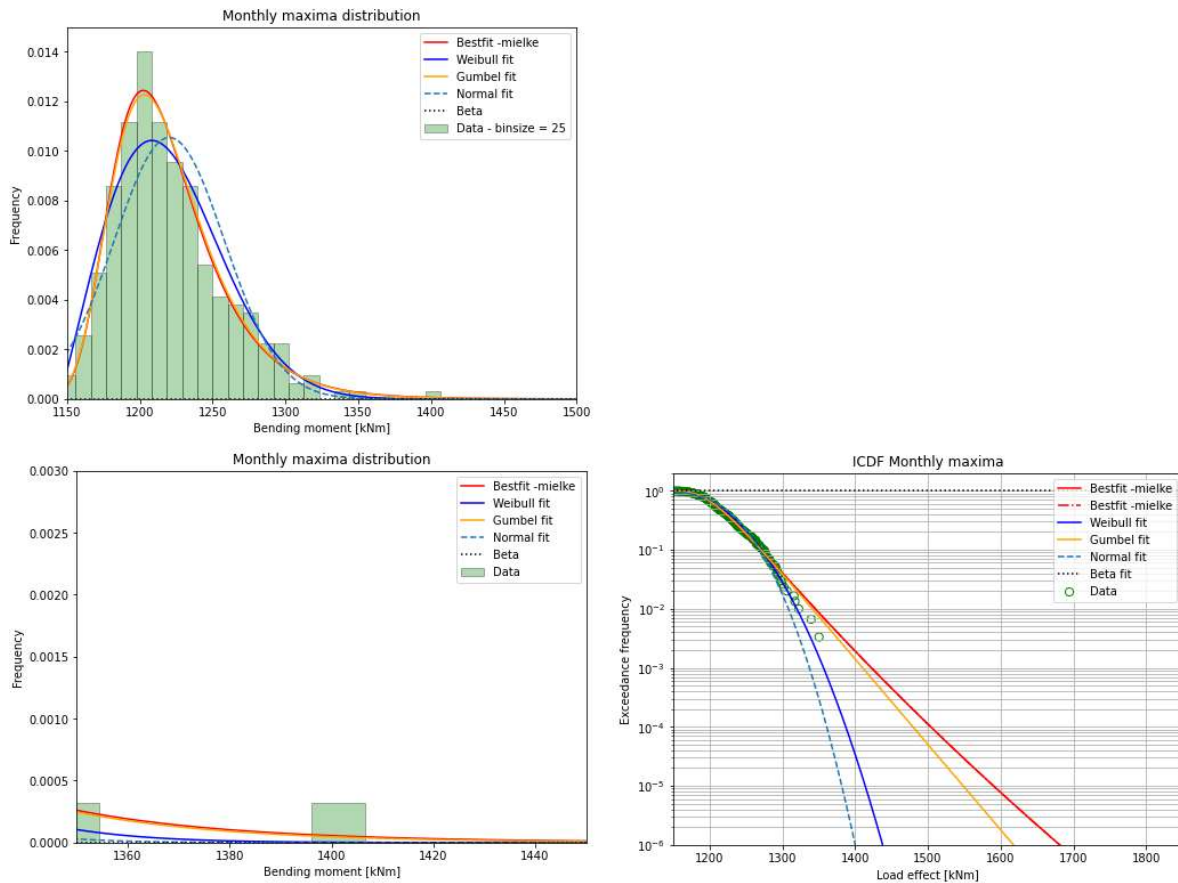


Figure 91 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for monthly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of monthly maxima and fitted distributions.

Usually, the best fitted distribution should be fitted to the data. Visually however the *mielke* and the Gumbel-distribution seem to be almost exact. Based on this, and the fact that Hellebrandt [1] also used a Gumbel distribution, the fitted distribution is a Gumbel-distribution with the following parameters<sup>36</sup>.

Parameter	Abbreviation (code / math)	Value
Location	$loc / \mu$	1203.0105580793575
Scale	$scale \beta$	30.00058334129722

Table 42 - Distribution parameters for fitted monthly maxima distribution.

### Yearly maxima

To determine the yearly maxima, the same procedure as for the monthly maxima is applied. The Gumbel-distribution for the monthly maxima with the parameters denoted in table 42 act as an input. 1 year is simulated by means of drawing 12 monthly maxima from the given Gumbel-distribution and storing the maximum of the monthly maxima. This is repeated for 1000 years. With making use of the *distfit* package, several distributions are fitted to the data. The results are shown below.

<sup>36</sup> [https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gumbel\\_r.html#scipy.stats.gumbel\\_r](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gumbel_r.html#scipy.stats.gumbel_r)



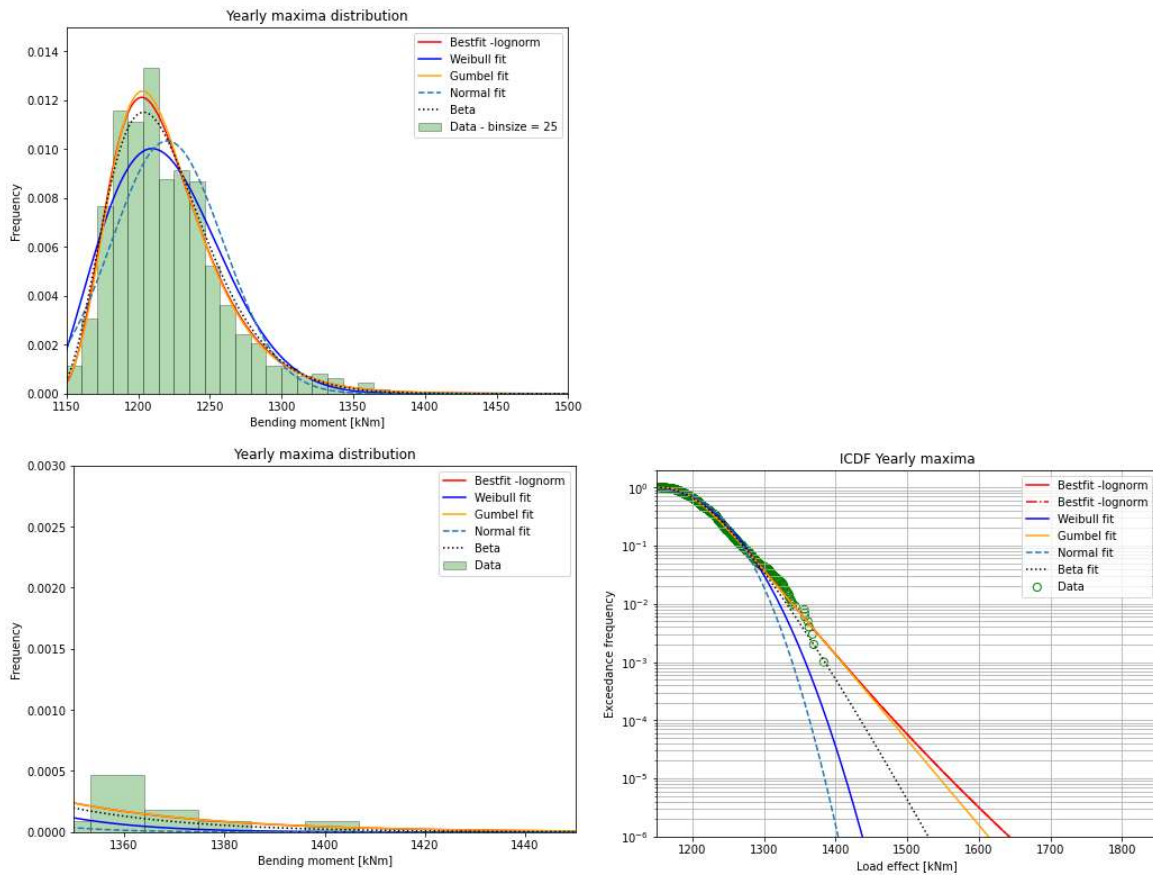


Figure 92 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for yearly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of yearly maxima and fitted distributions.

The figure shows a mixture of fitted distributions to the yearly maxima of 1000 simulated years. The *distfit* package determined that the *lognorm*, short for lognormal distribution, fits the data best. The following observations can be done:

- The Weibull distribution is underestimating the exceedance frequency.
- The Gumbel distribution is slightly overestimating the exceedance frequency for the tail of the distribution.
- The Normal distribution is underestimating the tail of the distribution.
- The Beta distribution underestimates the peak of the distribution and underestimates the tail of the distribution.
- The lognormal distribution is slightly overestimating the exceedance frequency for the tail of the distribution.

Once again, the *lognorm* distribution seems to fit the yearly maxima best. Once again, the Gumbel- and the lognormal distribution are visually almost identical. The figure below shows that the difference in the residual sum of squares for both distributions is negligible.

Based on the facts that the difference between both functions is negligible and that the monthly maxima distribution is also a Gumbel distribution, the fitted function for the yearly maxima is set to be a Gumbel distribution with the following parameters.

Parameter	Abbreviation (code / math)	Value
Location	$loc / \mu$	1227.9312247878345
Scale	$scale \beta$	28.938224976728158

Table 43 - Distribution parameters for fitted yearly maxima distribution.

## Overview

For the daily, monthly, and yearly maxima the best fitted distributions are chosen. The yearly maxima distribution will yield as the basic input for the Monte-Carlo simulation in Chapter 7.3.3. The figure below shows the daily, monthly and yearly maxima distributions together with their fitted distributions.

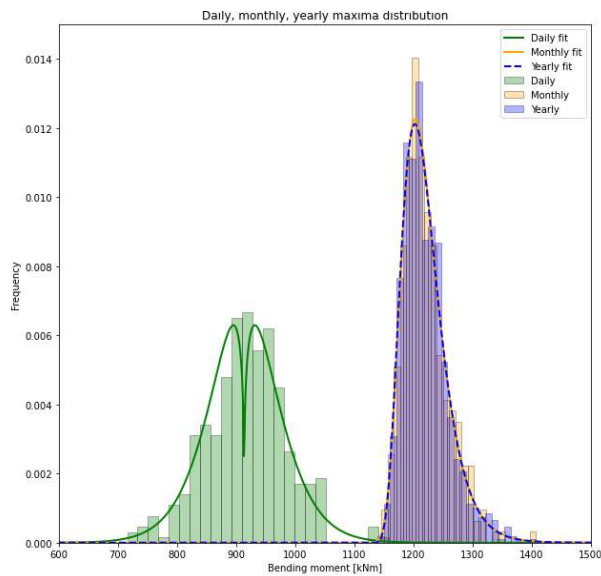


Figure 93 - Relative frequency histograms of daily-, monthly- and yearly maxima together with their respective fitted distribution.

The figure above shows that the monthly- and yearly maxima distributions are almost identical. This follows logically from the fact that the yearly maxima distribution is based on the monthly maxima distribution.

## Appendix F – LP data: Distribution fitting

### Daily maxima

When calculating the daily maxima, the following distribution is obtained. This rather skewed distribution is obtained due to static values for axles. Consequently, distribution fitting becomes rather difficult, as shown in the figures below.

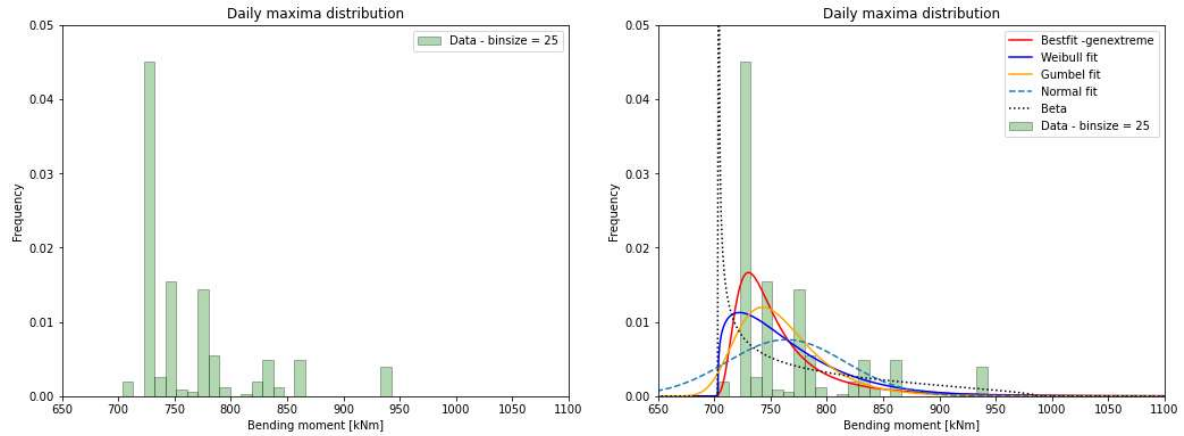


Figure 94 - Left: Relative frequency histogram of occurring load effect using LP model. Right: Relative frequency histogram with fitted distributions.

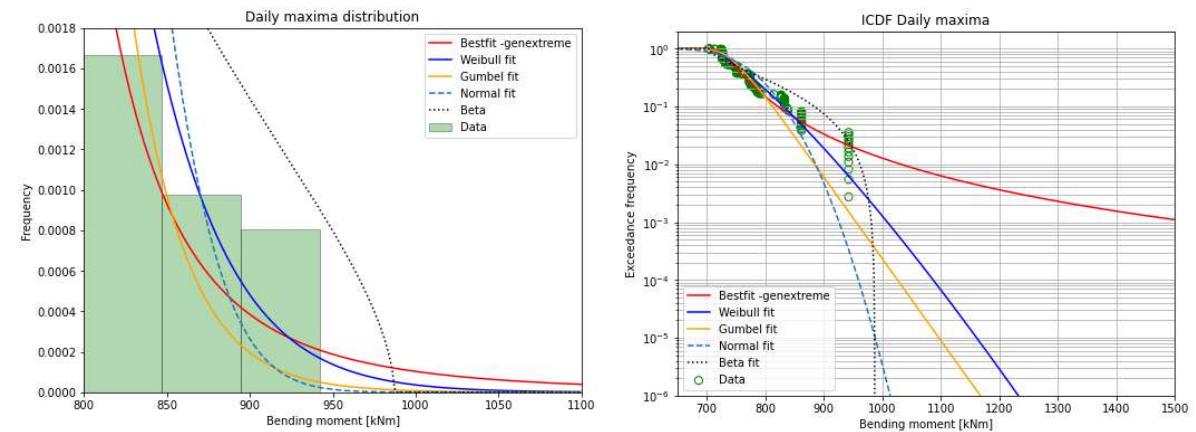


Figure 95 - Left: Tail of relative frequency histogram of daily maxima using LP model. Right: inverted cumulative distribution of daily maxima from LP data and fitted distributions.

From the top right figure, it follows that the best fitted distribution is the *generalized extreme value distribution*. The GEV-distribution does indeed describe the peak at 740 kNm properly, however is extremely inaccurate describing the tail, as shown in the bottom right figure. The Weibull distribution seems to fit the tail best, where the other distributions do not. Hence for the daily maxima a Weibull distribution is chosen with the following parameters<sup>37</sup>.

Parameter	Abbreviation (code)	Value
Shape	$c$	-0.3796551175784295
Location	$loc$	737.6251634809807
Scale	$scale$	23.573497456350264

Table 44 - Distribution parameters for fitted daily maxima distribution for LP model.

<sup>37</sup> [https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.weibull\\_min.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.weibull_min.html)

## Monthly maxima

The same procedure as for the daily maxima is now applied to determine the monthly maxima for a reference period of  $t$  years. In this example, a return period of 25 years is applied. Thus 25 years of monthly maxima is generated. With making use of the *distfit* package, several probability distributions are fitted. The following results are obtained.

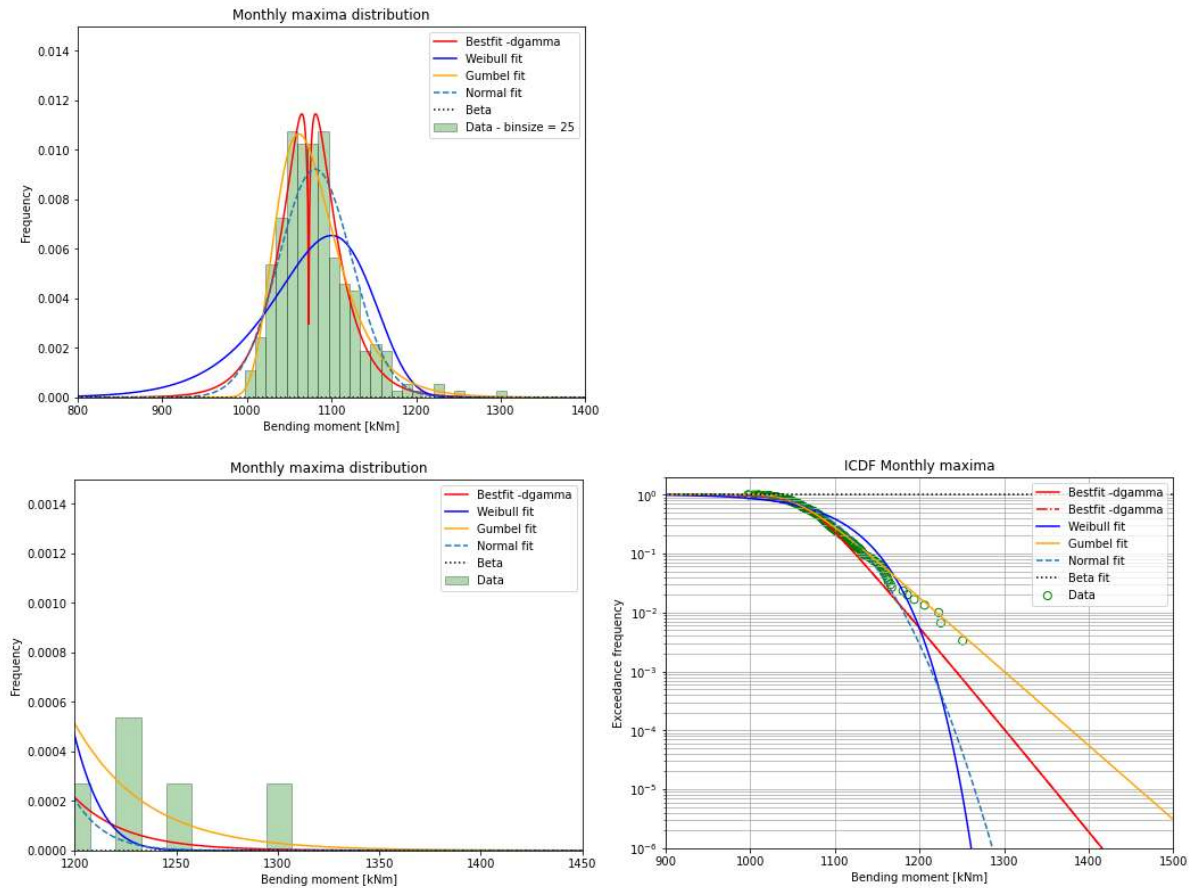


Figure 96 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for monthly maxima. Bottom left: Enhanced view of tail. Bottom right: inverted cumulative distribution of monthly maxima and fitted distributions.

The best fitted distribution is the *double gamma*<sup>38</sup> distribution, which is likely describing occurring the double peaks, which aren't visible in the histogram. However, when looking at the bottom graphs, the double gamma distribution seems to underestimate the occurring bending moments. The fitted Gumbel distribution however seems to accurately fit the tail, as well as fitting the 'main' peak in the top left figure. Hence, a fitted Gumbel distribution with the following parameters is adopted for the monthly maxima.

Parameter	Abbreviation (code / math)	Value
Location	$loc / \mu$	1061.1749837873122
Scale	$scale / \beta$	34.576516445534956

Table 45 - Distribution parameters for fitted monthly maxima distribution for LP model.

<sup>38</sup> <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.dgamma.html#scipy.stats.dgamma>

## Yearly maxima

To determine the yearly maxima, the same procedure as for the monthly maxima is applied. The Gumbel-distribution for the monthly maxima with the parameters denoted in table 45 act as an input. 1 year is simulated by means of drawing 12 monthly maxima from the given Gumbel-distribution and storing the maximum of the monthly maxima. This is repeated for 1000 years. With making use of the *distfit* package, several distributions are fitted to the data. The results are shown below.

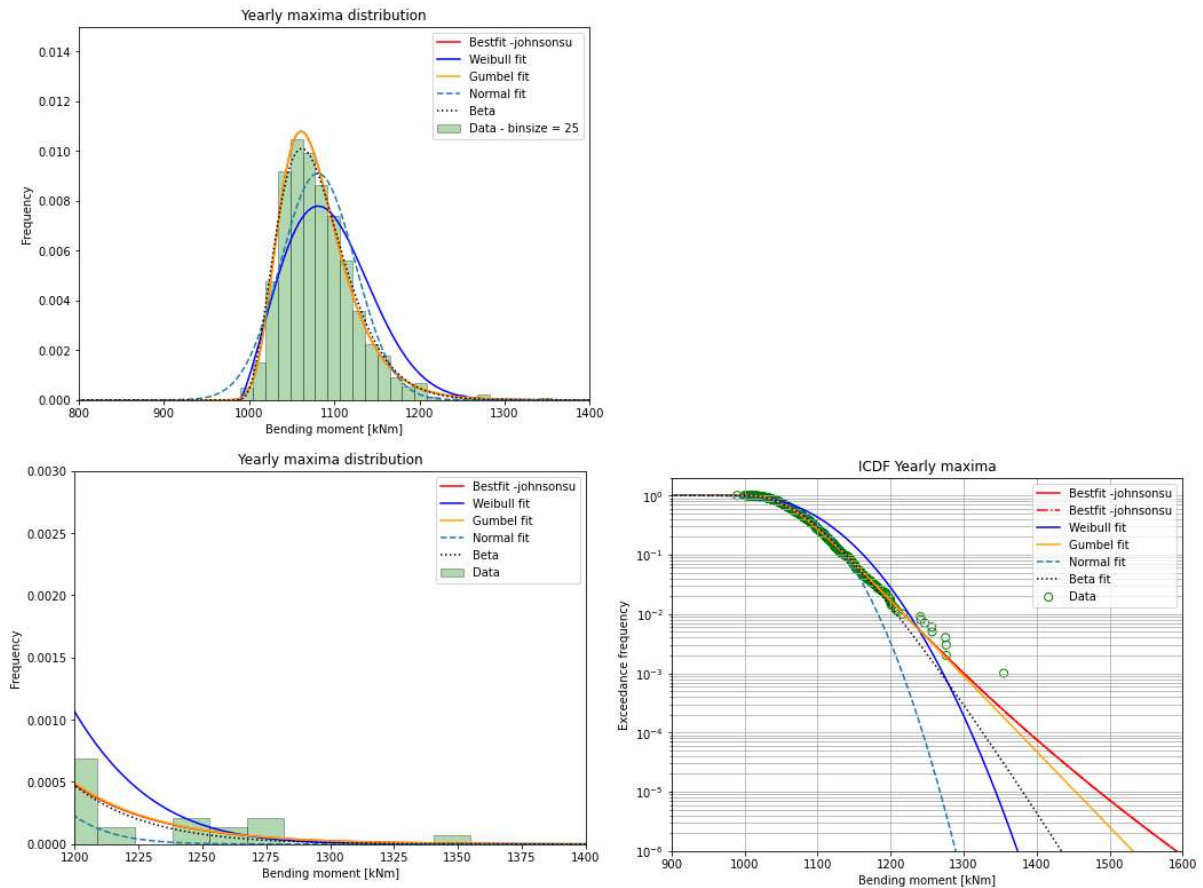


Figure 97 - Compilation of 3 figures. Top: Relative frequency histogram and fitted distributions for yearly maxima. Bottom left: Enhanced vision of tail. Bottom right: inverted cumulative distribution of yearly maxima and fitted distributions.

The calculated best fitted distribution is the *johnsonsu*<sup>39</sup> distribution, which except for the tail, is almost identical to the fitted Gumbel distribution. For the tail however, the *johnsonsu* distribution seems to accurately describe the tail. However, since the differences between the Gumbel- and the *johnsonsu* distribution are minor, a fitted Gumbel distribution with the following parameters is adopted for the yearly maxima.

Parameter	Abbreviation (code / math)	Value
Location	$loc / \mu$	1061.1361185982641
Scale	$scale / \beta$	34.07193556661929

Table 46 - Distribution parameters for best fitted yearly maxima distribution for LP model.

## Appendix G – WIM data: Distribution fitting

### Daily maxima

When calculating the daily maxima, the following distribution and fitted curves are obtained. The same procedure as for the modified LP data and LP data is used.

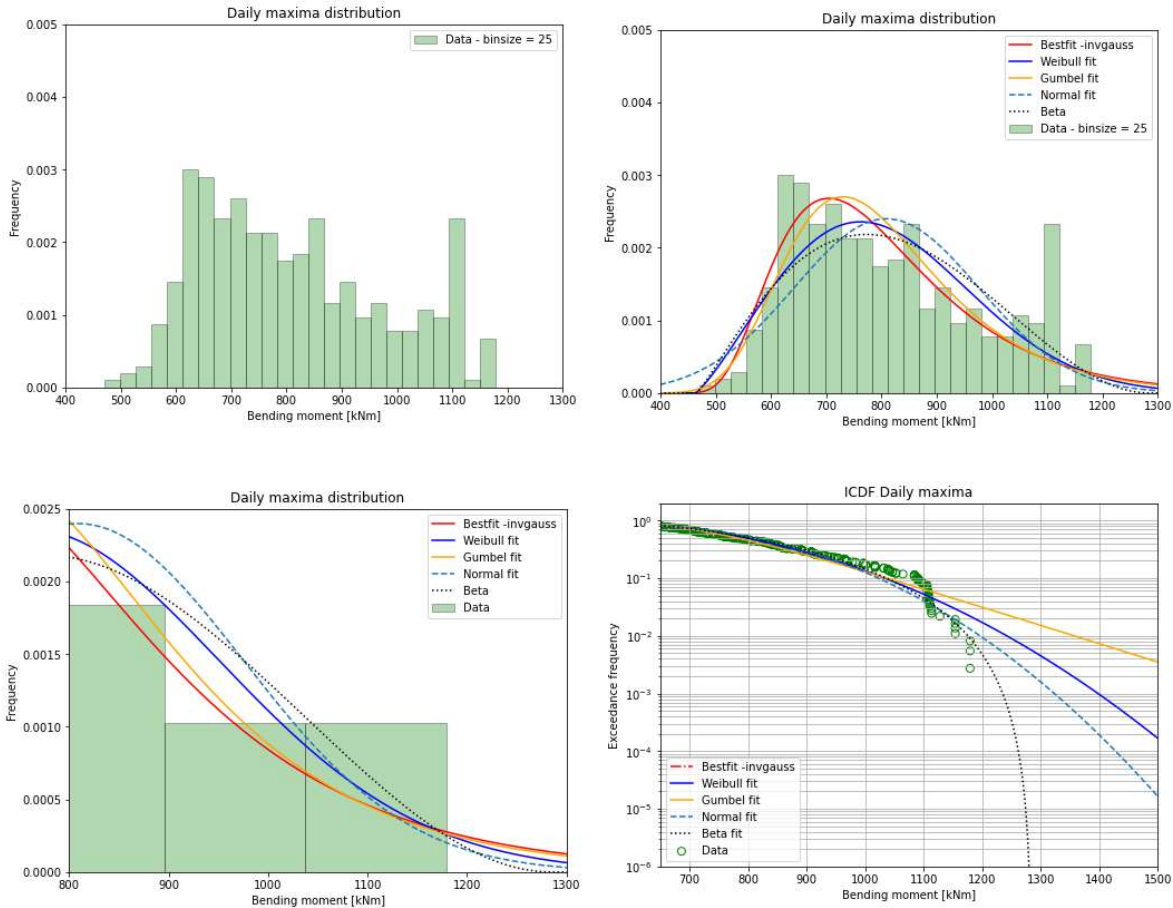


Figure 98 – Top left: Relative frequency histogram of occurring load effect using WIM model. Top right: Relative frequency histogram with fitted distributions. Bottom left: Tail of relative frequency histogram of daily maxima using WIM model. Bottom right: inverted cumulative distribution of daily maxima from WIM model and fitted distributions.

According to the *distfit* package, the best fitted distribution is the *invgauss*<sup>40</sup> (Inverse Gaussian Distribution). This distribution does indeed describe the peak at around 720 kNm well. As does the fitted Gumbel distribution. However, the tail is not properly described by the Gumbel and the Inverse Gaussian fits, as shown in the bottom right figure. The fitted Beta distribution seems to be fit the best, as well as the fitted unimodal normal distribution. However, fitted Beta distribution is capped at 1300 kNm which in practice might be exceeded. Hence the fitted normal distribution with the following parameters is chosen:

Parameter	Abbreviation (code)	Value
Location	$loc / \mu$	809.0168630136988
Scale	$scale / \sigma$	166.4033149501301

Table 47 - Distribution parameters for best fitted daily maxima distribution for WIM model.

<sup>40</sup> <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.invgauss.html>



## Monthly maxima

The fitted normal distribution acts as an input to determine the monthly maxima. A return period of 25 years is applied. Thus 25 years of monthly maxima is generated. With making use of the *distfit* package, several probability distributions are fitted. The following results are obtained.

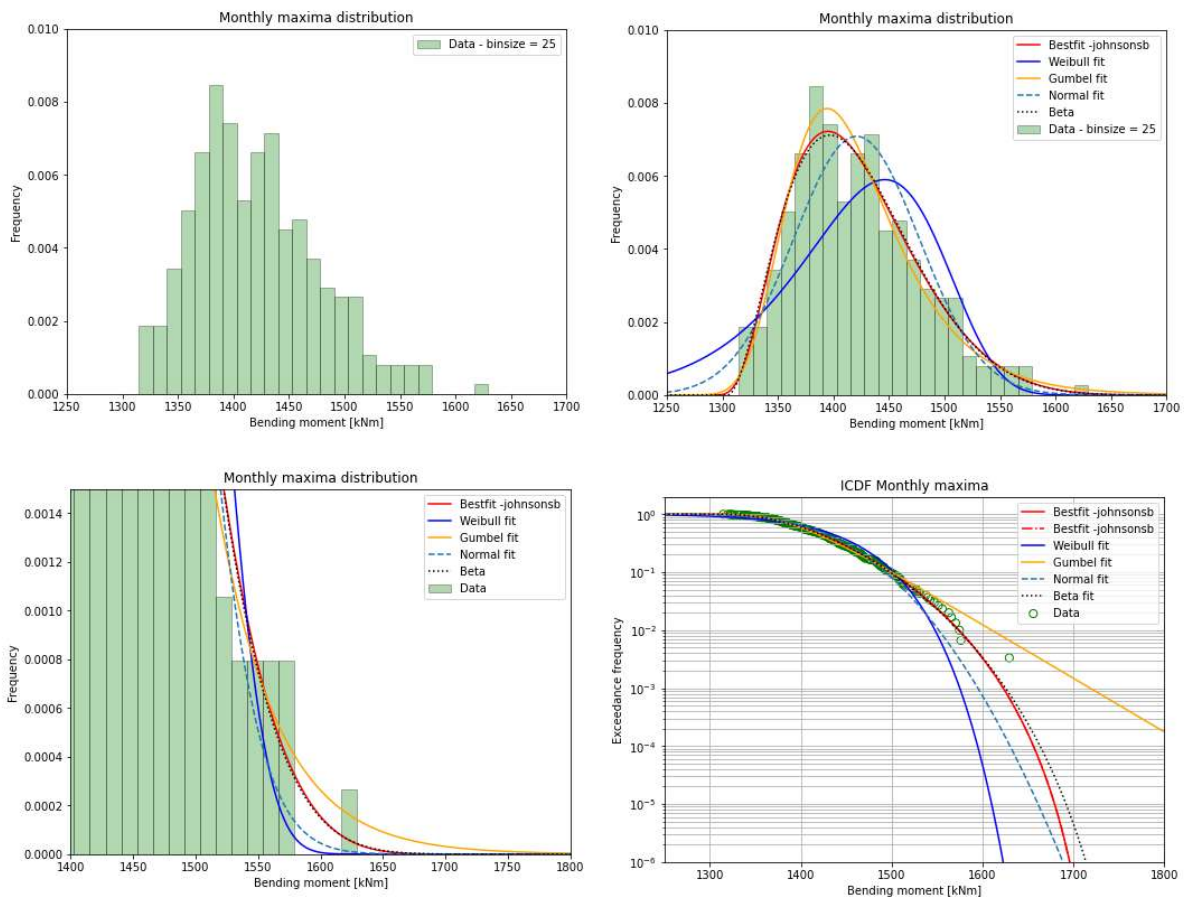


Figure 99 – Top left: Relative frequency histogram of monthly maxima using WIM model. Top right: Relative frequency histogram with fitted distributions. Bottom left: Tail of relative frequency histogram of monthly maxima using WIM model. Bottom right: inverted cumulative distribution of monthly maxima from WIM model and fitted distributions.

The best fitted distribution is the *johnsonsb*<sup>41</sup> distribution, a modified normal distribution. As can be seen, the *johnsonsb* fit is close to the fitted Beta distribution. Both distributions however underestimate the tail, especially for the highest occurring bending moment. The fitted Gumbel distribution however does, although slightly overestimating, describe the tail properly as well as describing the peak of the distribution properly. Hence, a fitted Gumbel distribution with the following parameters is applied.

Parameter	Abbreviation (code / math)	Value
Location	$loc / \mu$	1394.2954263955064
Scale	$scale / \beta$	46.9113226747704

Table 48 - Distribution parameters for best fitted monthly maxima distribution for WIM model.

<sup>41</sup> <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.johnsonsb.html#scipy.stats.johnsonsb>

## Yearly maxima

To determine the yearly maxima, the same procedure as for the monthly maxima is applied. The Gumbel-distribution for the monthly maxima with the parameters denoted in table 48 act as an input. 1 year is simulated by means of drawing 12 monthly maxima from the given Gumbel-distribution and storing the maximum of the monthly maxima. This is repeated for 1000 years. With making use of the *distfit* package, several distributions are fitted to the data. The results are shown below.

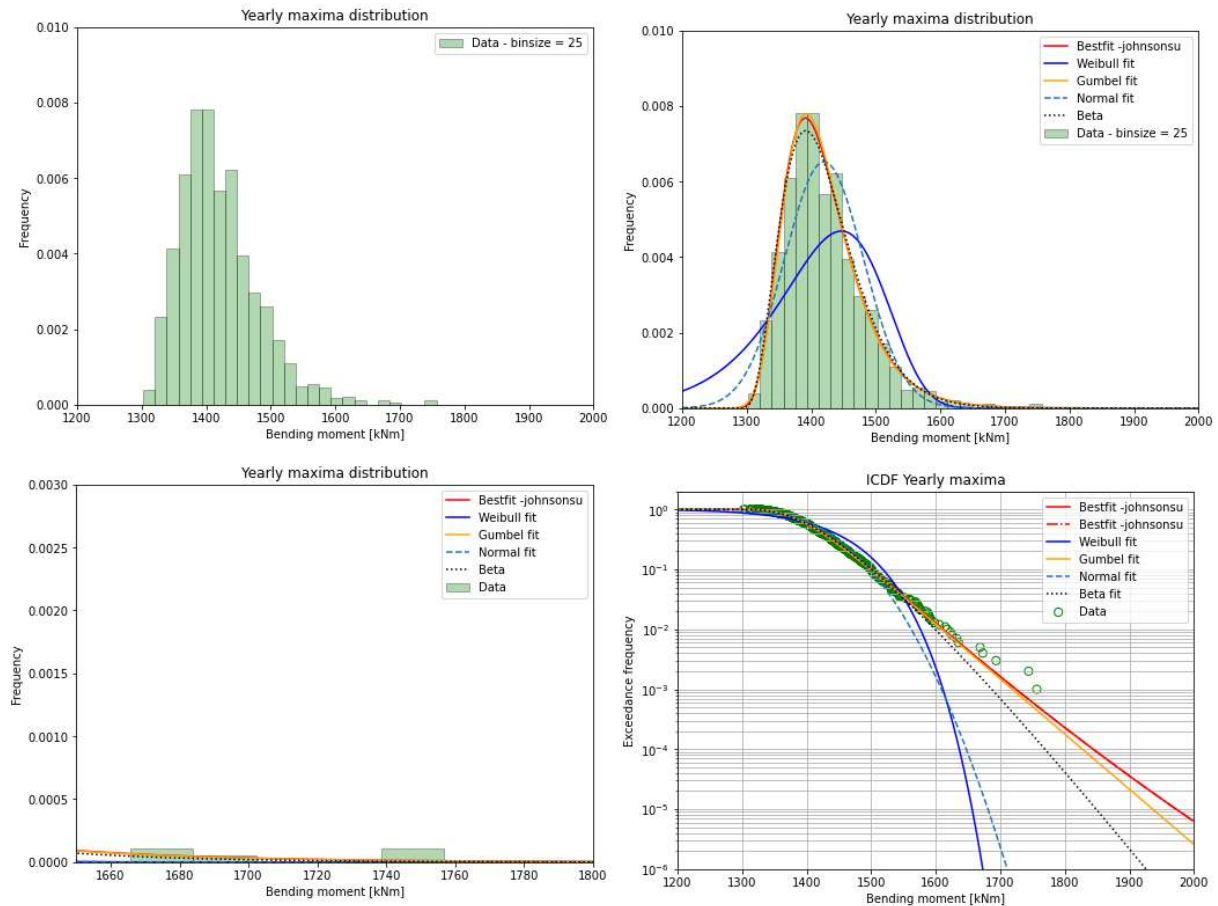


Figure 100 - Top left: Relative frequency histogram of yearly maxima using WIM model. Top right: Relative frequency histogram with fitted distributions. Bottom left: Tail of relative frequency histogram of yearly maxima using WIM model. Bottom right: inverted cumulative distribution of yearly maxima from WIM model and fitted distributions.

Using the *distfit* package, the best fitted distribution is the *johnsonsu* distribution. However, the bottom right figure shows that the fitted Gumbel distribution approximates the *johnsonsu* distribution as well. Both fits are slightly underestimating the tail of the distribution. However, all other functions are poorly fitted, and thus the fitted Gumbel distribution with the following parameters is adopted.

Parameter	Abbreviation (code / math)	Value
Location	$loc / \mu$	1392.3859985293516
Scale	$scale / \beta$	47.22886254482658

Table 49 - Distribution parameters for best fitted yearly maxima distribution for WIM model.



## Appendix H – Examples of $\eta_i$ for different vehicle types

In this appendix examples of the distribution for  $\eta_i$  will be given for each vehicle type used in this report. In total, three vehicle types are considered. The semi-trucks, both 2- and 3-axled, the full-trucks, both 2- and 3-axled, mobile cranes and tipper trucks. It must be noted that the figures below show distributions for **one registered vehicle** only, since the distribution is dependent on the axle load that follows from the LP data. For example, in figure 99 the first axle load is given as 75 kN. If the axle load of the first axle was set to be 115 kN, one could imagine that the distribution shifts to the left, as less values exceed this value.

The first considered vehicle type is the 2-axled semi-truck carrying any kind of trailer. The figure <sup>42</sup>below shows a 2-axled semi-truck carrying a single axle trailer, with the distribution of  $\eta_i$  for all three different axles. The red lines indicated the value of the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

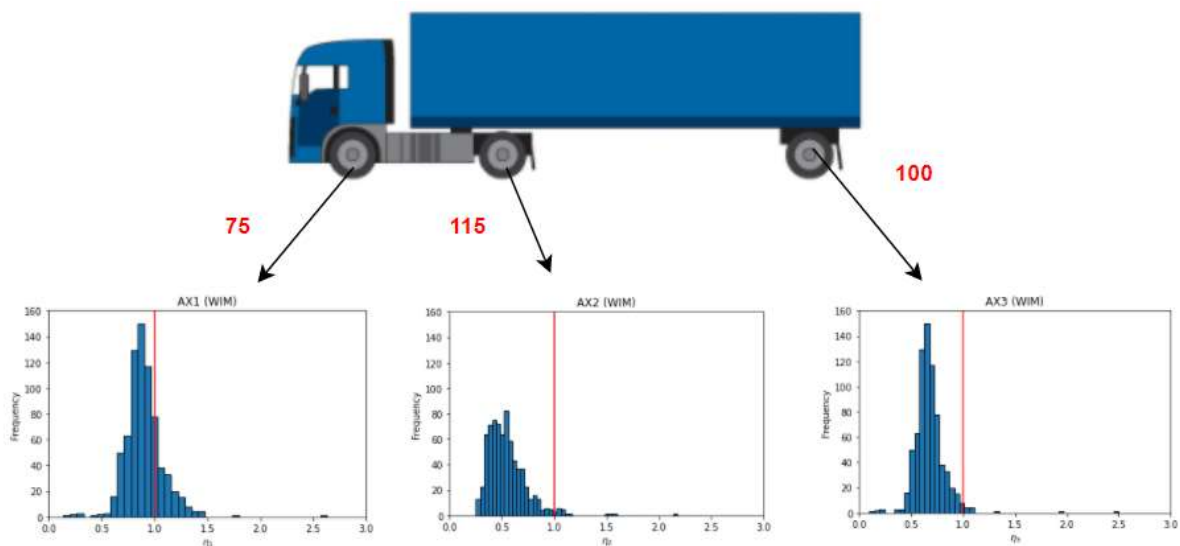


Figure 101- Distribution of  $\eta_i$  for a 2-axled semi-truck carrying a single axle trailer.

In the figure below the distribution of  $\eta_i$  is given for a 3-axled semi-truck carrying a 2-axled semi-trailer. The red lines indicated the value of the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

<sup>42</sup> Source of figures:

[http://onlinemanuals.txdot.gov/txdotmanuals/tri/images/FHWA\\_Classification\\_Chart\\_FINAL.png](http://onlinemanuals.txdot.gov/txdotmanuals/tri/images/FHWA_Classification_Chart_FINAL.png)

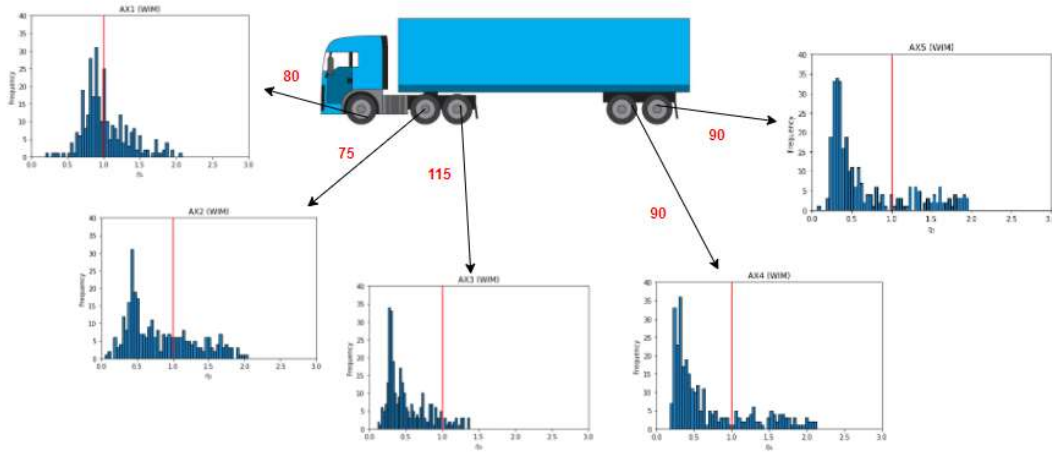


Figure 102 - Distribution of  $\eta_i$  for a 3-axled semi-truck carrying a 2-axled trailer.

In the figure below the distribution of  $\eta_i$  is given for a 2-axled full-truck carrying no trailer. The red lines indicated the value of the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

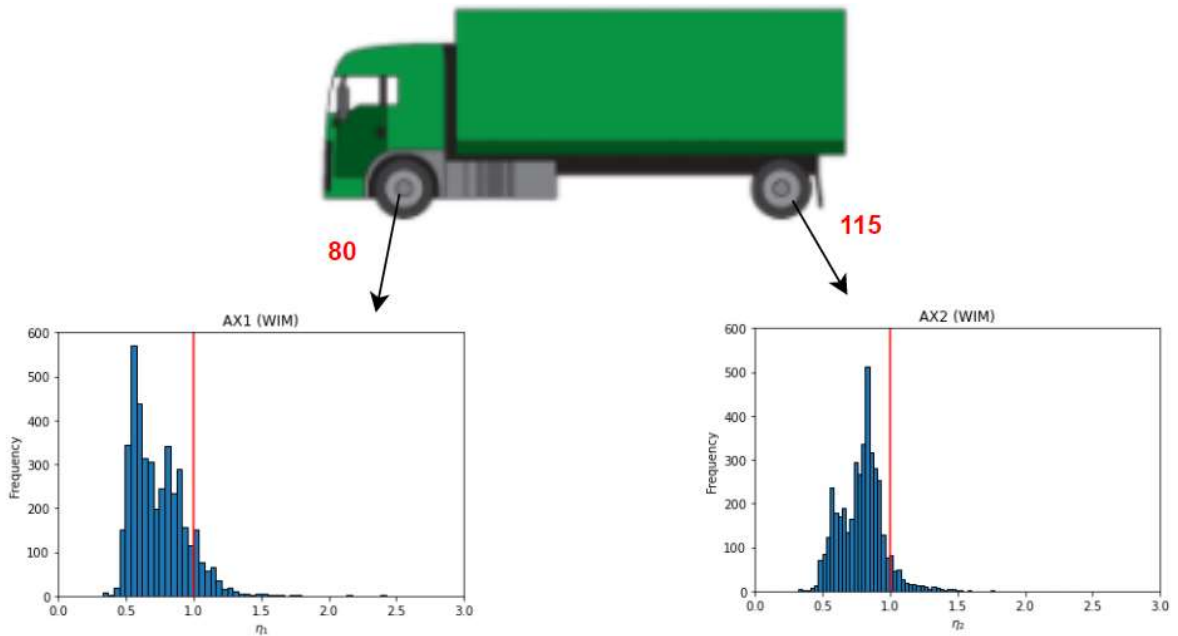


Figure 103 – Distribution of  $\eta_i$  for a 2-axled full truck carrying no trailer.

In the figure below the distribution of  $\eta_i$  is given for a 3-axled full-truck carrying no trailer. The red lines indicated the value of the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

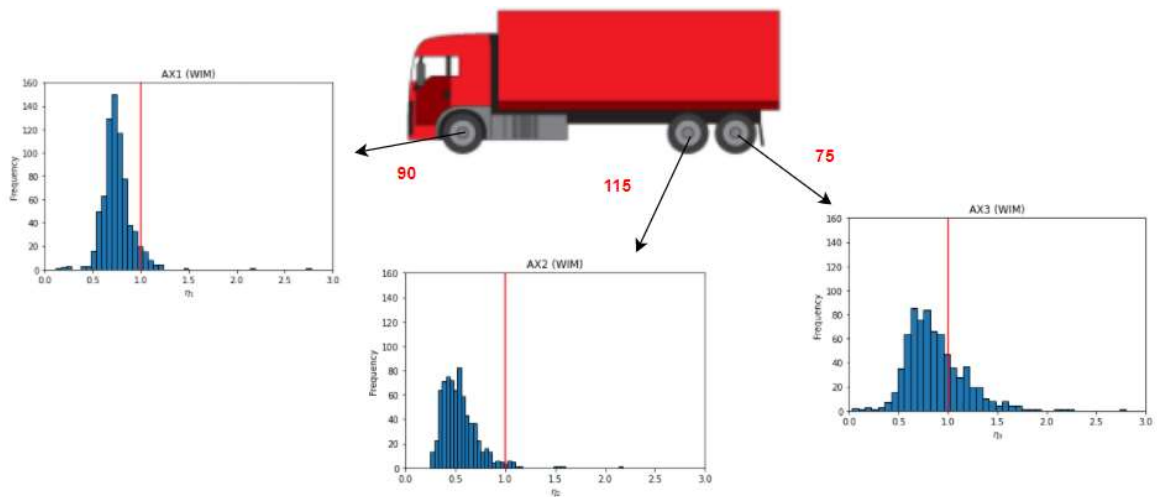


Figure 104 – Distribution of  $\eta_i$  for a 3-axled full truck carrying no trailer.

In the figure <sup>43</sup>below the distribution of  $\eta_i$  is given for a 4-axled tipper truck carrying no trailer. The red lines indicated the value of the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

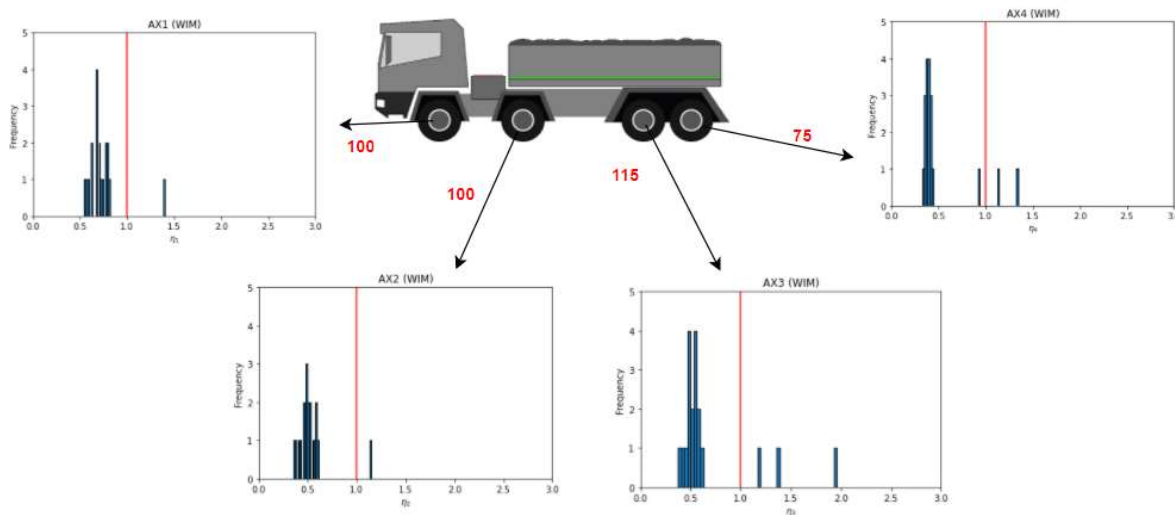


Figure 105 – Distribution of  $\eta_i$  for a 4-axled tipper/dump truck carrying no trailer.

In the figure <sup>44</sup>below the distribution of  $\eta_i$  is given for a 5-axled mobile crane carrying no trailer. The red lines indicated the value of the axle load as given by the LP data, i.e.  $\eta_i = 1.0$ .

<sup>43</sup> Source of figure: <https://www.grabco.co.uk/grab-hire>

<sup>44</sup> Source of figure: <https://www.1999.co.jp/eng/10429274>

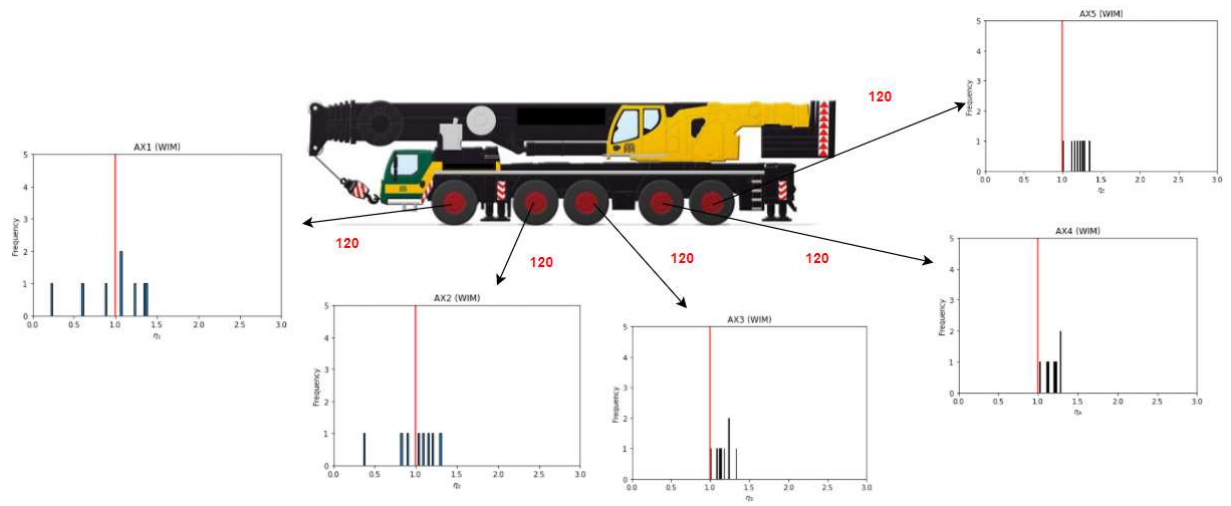


Figure 106 – Distribution of  $\eta_i$  for a 5-mobile crane carrying no trailer.

## Appendix I – Python codes

# File 1 - Constructing load model.

Calculating load effects for L=10. New version including under-overloading parameter  $\eta$ .

```
In [ ]: import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
import random
import math
import matplotlib.pyplot as plt
```

```
In [ ]: df_N3 = pd.read_csv('2019_4400C_N3.csv', sep=',')
df_O4 = pd.read_csv('2019_4400C_O4.csv', sep=',')
df_wim = pd.read_csv('WIM.csv')
```

```
In [ ]: df_wim = df_wim[df_wim['GVW'] > 120]
```

```
In [ ]: df_O4 = df_O4.drop(columns=['New_ID', 'ID', 'Pass ID', 'First(VehicleType)', 'First(MaxA
, 'First*(VehicleLength)'])
df_n3op1 = df_N3[(df_N3['Soort'] == 'opleggertrekker') & (df_N3['Oplegger [kg]'] > 0)
df_o4op1 = df_O4[(df_O4['Voertuigsoort'] == 'Oplegger') & (df_O4['Wielbasis [m]'] > 0)

df_kip = df_N3[df_N3['Soort'] == 'kipper']

df_O4_xtr = df_O4[df_O4['Maximum mass [kg]'] > 70000]
df_N3_xtr = df_N3[(df_N3['Axles'] == 4) & (df_N3['Soort'] == 'opleggertrekker')]

#Vereenvoudigen code
a = df_n3op1
b = df_o4op1
d = df_kip
```

```
In [ ]: #Voor N3: kolommen 5 t/m 13 zijn F-Lasten
#voor O4: kolommen 3 t/m 11 zijn F-Lasten
F_N3 = a[['F1 [kg]', 'F2 [kg]', 'F3 [kg]', 'F4 [kg]', 'F5 [kg]', 'F6 [kg]', 'F7 [kg]', 'F8
F_N3 = F_N3.to_numpy()
```

The cell below shows a script used to couple a random semi-truck to a random semi-trailer. N3 is governing and datetime stamp of N3 vehicle is used. A new loadsample is created. Interspatial axle distances are based on: The cell below is inserted to calculate load factor  $\eta$ . This is now done for 2- and 3-axled semi-trucks. For 4-axled semi-trucks no comparison was made.

- N3: number of axles
- N3: wheelbase
- O4: number of axles
- O4: wheelbase

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111) | (df_wim['CONFIG'] == 1121) |
(df_wim['CONFIG'] == 311) | (df_wim['CONFIG'] == 11111) |
(df_wim['CONFIG'] == 411) | (df_wim['CONFIG'] == 111111) |
(df_wim['CONFIG'] == 1111) | (df_wim['CONFIG'] == 121) | (df_wim['CONFIG']
(df_wim['CONFIG'] == 123) | (df_wim['CONFIG'] == 124) |
(df_wim['CONFIG'] == 1211) | (df_wim['CONFIG'] == 1221) |
(df_wim['CONFIG'] == 12111) | (df_wim['CONFIG'] == 121111)]
df_fw = np.zeros((0,36))
df_adds = np.zeros((0,10))
for k in range(1): #aantal keer
for j in range(len(df_n3op1)): #Len(df_n3op1)
```

```

fn3 = F_N3[j]; fn3 = np.trim_zeros(fn3,'b') #aslasten N3 selecteren
maxmassn3 = a.iloc[j,22] #Oplegger [kg]

subset = b[b['Maximum mass [kg]'] <= maxmassn3] #per n3 een subset maken van
d = subset.sample(n=1) #Random entry uit de passages pakken

fo4 = np.array([d.iloc[0,3],d.iloc[0,4],d.iloc[0,5],d.iloc[0,6],d.iloc[0,7],
fo4 = np.trim_zeros(fo4) #trimmen van 0

t=18 #max aantal assen
c = np.hstack((fn3,fo4)) #samenvoegen van de aslasten F1 t/m F18
rest = np.zeros(t- len(c))
c = np.hstack((c,rest)) #klaarmaken van de aslasten F1 t/m F18

lp_n3 = a.iloc[j,4] #lp n3
lp_o4 = d.iloc[0,2] #lp o4
date = a.iloc[j,2] #date n3
time = a.iloc[j,3] #time n3
axn3 = a.iloc[j,23] #axle n3
ln3 = a.iloc[j,15]/100 #Length n3
wn3 = a.iloc[j,16]/100 #wheelbase n3
lo4 = d.iloc[0,14]/100 #Length o4
wo4 = d.iloc[0,16]/100 #wheelbase o4
srt = a.iloc[j,18] #Type of N3 vehicle

o4_axle = (np.count_nonzero(c)) - axn3 #determining number of axles of trail
wmatrix = np.where(c > 0,1,0);wmatrix=wmatrix[1:1000] #maken I-matrix: hoeve
wnew = np.zeros((18))
wxn3 = np.random.uniform(1.25,1.35,1)
wxo4 = np.random.uniform(1.25,1.35,1)

#2 assen N3
if axn3 == 2:
    for j in range(len(wnew)-1):
        wnew[0] = wn3
        wnew[j+1] = wo4 - (o4_axle-(j+1))*wxo4
        wnew = np.where(wnew <= wo4,wnew,0)
        wnew[2:18] = np.where(wnew[2:18] > 0,wxo4,0)

#3 assen N3
if axn3 == 3:
    for j in range(len(wnew)-2):
        wnew[0] = wn3-wxn3
        wnew[1] = wxn3
        wnew[j+2] = wo4 - (o4_axle-(j+1))*wxo4
        wnew = np.where(wnew <= wo4,wnew,0)
        wnew[3:18] = np.where(wnew[3:18] > 0,wxo4,0)

#4 assen N3
if axn3 == 4:
    for j in range(len(wnew)-3):
        wnew[0] = wn3-wxn3
        wnew[1] = wxn3
        wnew[2] = wxn3
        wnew[j+3] = wo4 - (o4_axle-(j+1))*wxo4
        wnew = np.where(wnew <= wo4,wnew,0)
        wnew[4:18] = np.where(wnew[4:18] > 0,wxo4,0)

axtot = axn3+o4_axle; ee = aa[aa['Axles'] == axtot]
if (np.count_nonzero(c)) < 7:
    for r in range(axtot):
        vec = ee.iloc[:,3+r].to_numpy()*100 / (c[r])
        #print(vec)
        alpha = random.choice(vec)
        c[r] =c[r] * alpha

```



```

adds = np.array([lp_n3,lp_o4,date,time,axn3,ln3,wn3,lo4,wo4,srt])
c = np.hstack((c,wnew))
df_fw = np.vstack((df_fw,c))
df_adds = np.vstack((df_adds,adds))

```

```
In [ ]: df_n3op1
```

In the cell below tipper trucks, cement trucks etc. are added to the database.

```

In [ ]: aa = df_wim[((df_wim['Axles'] == 4) & (df_wim['DT12'] < 200) & (df_wim['DT23'] < 200)
& (df_wim['DT34'] < 200) & (df_wim['DT45'] < 200)) | ((df_wim['Axles'] == 5)
& (df_wim['DT34'] < 200) & (df_wim['DT45'] < 200))]

df_kip = df_N3[(df_N3['Soort'] == 'kipper') | (df_N3['Soort'] == 'asfaltkipper') | (
| (df_N3['Soort'] == 'betonmixer') | (df_N3['Soort'] == 'betonpomp')]

#Voor N3: kolommen 5 t/m 13 zijn F-Lasten
f_kip = df_kip[['F1 [kg]', 'F2 [kg]', 'F3 [kg]', 'F4 [kg]', 'F5 [kg]', 'F6 [kg]', 'F7 [kg]']]
f_kip = f_kip.to_numpy()
df_fw_kip = np.zeros((0,36))
df_adds_kip = np.zeros((0,10))
for k in range(1):
    for j in range(len(df_kip)): #Len(df_kip)
        fn3kip = f_kip[j]; fn3kip = np.trim_zeros(fn3kip,'b')
        t = 18
        rest = np.zeros((t - len(fn3kip)))
        c = np.hstack((fn3kip,rest))

        lp_n3 = df_kip.iloc[j,4] #lp n3
        date = df_kip.iloc[j,2] #date n3
        time = df_kip.iloc[j,3] #time n3
        axn3 = df_kip.iloc[j,23] #axle n3
        ln3 = df_kip.iloc[j,15]/100 #Length n3
        wn3 = df_kip.iloc[j,16]/100 #wheelbase n3
        lp_o4 = 0
        lo4 = 0
        wo4 = 0
        srt = df_kip.iloc[j,18] #N3 type

        wmatrix = np.where(c > 0,1,0);wmatrix=wmatrix[1:1000] #maken I-matrix: hoeve
wnew = np.zeros((18))
wxn3 = np.random.uniform(0.90,1.35,1)
#2 assen N3 kip
if axn3 == 2:
    for j in range(axn3-1):
        wnew[0] = wn3

#3 assen N3 kip. Looks Like regular truck. 2 axles closely placed at the bac
if axn3 == 3:
    for j in range(axn3-1):
        wnew[0] = wn3 - wxn3
        wnew[1] = wxn3

#4 assen N3 kip. assumes: axles evenly distributed
if axn3 == 4:
    for j in range(axn3-1):
        wnew[j] = wn3/(axn3-1)

#5 assen N3 kip. Assumes: 5 axles evenly distributed along the wheelbase
if axn3 == 5:
    for j in range(axn3-1):
        wnew[j] = wn3/(axn3-1)

```

```

if (np.count_nonzero(c)) > 3:
    axtot = axn3; ee = aa[aa['Axles'] == axtot]
    for r in range(axtot):
        vec = ee.iloc[:,3+r].to_numpy()*100 / (c[r])
        alpha = random.choice(vec)
        c[r] = c[r] * alpha
adds_kip = np.array([lp_n3,lp_o4,date,time,axn3,ln3,wn3,lo4,wo4,srt])
c = np.hstack((c,wnew))
df_fw_kip = np.vstack((df_fw_kip,c))
df_adds_kip = np.vstack((df_adds_kip,adds_kip))

```

In the cell below mobile cranes are added to the database.

```

In [ ]: aa = df_wim[((df_wim['Axles'] == 4) & (df_wim['DT12'] < 300) & (df_wim['DT23'] < 300)
                & (df_wim['DT34'] < 300) & (df_wim['DT45'] < 300) & (df_wim['LENGTH'] >
                & (df_wim['DT34'] < 300) & (df_wim['DT45'] < 300) & (df_wim['LENGTH'] >
df_krn = df_N3[df_N3['Soort'] == 'mobiele kraan']
#Voor N3: kolommen 5 t/m 13 zijn F-Lasten
f_krn = df_krn[['F1 [kg]', 'F2 [kg]', 'F3 [kg]', 'F4 [kg]', 'F5 [kg]', 'F6 [kg]', 'F7 [kg]']]
f_krn = f_krn.to_numpy()
df_fw_krn = np.zeros((0,36))
df_adds_krn = np.zeros((0,10))

for k in range(1):
    for j in range(len(df_krn)):
        fn3krn = f_krn[j]#; fn3krn = np.trim_zeros(fn3krn, 'b')
        t = 18
        rest = np.zeros((t - len(fn3krn)))
        c = np.hstack((fn3krn,rest))

        lp_n3 = df_krn.iloc[j,4] #lp n3
        date = df_krn.iloc[j,2] #date n3
        time = df_krn.iloc[j,3] #time n3
        axn3 = df_krn.iloc[j,23] #axle n3
        ln3 = df_krn.iloc[j,15]/100 #Length n3
        wn3 = df_krn.iloc[j,16]/100 #wheelbase n3
        lp_o4 = 0
        lo4 = 0
        wo4 = 0
        srt = df_krn.iloc[j,18] #N3 type

        wmatrix = np.where(c > 0,1,0);wmatrix=wmatrix[1:1000] #maken I-matrix: hoeve
        wnew = np.zeros((18))
        wxn3 = np.random.uniform(1.40,1.75,1) #Let op: onderwaarde 1.40, bovenwaarde

        #2 assen N3 kraan
        if axn3 == 2:
            for r in range(axn3 -1):
                wnew[0] = wn3

        #3 assen N3 kraan. Looks like regular truck.
        if axn3 == 3:
            for r in range(axn3 -1):
                wnew[0] = wn3 - wxn3
                wnew[1] = wxn3

        #4&5 assen N3 kraan. Assumes axles evenly distributed
        if axn3 in range(4,7):
            for r in range(axn3 - 1):
                wnew[r] = wn3 / (axn3-1)

        if (axn3 == 5) & (df_krn.iloc[j,22] > 5000):
            wnew[axn3-1] = 6.68 - 2*(5.84/3) #w5
            wnew[5] = 1.95 #w6
            wnew[6] = 1.95 #w7

```

```

    for r in range(3):
        c[axn3+r] = 8e3 #aslast van 8000

    if 3 < axn3 < 6:
        axtot = axn3; ee = aa[aa['Axles'] == axtot]
        for r in range(axtot):
            vec = ee.iloc[:,3+r].to_numpy()*100 / (c[r])
            alpha = random.choice(vec)
            c[r] =c[r] * alpha
        adds_krn = np.array([lp_n3,lp_o4,date,time,axn3,ln3,wn3,lo4,wo4,srt])
        c = np.hstack((c,wnew))

df_fw_krn = np.vstack((df_fw_krn,c))
df_adds_krn = np.vstack((df_adds_krn,adds_krn))

```

In the cell below full trucks are added to the database.

```

In [ ]: aa = df_wim[(df_wim['CONFIG'] == 11) | (df_wim['CONFIG'] == 111) | (df_wim['CONFIG']
    (df_wim['CONFIG'] == 2111) | (df_wim['CONFIG'] == 12) | (df_wim['CONFIG']
    (df_wim['CONFIG'] == 2121) | (df_wim['CONFIG'] == 31) | (df_wim['CONFIG'] =
    (df_wim['CONFIG'] == 2122)]

n3list = ['gesloten opbouw','gecond. met temperatuurreg.','geconditioneerd voertuig'
    'huifopbouw','voor vervoer voertuigen']
#maken van F_matrix (f_vv) en bijbehorende slices van database df_vv
#maken van o4 database met (autonome aanhangers / aanhangers / middenasaanhangwagen)
f_vv = np.zeros((1,9))
df_vv = df_N3[:1]
for r in range(len(n3list)):
    df_vv = df_N3[df_N3['Soort'] == n3list[r]]
    a = (df_vv.iloc[:,5:14]) ; a = a.to_numpy()
    f_vv = np.vstack((f_vv,a))
    df_vv = df_vv.append(df_vv)

a = df_N3[df_N3['Soort'] == n3list[0]]
for r in range(len(n3list)-1):
    df_vv = df_N3[df_N3['Soort'] == n3list[r+1]]
    a = a.append(df_vv)
df_vv = a
f_vv = f_vv[1:]

```

```

In [ ]: df_fw_vv = np.zeros((0,36))
df_adds_vv = np.zeros((0,10))
for k in range(1): #aantal keer
    for j in range(len(df_vv)): #Len(df_vv)
        fn3 = f_vv[j]; fn3 = np.trim_zeros(fn3,'b')
        maxmasso4 = df_vv.iloc[j,17] - df_vv.iloc[j,14] #Max mass combi - Max mass N

        lp_n3 = df_vv.iloc[j,4] #lp n3
        date = df_vv.iloc[j,2] #date n3
        time = df_vv.iloc[j,3] #time n3
        axn3 = df_vv.iloc[j,23] #axle n3
        ln3 = df_vv.iloc[j,15]/100 #Length n3
        wn3 = df_vv.iloc[j,16]/100 #wheelbase n3
        srt = df_vv.iloc[j,18] #Type of N3 vehicle

        if maxmasso4 > (np.min(df_O4['Maximum mass [kg]'))): #n3 + o4
            subset = df_O4[df_O4['Maximum mass [kg]'] <= maxmasso4]
            d = subset.sample(n=1)
            fo4 = np.array(d.iloc[0,3:12]); fo4 = np.trim_zeros(fo4) #trimmen van 0
            t = 18 #max aantal assen
            c = np.hstack((fn3,fo4))

```

```

rest = np.zeros(t - len(c))
c = np.hstack((c,rest)) #klaarmaken aslasten F1 t/m F18

#04 info
lp_o4 = d.iloc[0,2] #lp o4
lo4 = d.iloc[0,14]/100 #length o4
wo4 = d.iloc[0,16]/100 #wheelbase o4
o4_axle = (np.count_nonzero(c)) - axn3 #determining number of axles of t

wmatrix = np.where(c > 0,1,0);wmatrix=wmatrix[1:1000] #maken I-matrix: h
wnew = np.zeros((18))
wxn3 = np.random.uniform(0.90,1.35,1)
wxo4 = np.random.uniform(0.90,1.35,1)

#2 assen N3 vv
if axn3 == 2:
    wnew[0] = wn3 #w1
    if (wo4/o4_axle < 1.3): #GECONCENTREERDE AANH
        wnew[1] = wxn3 +(lo4-wo4)/2 #w2
        for r in range(o4_axle -1): #w3 t/m rest
            wnew[r+2] = wo4
    else: #NIET GECONCENTREERDE AANH
        wnew[1] = wxn3 #w2 - drawbar
        if (o4_axle == 2): #2 ASSIGE AANH
            wnew[2] = wo4 #w3
        if (o4_axle == 3): #3 ASSIGE AANH
            wnew[2] = wo4 - wxo4 #w3
            wnew[3] = wxo4 #w4

#3 assen N3 vv
if axn3 == 3:
    wnew[0] = wn3 - wxn3 #w1
    wnew[1] = wxn3 #w2
    if (wo4/o4_axle < 1.3): #GECONCENTREERDE AANH
        wnew[2] = wxn3 + (lo4-wo4)/2 #w3
        for r in range(o4_axle -1): #w4 t/m rest
            wnew[r+3] = wo4 /(o4_axle -1) #assen gelijkmatig verdeeld
    else: #NIET GECONCENTREERDE AANH
        wnew[2] = wxn3 #w3 -drawbar
        if (o4_axle == 2): #2 ASSIGE AANH
            wnew[3] = wo4 #w4
        if (o4_axle == 3): #3 assige aanh
            wnew[3] = wo4 - wxo4 #w4
            wnew[4] = wxo4 #w5

#4 assen N3 vv
if axn3 == 4:
    for r in range(axn3 -1): #w1,w2,w3
        wnew[r] = wn3 / (axn3-1) #w3 = wnew[2]

    if (wo4/o4_axle < 1.3): #GECONCENTREERDE AANH
        wnew[3] = 1.5 # w4
        for r in range(o4_axle-1): #w5 t/m rest
            wnew[r+4] = wo4/(o4_axle -1)

    else: #NIET GECONCENTREERDE AANH
        wnew[3] = wxn3 #w4 - drawbar
        if (o4_axle ==2): #2 ASSIGE AANH
            wnew[4] = wo4 #w5
        if (o4_axle == 3): #3 ASSIGE AANH
            wnew[4] == wo4-wxo4 #w5
            wnew[5] = wxo4 #w6
        if (o4_axle == 4): #4 ASSIGE AANH
            wnew[4] = wxo4 #w5
            wnew[5] = wo4-2*wxo4 #w6
            wnew[6] = wxo4 #w7

```

```

#5 assen N3 vv
if axn3 == 5:
    for r in range(axn3 -1): #w1,w2,w3,w4
        wnew[r] = wn3 / (axn3 - 1) #gelijkmatig verdeeld

    if (wo4/o4_axle < 1.3): #GECONCENTREERDE AANH
        wnew[4] == (wxn3 + (lo4-wo4)/2) # w5
        for r in range(o4_axle-1): #w6 t/m rest
            wnew[r+5] = wo4/(o4_axle -1)

    else: #NIET GECONCENTREERDE AANH
        wnew[4] = wxn3 #w5 - drawbar
        if (o4_axle ==2): #2 ASSIGE AANH
            wnew[5] = wo4 #w6
        if (o4_axle == 3): #3 ASSIGE AANH
            wnew[5] == wo4-wxo4 #w6
            wnew[6] = wxo4 #w7
        if (o4_axle == 4): #4 ASSIGE AANH
            wnew[5] = wxo4 #w6
            wnew[6] = wo4-2*wxo4 #w7
            wnew[7] = wxo4 #w8

adds_vv = np.array([lp_n3,lp_o4,date,time,axn3,ln3,wn3,lo4,wo4,srt])

#####GEEN AANHANGER#####

else: #n3 , geen aanh
    c = fn3
    t = 18
    c = np.hstack((c,(np.zeros(t-len(c))))))

    lp_o4 = 0 #Lp o4
    lo4 = 0 #Length o4
    wo4 = 0 #wheelbase o4
    o4_axle = 0 #determining number of axles of trailer

    wmatrix = np.where(c > 0,1,0);wmatrix=wmatrix[1:1000] #maken I-matrix: h
    wnew = np.zeros((18))
    wxn3 = np.random.uniform(0.90,1.35,1)
    wxo4 = np.random.uniform(0.90,1.35,1)

#2 assen N3 vv
if axn3 == 2:
    for r in range(axn3 -1):
        wnew[0] = wn3

#3 assen N3 vv
if axn3 == 3:
    for r in range(axn3 - 1):
        wnew[0] =wn3 - wxn3
        wnew[1] = wxn3

#4 assen N3 vv. two pairs of closely spaced axles
if axn3 == 4:
    for r in range(axn3 -1):
        wnew[0] = wxn3 #w1
        wnew[1] = wn3 - 2*wxn3 #w2
        wnew[2] = wxn3 #w3

#5 assen N3 vv. Evenly distributed along wheelbase
if axn3 == 5:
    for r in range(axn3-1):
        wnew[r] = wn3/(axn3-1)

adds_vv = np.array([lp_n3,lp_o4,date,time,axn3,ln3,wn3,lo4,wo4,srt])

```

```

axtot = axn3+o4_axle; ee = aa[aa['Axles'] == axtot]

if axtot > 2:
    for r in range(axtot):
        vec = ee.iloc[:,3+r].to_numpy()*100 / (c[r])
        alpha = random.choice(vec)
        c[r] =c[r] * alpha
if axtot > 7:
    print(axtot)
c = np.hstack((c,wnew))
df_fw_vv = np.vstack((df_fw_vv,c))
df_adds_vv = np.vstack((df_adds_vv,adds_vv))

```

In the cell below the EHC category is added.

```

In [ ]: #Extreme gevallen is net andersom. Hierbij is 04 leidend en niet 03. Nu wordt bij 04

df_fw_xtr = np.zeros((0,36))
df_adds_xtr = np.zeros((0,10))

f_xtr = np.zeros((1,9))
for k in range(1):
    for j in range(len(df_04_xtr)): #Len(df_04_xtr)
        #selecteren van random N3 uit df_N3_xtr. Gaat nu net andersom. ALTIJD 4 asse
        d = df_N3_xtr.sample(n=1)

        #Gegevens N3
        lp_n3 = d.iloc[0,4] #lp n3
        axn3 = d.iloc[0,23] #axle n3
        ln3 = d.iloc[0,15]/100 #Length n3
        wn3 = d.iloc[0,16]/100 #wheelbase n3
        fn3 = (d.iloc[0,5:14]); fn3 = fn3.to_numpy(); fn3 = np.trim_zeros(fn3,'b')

        #Gegevens 04
        fo4 = (df_04_xtr.iloc[j,3:12]); fo4 = fo4.to_numpy(); fo4= np.trim_zeros(fo4
        wo4 = df_04_xtr.iloc[j,16]/100 #wheelbase o4
        lo4 = df_04_xtr.iloc[j,14]/100 #length o4
        lp_o4 = df_04_xtr.iloc[j,2] #license plate o4
        date = df_04_xtr.iloc[j,0] #date time o4
        time = 0
        srt = str('Extreme')
        t = 18 #max aantal assen
        c = np.hstack((fn3,fo4))
        rest = np.zeros(t-len(c))
        c = np.hstack((c,rest))

        o4_axle = (np.count_nonzero(c)) - axn3 #determining number of axles of trail
        wmatrix = np.where(c > 0,1,0);wmatrix=wmatrix[1:1000] #maken I-matrix: hoeve
        wnew = np.zeros((18))
        wxn3 = np.random.uniform(1.25,1.35,1)
        wxo4 = np.random.uniform(1.25,1.35,1)

        #vullen van wnew voor N3
        wnew[0] = wn3 - 2*wxn3 #w1
        wnew[1] = wxn3 #w2
        wnew[2] = wxn3 #w3

        if o4_axle ==5: #5 axles. evenly distributed
            wnew[3] = wxn3
            for r in range(o4_axle-1):
                wnew[r+4] = wo4/(o4_axle-1)

```



```

y = np.vstack((y1,y2))
for r in range(16): #17 w's vullen. Eindigt op kolom 35
    c = y[r+1] - df.iloc[j,r+19]
    y = np.vstack((y,c))
y = np.where(y<0,0,y) #negatieve waarden verwijderen
y = np.where(y>L,0,y) #waarden groter dan L verwijderen
return y

```

```

In [ ]: print(len(df))
        L=10

```

In the cell below a algorithm is written to determine McMax per entry.

```

In [ ]: df_adds = df.iloc[:,36:46]
df = df.iloc[:,0:36]
McMax = np.zeros(len(df))
F_matrix = (df_fw.iloc[:, : 18]*(10/1000)).to_numpy() #selecteren van alle belasting
for j in range(len(df)): #Len (df)
    print(j)
    y_matrix = Y(L) #y1 = y_matrix[1]

    Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
    Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

    #Av
    for r in range(18): #in totaal 18 F's, dus 18 Av.
        Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
    #Mc
    for r in range(18):
        Mc[r] = np.where(y_matrix[r] < (L/2),
                        (Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
                        Av[r]*(L/2))
        Mc[r] = np.where(Mc[r]<0,0,Mc[r])

    Mc = Mc.sum(axis=0) #opsommen
    Mc = np.amax(Mc)
    McMax[j] = Mc
df = pd.concat([df_fw,df_adds], axis =1)

```

```

In [ ]: df['McMax'] = McMax

```

```

In [ ]: collist = ['F1 [kg]', 'F2 [kg]', 'F3 [kg]', 'F4 [kg]', 'F5 [kg]', 'F6 [kg]', 'F7 [kg]', 'F8 [kg]',
                'w1 [m]', 'w2 [m]', 'w3 [m]', 'w4 [m]', 'w5 [m]', 'w6 [m]', 'w7 [m]', 'w8 [m]',
                'LP N3', 'LP O4', 'Date', 'Time', 'Axles', 'Length N3 [m]', 'Wheelbase N3 [m]']
df.columns=collist

```

```

In [ ]: df['Max Axle Load [kg]'] = np.sum((df.iloc[:, : 18]),axis =1)

```

```

In [ ]: #Maken scatterplot (TS)
srt = ['opleggertrekker', 'mobiele kraan', 'kipper']
ts_mc = df.tail(1)
ts_x = ts_mc['Max Axle Load [kg]']
ts_y = ts_mc['McMax [kNm]'] #+ (1/8 * (9) * L**2)

data = df[df['Soort'] == srt[0]]
opl_x = data['Max Axle Load [kg]']
opl_y = data['McMax [kNm]']

data = df[df['Soort'] == srt[1]]
krn_x = data['Max Axle Load [kg]']
krn_y = data['McMax [kNm]']

```



```

data = df[df['Soort'] == srt[2]]
kip_x = data['Max Axle Load [kg]']
kip_y = data['McMax [kNm]']

data = df[(df['Soort'] != srt[0]) & (df['Soort'] != srt[1]) & (df['Soort'] != srt[2])
          (df['Soort'] != 'Extreme')]
vv_x = data['Max Axle Load [kg]']
vv_y = data['McMax [kNm]']

data=df[df['Soort'] == 'Extreme']
xtr_x = data['Max Axle Load [kg]']
xtr_y = data['McMax [kNm]']

print(ts_y)
print(np.max(krn_y))

```

Removing false entries for  $w_i$ .

```

In [ ]: df = df.drop(df[df['w1 [m]'] == 0].index)
df = df.drop(df[df['w2 [m]'] < 0].index)
df = df.drop(df[df['w3 [m]'] < 0].index)
df = df.drop(df[df['w4 [m]'] < 0].index)

collist = df.columns
for r in range(18):
    df = df.reset_index(drop=True)
    t = df.iloc[:,(r+18):(r+18+1)]
    t2 = t[(t[collist[r+18]] > 0) & (t[collist[r+18]] < 0.75)]
    df.drop(t2.index, inplace=True)
#df.to_csv('df_4400C_Load.csv', sep=',', index=False)

```

```

In [ ]: import matplotlib.pyplot as plt

plt.figure(figsize=(13,5), dpi=80)

plot_ts = plt.scatter(ts_x/1e3,ts_y,alpha = 1, color='red',label='LM1 - Tandem', mar
plot_opl = plt.scatter(opl_x/1e3,opl_y,alpha = 0.5,facecolor='none',edgecolor='blue'
plot_krn = plt.scatter(krn_x/1e3,krn_y,alpha = 0.5, color = 'green', label='Mobile cr
plot_kip = plt.scatter(kip_x/1e3,kip_y,alpha = 0.5, facecolor='none',edgecolor = 'bla
plot_vv = plt.scatter(vv_x/1e3,vv_y,alpha = 0.5, facecolor='none',edgecolor = 'orang
plot_xtr = plt.scatter(xtr_x/1e3,xtr_y,alpha = 0.5, facecolor='none',edgecolor = 'pu

plt.xlim(0e3,110)
plt.ylim(0,2.2e3)

leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [t]')
plt.ylabel('Bending moment [kNm]')
plt.title('Bending moment vs GVW for a bridge of 10 meters')
plt.show()

```

```

In [ ]: df2 = pd.read_csv('df_4400C_load_LP.csv',low_memory=False)

```

```

In [ ]: plt.figure(figsize=(13,5), dpi=80)
plt.scatter(df['Max Axle Load [kg]']/1e3,df['McMax [kNm]'],facecolor='none',edgecolor=
alpha= 0.5, marker='^', label='WIM influenced LP')
#plt.scatter(df2['Max Axle Load [kg]'],df2['McMax [kNm]'],facecolor='none',edgecolor
#
alpha= 0.5, marker='o', Label='True LP')
plt.scatter(df_wim['GVW']/10,df_wim['McMax [kNm]'],facecolor='none',edgecolor='orang
alpha= 0.5, marker='s', label='True WIM')

```

```
plt.xlabel('GVW [t]')
plt.ylabel('Bending moment [kNm]')
plt.title('Bending moment vs GVW for a bridge of 10 meters');plt.legend(loc='upper l
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)
plt.show()
```

```
In [ ]: plt.figure(figsize=(6,6), dpi=80)
plt.scatter(df['Max Axle Load [kg]']/1e3,df['McMax [kNm]'],facecolor='none',edgecolor=
alpha= 1, marker='^', label='WIM influenced LP')
plt.scatter(df2['Max Axle Load [kg]']/1e3,df2['McMax [kNm]'],facecolor='none',edgecolor=
alpha= 0.5, marker='o', label='True LP')
plt.scatter(df_wim['GVW']/10,df_wim['McMax [kNm]'],facecolor='none',edgecolor='orang
alpha= 0.8, marker='s', label='True WIM')
plt.xlabel('GVW [t]');plt.ylabel('Bending moment [kNm]');plt.title('Bending moment v
plt.xlim(70,110);plt.ylim(600,1350)
plt.legend(loc='upper right')
leg = plt.legend(loc='lower right')
for lh in leg.legendHandles:
    lh.set_alpha(1)
plt.show()
```

```
In [ ]: df[(df['Axles'] == '3') & (df['Wheelbase N3 [m]'] > '6')]
```

```
In [ ]: aa = df_wim[(df_wim['Axles'] == 5) & (df_wim['DT12'] < 300) & (df_wim['DT23'] < 300
& (df_wim['DT34'] < 300) & (df_wim['DT45'] < 300) & (df_wim['LENGTH'] >
plt.hist(aa['AXW1']/120,bins=50,ec="k");
plt.vlines(1,0,600,color='red')
plt.xlabel('\eta_{1}');plt.ylabel('Frequency');plt.title('AX1 (WIM)');plt.xlim(0,3
plt.show()
plt.vlines(1,0,600,color='red')
plt.hist(aa['AXW2']/120,bins=50,ec="k");
plt.xlabel('\eta_{2}');plt.ylabel('Frequency');plt.title('AX2 (WIM)');plt.xlim(0,3
plt.show()
plt.vlines(1,0,200,color='red')
plt.hist(aa['AXW3']/120,bins=50,ec="k");
plt.xlabel('\eta_{3}');plt.ylabel('Frequency');plt.title('AX3 (WIM)');plt.xlim(0,3
plt.show()
plt.vlines(1,0,200,color='red')
plt.hist(aa['AXW4']/120,bins=50,ec="k");
plt.xlabel('\eta_{4}');plt.ylabel('Frequency');plt.title('AX4 (WIM)');plt.xlim(0,3
plt.show()
plt.vlines(1,0,200,color='red')
plt.hist(aa['AXW5']/120,bins=50,ec="k");
plt.xlabel('\eta_{5}');plt.ylabel('Frequency');plt.title('AX5 (WIM)');plt.xlim(0,3
plt.show()
```

## File 2 - Monte Carlo simulation.

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

Reading databases. Drop LM1 value.

```
In [ ]: df = pd.read_csv('df_4400C_load.csv', low_memory=False)
df2 = pd.read_csv('WIM.csv', low_memory=False)
df.drop(df.tail(1).index, inplace=True)
```

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = df['McMax [kNm]'].to_numpy()
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)

dmax[::-1].sort()

occdmax= np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))

#Berekenen parameters GumbelR
parameters = scstats.gumbel_r.fit(dmax)
mu_d = parameters[0]
beta_d = parameters[1]
##FUNCTIONS##
def gumbel(x, mu, beta):
    z = (x - mu) / beta
    return (1/beta) * np.exp(-(z + np.exp(-z)))
def invgumbelpdf(x, mu, beta):
    return (1 - (np.exp(-np.exp(-(x - mu) / beta))))
x = np.linspace(0, 10e3, 100001)

daily = gumbel(x, mu_d, beta_d)
maanden = 12*15 #15 jaar simuleren
mmax = np.zeros(maanden)
for k in range(maanden): #12 maanden --> 1 yr
    mtemp = np.zeros(30*150)
    for j in range(30*150): #30 dagen / maand, 150 voertuigen per dag
        d = np.random.gumbel(loc=mu_d, scale=beta_d)
        mtemp[j] = d
    mmax[k] = np.max(mtemp)

#Berekenen parameters GumbelR
parameters = scstats.gumbel_r.fit(mmax)
mu_m = parameters[0]
beta_m = parameters[1]

##PLOT##
```

```

mmax[:, :-1].sort()
occmmmax= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmmax[r] = (r/(len(mmax)+1))
monthly = gumbel(x,mu_m,beta_m)

#yearly maxima wordt berekend door dist van monthly maxima te simuleren voor 1000 ja
yearly = gumbel(x,mu_m,beta_m)
years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12):#1 jaar heeft 12 maanden
        d = np.random.gumbel(loc=mu_m,scale=beta_m)
        ytemp[j] = d
    yrmax[k] = np.max(d)

#Berekenen parameters GumbelR
parameters = scstats.gumbel_r.fit(yrmax)
mu_yr = parameters[0]
beta_yr = parameters[1]
##PLOT##
yrmax[:, :-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = (r/(len(yrmax)+1))
yearly = gumbel(x,mu_yr,beta_yr)

```

Vanaf hieronder komt de resistance van een concrete beam

```

In [ ]: ##FUNCTIONS##
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def normal(x,mu,sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x,v):
    return scstats.t.pdf(x,v)
def lognorm_sample(mu,sigma):
    return scstats.lognorm.rvs(s=sigma,scale=np.exp(mu))
def gumbel_sample(mu,beta):
    return scstats.gumbel_r.rvs(loc=mu,scale=beta)
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu,sigma):
    return scstats.norm.rvs(loc=mu,scale=sigma)
def weibull(x,shape,loc,scale):
    return scstats.weibull_min.pdf(x,c=shape,loc=loc,scale=scale)

###JCSS CONC###
x = np.linspace(0,100e3,1000000)
m = 3.65;v=10;s=0.12;n=3
mu_y1 = 1; sigma_y1=0.06
e=100000
fc_arr = np.zeros(e);ft_arr=np.zeros(e);ec_arr = np.zeros(e);eps_arr = np.zeros(e);
for r in range(e):
    fc_arr[r] = (np.exp(m+student_t_sample(v)*s*(1+(1/n))**0.5))
    ft_arr[r] = 0.3*(fc_arr[r]**(2/3))
    ec_arr[r] = 10.5*(fc_arr[r]**(1/3))/(1+0.7*2.5)
    eps_arr[r] = 6e-3 * (fc_arr[r]**(-1/6))*(1+0.7*2.5)

##Fitting dist##
fcp = scstats.lognorm.fit(fc_arr,loc=0,scale=1);

```

```

ftp = scstats.lognorm.fit(ft_arr,loc=0,scale=1);
ecp = scstats.lognorm.fit(ec_arr,loc=0,scale=1);
eps = scstats.lognorm.fit(eps_arr);

##JCSS Reinforcing steel + Moment resistance##
def F_steel(Sxx,n,d): #F_steel in kN
    X12 = normal_sample(0,22)
    X13 = normal_sample(0,8)
    mu1 = Sxx + 2*30
    mu11 = mu1 * ((0.87 + 0.13*np.exp(-0.08*d))**-1)
    X11 = normal_sample(mu11,19)
    fyd = X11 + X12 + X13

    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)

    return fyd * As_sample / 1e3 , fyd

def F_conc(Sxx,n,d): #F_conc in kN
    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)
    xc = (As_sample)*F_steel(Sxx,n,d)[1] / (0.85*lognorm_sample(np.log(fcp[2]),(fcp[
    Fcd = 0.85*xc * lognorm_sample(np.log(fcp[2]),fcp[0])
    return Fcd/1e3

def M_res(Sxx,n,d,z): #M_res in kNm
    fs = F_steel(Sxx,n,d)[0]
    fc = F_conc(Sxx,n,d)
    focc = np.minimum(fs,fc)
    mres = focc*z
    return mres/1e3

```

```
In [ ]: print(np.mean(fc_arr),np.mean(ft_arr),np.mean(ec_arr))
        F_steel(435,1,25)
```

Part below is the MC simulation.

```
In [ ]: def MCsim(Sxx,n,d,z,e):
        print('Running a Monte-Carlo simulation for',e,'times.')
        loadarr = np.zeros(e)
        resarr = np.zeros(e)
        zarr = np.zeros(e)
        zarr2 = np.zeros(e)

        with tqdm(total=e, file=sys.stdout) as pbar:
            for r in range(e):
                pbar.set_description('MC-simulation')
                pbar.update(1)
                loadarr[r] = gumbel_sample(mu_yr,beta_yr)
                resarr[r] = M_res(Sxx,n,d,z)
                zarr[r] = loadarr[r]/resarr[r]
                zarr2[r] = resarr[r] - loadarr[r]

        ##ICDF plot##
        zarr[::-1].sort()
        occzarr= np.zeros_like(zarr)
        for r in range(len(zarr)):
            occzarr[r] = (r/(len(zarr)+1))
        ##Fitting ICDF##
        params = scstats.gumbel_r.fit(zarr,loc=0,scale=1); params2 = scstats.weibull_min
        ##PLOTS##
        plt.figure(figsize=(5,5))
        plt.scatter(resarr,loadarr,facecolors='none',color='green',marker='o',s=1);plt.p

```

```

plt.xlim(0,4000); plt.ylim(0,4000); plt.xlabel('Resistance [kNm]');plt.ylabel

plt.figure(figsize=(5,5))
plt.hist(loadarr,bins=50,alpha=0.5,label='4400C Load',density=True) ; plt.hist
plt.legend(loc='upper right');plt.xlabel('Bending moment [kNm]');plt.ylabel('Pro
plt.show()

plt.figure(figsize=(5,5))
plt.scatter(zarr,occzarr,color='green',facecolors='none',s=50,label='Data');plt.

plt.plot(x,(1-scstats.gumbel_r.cdf(x,params[0],params[1])),label='GUMB fit');
plt.title('ICDF MC-sim');plt.xlabel('Load / Resistance');plt.ylabel('Exceedance
plt.show()

#calculate prob of failure
x_pof = zarr; y_pof = occzarr;
l = len(x_pof[x_pof > 1]) ; ecdf = ECDF(zarr);

plt.plot(x,scstats.gumbel_r.cdf(x,params[0],params[1]),label='Fitted CDF');
plt.plot(ecdf.x,ecdf.y,label='True CDF'); plt.ylabel('P(x)');plt.xlabel('R(x)
print("")
pof = y_pof[l]
beta = (1-np.mean(zarr))/(scstats.gumbel_r.std(loc=params[0],scale=params[1]))
print('The probability of failure is',"{: .2E}".format(Decimal(pof)));
print('The reliability index is', beta) #CIE4130

return (loadarr,resarr,zarr,occzarr,zarr2,params[0],params[1],pof)

```

```
In [ ]: f= MCsim(500,17,32,500,1000000) #testing purposes
```

```
In [ ]: plt.figure(figsize=(5,5))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca

plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='GUMB fit');
plt.title('ICDF MC-sim');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceedance freq
plt.show()

```

## File 3 - Curve fitting

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

```
In [ ]: df = pd.read_csv('df_4400C_load.csv', low_memory=False)
df.drop(df.tail(1).index, inplace=True)

x = np.linspace(0, 100e3, 100e3)
dist_names = [ 'alpha', 'anglit', 'arcsine', 'beta', 'betaprime', 'bradford', 'burr'
```

```
In [ ]: ##FUNCTIONS##
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def normal(x, mu, sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x, v):
    return scstats.t.pdf(x, v)
def lognorm_sample(mu, sigma):
    return scstats.lognorm.rvs(s=sigma, scale=np.exp(mu))
def gumbel_sample(mu, beta):
    return scstats.gumbel_r.rvs(loc=mu, scale=beta)
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu, sigma):
    return scstats.norm.rvs(loc=mu, scale=sigma)
def weibull(x, shape, loc, scale):
    return scstats.weibull_min.pdf(x, c=shape, loc=loc, scale=scale)
def gumbel(x, mu, beta):
    z = (x - mu)/beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x, mu, beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))
```

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = df['McMax [kNm]'].to_numpy()
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)

dmax[::-1].sort()

occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))
```

```
In [ ]: dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
```

```

for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(dmax,loc=0,scale=1)
c_d = parameters[0]; loc_d = parameters[1]; scale_d = parameters[2]
daily_weib = weibull(x,c_d,loc_d,scale_d)

parameters = scstats.gumbel_r.fit(dmax)
mu_d = parameters[0]
beta_d = parameters[1]
daily_gumb = gumbel(x,mu_d,beta_d)

parameters = scstats.genextreme.fit(dmax)
c_dg = parameters[0]; loc_dg = parameters[1]; scale_dg = parameters[2]
daily_gev = scstats.genextreme.pdf(x,c_dg,loc_dg,scale_dg)

parameters = scstats.norm.fit(dmax)
loc_dn = parameters[0]; scale_dn = parameters[1]
daily_norm = scstats.norm.pdf(x,loc_dn,scale_dn)

parameters = scstats.beta.fit(dmax)
a_d,b_d,loc_db,scale_db = parameters
daily_beta = scstats.beta.pdf(x,a_d,b_d,loc_db,scale_db)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(dmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

```

```

In [ ]: if len(params) == 2:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
    if len(params) == 3:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
    if len(params) == 4:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],size=1)
    if len(params) == 5:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params[4])
    if len(params) == 6:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params[4],params[5])

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
        plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')

        if len(params) == 2:
            plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
        if len(params) == 3:
            plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
        if len(params) == 4:
            plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
        if len(params) == 5:
            plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))

```



```

plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
if len(params) == 6:
    plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))

plt.plot(x,daily_weib,label='Weibull fit',color='blue');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black');plt.xlabel('Bending mo
plt.xlim(650,1400);plt.ylim(0,0.008);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+str
if len(params) == 3:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit
if len(params) == 4:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3]),color='Red',labe
if len(params) == 5:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4]),color=
if len(params) == 6:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(1100,1200);plt.ylim(0,0.0008);

plt.figure(figsize=(12,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.yasca
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit -'+str
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit -'+str(
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
if len(params) == 5:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange'
plt.plot(1-scstats.norm.cdf(x,loc_dn,scale_dn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_d,b_d,loc_db,scale_db),label='Beta fit',linestyle=':
plt.title('ICDF Daily maxima');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceedanc
plt.show()

```

In [ ]: params

-----MONTHLY-----

Please note: daily\_bestfit\_rvs does not properly work, so copy the function and let the code run.

In [ ]: maanden = 12\*25 #15 jaar simuleren  
mmax = np.zeros(maanden)

```

for k in range(maanden): #12 maanden --> 1 yr
    mtemp = np.zeros(30*150)
    for j in range(30*150): #30 dagen / maand , 150 voertuigen per dag
        d = bestfit.rvs(params[0],params[1],params[2],size=1)
        mtemp[j] = d
    mmax[k] = np.max(mtemp)

```

```

In [ ]: ###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(mmax,loc=0,scale=1)
c_m = parameters[0]; loc_m = parameters[1]; scale_m = parameters[2]
monthly_weib = weibull(x,c_m,loc_m,scale_m)

parameters = scstats.gumbel_r.fit(mmax)
mu_m = parameters[0]
beta_m = parameters[1]
monthly_gumb = gumbel(x,mu_m,beta_m)

parameters = scstats.genextreme.fit(mmax)
c_mg = parameters[0]; loc_mg = parameters[1]; scale_mg = parameters[2]
monthly_gev = scstats.genextreme.pdf(x,c_mg,loc_mg,scale_mg)

parameters = scstats.norm.fit(mmax)
loc_mn = parameters[0]; scale_mn = parameters[1]
monthly_norm = scstats.norm.pdf(x,loc_mn,scale_mn)

parameters = scstats.beta.fit(mmax)
a_m,b_m,loc_mb,scale_mb = parameters
monthly_beta = scstats.beta.pdf(x,a_m,b_m,loc_mb,scale_mb)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

##PLOT##
mmax[:, :-1].sort()
occmmmax= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmmax[r] = (r/(len(mmax)+1))

```

```

In [ ]: if len(params) == 2:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
if len(params) == 3:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
if len(params) == 4:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],size=1)
if len(params) == 5:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params
if len(params) == 6:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
if len(params) == 2:
    plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 3:
    plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 4:
    plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 5:
    plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 6:
    plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')
plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(1150,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str
if len(params) == 3:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit
if len(params) == 4:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3]),color='Red',labe
if len(params) == 5:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4]),color=
if len(params) == 6:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')
plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
plt.title('Monthly maxima distribution');
plt.xlim(1350,1450);plt.ylim(0,0.003);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6))
plt.scatter(mmax,ocmmmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+s
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str(
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
if len(params) == 5:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param

plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange'
plt.plot(1-scstats.norm.cdf(x,loc_mn,scale_mn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_m,b_m,loc_mb,scale_mb),label='Beta fit',linestyle=':

```

```
plt.title('ICDF Monthly maxima');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceeda');plt.show()
```

-----YEARLY-----

```
In [ ]: mu_m,beta_m
```

```
In [ ]: years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12):#1 jaar heeft 12 maanden
        d = np.random.gumbel(loc=mu_m,scale=beta_m)
        ytemp[j] = d
    yrmax[k] = np.max(d)
```

```
In [ ]: ##PLOT##
yrmax[:,::-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = (r/(len(yrmax)+1))

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(yrmax,loc=0,scale=1)
c_yr = parameters[0]; loc_yr = parameters[1]; scale_yr = parameters[2]
yearly_weib = weibull(x,c_yr,loc_yr,scale_yr)

parameters = scstats.gumbel_r.fit(yrmax)
mu_yr = parameters[0]
beta_yr = parameters[1]
yearly_gumb = gumbel(x,mu_yr,beta_yr)

parameters = scstats.genextreme.fit(yrmax)
c_yrg = parameters[0]; loc_yrg = parameters[1]; scale_yrg = parameters[2]
yearly_gev = scstats.genextreme.pdf(x,c_yrg,loc_yrg,scale_yrg)

parameters = scstats.norm.fit(yrmax)
loc_yrn = parameters[0]; scale_yrn = parameters[1]
yearly_norm = scstats.norm.pdf(x,loc_yrn,scale_yrn)

parameters = scstats.beta.fit(yrmax)
a_yr,b_yr,loc_yrb,scale_yrb = parameters
yearly_beta = scstats.beta.pdf(x,a_yr,b_yr,loc_yrb,scale_yrb)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(yrmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))
```

```
In [ ]: if len(params) == 2:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
if len(params) == 3:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
if len(params) == 4:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
```

```

yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],size=1)
if len(params) == 5:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
yearly_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params[
if len(params) == 6:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params[

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
if len(params) == 2:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 3:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 4:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 5:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 6:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(1150,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str
if len(params) == 3:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit
if len(params) == 4:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3]),color='Red',labe
if len(params) == 5:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4]),color=
if len(params) == 6:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');plt.hist(yrmax,density=True
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.title('Yearly maxima distribution');
plt.xlim(1350,1450);plt.ylim(0,0.003);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Data');plt.ys
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+s
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str(
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
if len(params) == 5:

```



```

plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),color='green',label='Lognormal fit',color='green',lw=2)
plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),color='blue',label='Weibull fit',color='blue',lw=2)
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5]),color='orange',label='Gumbel fit',color='orange',lw=2)
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5]),color='red',label='Beta fit',color='red',lw=2)

plt.plot(x,(1-scstats.weibull_min.cdf(x,c_yr,loc_yr,scale_yr)),label='Weibull fit',color='blue',lw=2)
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_yr,beta_yr)),label='Gumbel fit',color='orange',lw=2)
plt.plot(1-scstats.norm.cdf(x,loc_yrn,scale_yrn),label='Normal fit',linestyle='--',color='green',lw=2)
plt.plot(1-scstats.beta.cdf(x,a_yr,b_yr,loc_yrb,scale_yrb),label='Beta fit',linestyle='--',color='red',lw=2)

plt.title('ICDF Yearly maxima');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceedance probability');plt.show()

```

```

In [ ]: plt.figure(figsize=(10,10));plt.title('Daily, monthly, yearly maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Daily');
plt.plot(daily_bestfit_pdf,color='green',label='Daily fit',lw=2)
plt.hist(mmax,density=True,bins=25,color='orange',alpha=0.3,ec="k",label='Monthly');
plt.plot(monthly_gumb,color='orange',label='Monthly fit',lw=2)
plt.hist(yrmax,density=True,bins=25,color='blue',alpha=0.3,ec="k",label='Yearly');
plt.plot(yearly_bestfit_pdf,color='blue',label='Yearly fit',lw=2,linestyle='--')

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');

plt.xlim(600,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

```

-----PLOT FOR REPORT-----

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');plt.xlabel('Bending moment [kNm]');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Daily maxima');
plt.plot(daily_bestfit_pdf,color='green',label='Lognormal');
plt.plot(x,daily_weib,label='Weibull fit',color='blue');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit');
plt.xlim(650,1400);plt.ylim(0,0.008);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='green',label='Lognormal');
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Daily maxima');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit');
plt.title('Daily maxima distribution');plt.legend(loc='upper right');
plt.xlim(1100,1200);plt.ylim(0,0.0008);

plt.figure(figsize=(12,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ylabel('Exceedance probability');
plt.plot(1-daily_bestfit_cdf,color='green',label='Lognormal');
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',color='blue',lw=2);
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='orange',lw=2);

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceedance probability');plt.show()

```

\_\_MONTHLY\_\_

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');plt.xlabel('Bending moment [kNm]');
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='monthly maxima');
plt.plot(monthly_bestfit_pdf,color='green',label='Mielke fit');
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit');
plt.xlim(1100,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

```

```

plt.figure(figsize=(8,6));
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf,color='Green',label='Mielke fit')
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True)
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.title('Monthly maxima distribution');plt.legend(loc='upper right')
plt.xlim(1300,1500);plt.ylim(0,0.003);

plt.figure(figsize=(12,6))
plt.scatter(mmax,occmmax,color='green',facecolors='none',s=50,label='Monthly maxima')
plt.plot(1-monthly_bestfit_cdf,color='Green',label='Mielke fit')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',color='blue')
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange')

plt.title('ICDF Monthly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceedance')
plt.show()

```

-----YEARLY-----

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');plt.xlabel('Bending moment [kNm]');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='yearly maxima')
plt.plot(yearly_bestfit_pdf,color='Green',label='Lognorm fit')
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.xlim(1100,1650);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(yearly_bestfit_pdf,color='Green',label='Lognorm fit')
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');plt.hist(yrmax,density=True)
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.title('Yearly maxima distribution');plt.legend(loc='upper right')
plt.xlim(1350,1650);plt.ylim(0,0.003);

plt.figure(figsize=(12,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Yearly maxima')
plt.ylim(1e-6,2);plt.xlim(1150,1700)
plt.plot(1-yearly_bestfit_cdf,color='Green',label='Lognorm fit')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_yr,loc_yr,scale_yr)),label='Weibull fit',color='blue')
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_yr,beta_yr)),label='Gumbel fit',color='Orange')

plt.title('ICDF Yearly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceedance')
plt.show()

```

```

In [ ]: df = pd.DataFrame(yrmax)
df.to_excel('YRMAX_FINAL.xlsx')

```

```

In [ ]:

```

# Curve fitting for LP data

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

```
In [ ]: df = pd.read_csv('df_4400C_load_LP.csv', low_memory=False)
df.drop(df.tail(1).index, inplace=True)
x = np.linspace(0, 100e3, 100e3)
dist_names = [ 'alpha', 'anglit', 'arcsine', 'beta', 'betaprime', 'bradford', 'burr'
```

```
In [ ]: ##FUNCTIONS##
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def normal(x, mu, sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x, v):
    return scstats.t.pdf(x, v)
def lognorm_sample(mu, sigma):
    return scstats.lognorm.rvs(s=sigma, scale=np.exp(mu))
def gumbel_sample(mu, beta):
    return scstats.gumbel_r.rvs(loc=mu, scale=beta)
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu, sigma):
    return scstats.norm.rvs(loc=mu, scale=sigma)
def weibull(x, shape, loc, scale):
    return scstats.weibull_min.pdf(x, c=shape, loc=loc, scale=scale)
def gumbel(x, mu, beta):
    z = (x - mu) / beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x, mu, beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))
```

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = df['McMax [kNm]'].to_numpy()
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)
dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))
```

```
In [ ]: plt.hist(dmax)
```

```
In [ ]: dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
```



```

occdmax[r] = (r/(len(dmax)+1))

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(dmax,loc=0,scale=1)
c_d = parameters[0]; loc_d = parameters[1]; scale_d = parameters[2]
daily_weib = weibull(x,c_d,loc_d,scale_d)

parameters = scstats.gumbel_r.fit(dmax)
mu_d = parameters[0]
beta_d = parameters[1]
daily_gumb = gumbel(x,mu_d,beta_d)

parameters = scstats.genextreme.fit(dmax)
c_dg = parameters[0]; loc_dg = parameters[1]; scale_dg = parameters[2]
daily_gev = scstats.genextreme.pdf(x,c_dg,loc_dg,scale_dg)

parameters = scstats.norm.fit(dmax)
loc_dn = parameters[0]; scale_dn = parameters[1]
daily_norm = scstats.norm.pdf(x,loc_dn,scale_dn)

parameters = scstats.beta.fit(dmax)
a_d,b_d,loc_db,scale_db = parameters
daily_beta = scstats.beta.pdf(x,a_d,b_d,loc_db,scale_db)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(dmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

```

```

In [ ]: if len(params) == 2:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
    if len(params) == 3:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
    if len(params) == 4:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_rvs = bestfit.rvs(x,params[0],params[1],params[2],params[3],size=1)
    if len(params) == 5:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params[4])
    if len(params) == 6:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params[4],params[5])
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency')
plt.xlim(650,1100);plt.ylim(0,0.05);plt.legend(loc='upper right');plt.show()

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlabel('Bending moment [kNm]');plt.ylabel('Frequency')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');

```

```

plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.xlim(650,1100);plt.ylim(0,0.05);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(800,1100);plt.ylim(0,0.0018);plt.show()

plt.figure(figsize=(8,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
#plt.plot(1-daily_bestfit_cdf,color='Red',label='Bestfit -'+str(name))
plt.plot(bestfit.sf(x,params[0],params[1],params[2]),color='Red',label='Bestfit -'+
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange'
plt.plot(1-scstats.norm.cdf(x,loc_dn,scale_dn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_d,b_d,loc_db,scale_db),label='Beta fit',linestyle=':

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceed
plt.show()

```

-----MONTHLY-----

Please note: daily\_bestfit\_rvs does not properly work, so copy the function and let the code run.

```

In [ ]: ###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(dmax,loc=0,scale=1)
c_d = parameters[0]; loc_d = parameters[1]; scale_d = parameters[2]
daily_weib = weibull(x,c_d,loc_d,scale_d)

daily = daily_weib
maanden = 12*25 #15 jaar simuleren
mmax = np.zeros(maanden)
for k in range(maanden): #12 maanden --> 1 yr
    mtemp = np.zeros(30*150)
    for j in range(30*150): #30 dagen / maand , 150 voertuigen per dag
        d = scstats.weibull_min.rvs(c=c_d,loc=loc_d,scale=scale_d,size=1)
        mtemp[j] = d
    mmax[k] = np.max(mtemp)

```

```

In [ ]: ###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(mmax,loc=0,scale=1)
c_m = parameters[0]; loc_m = parameters[1]; scale_m = parameters[2]
monthly_weib = weibull(x,c_m,loc_m,scale_m)

parameters = scstats.gumbel_r.fit(mmax)
mu_m = parameters[0]
beta_m = parameters[1]
monthly_gumb = gumbel(x,mu_m,beta_m)

parameters = scstats.genextreme.fit(mmax)
c_mg = parameters[0]; loc_mg = parameters[1]; scale_mg = parameters[2]
monthly_gev = scstats.genextreme.pdf(x,c_mg,loc_mg,scale_mg)

parameters = scstats.norm.fit(mmax)
loc_mn = parameters[0]; scale_mn = parameters[1]

```

```

monthly_norm = scstats.norm.pdf(x,loc_mn,scale_mn)

parameters = scstats.beta.fit(mmax)
a_m,b_m,loc_mb,scale_mb = parameters
monthly_beta = scstats.beta.pdf(x,a_m,b_m,loc_mb,scale_mb)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

##PLOT##
mmax[:, :-1].sort()
occmmx= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmx[r] = ((r+1)/(len(mmax)+1))

```

```

In [ ]: if len(params) == 2:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
if len(params) == 3:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
if len(params) == 4:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    monthly_bestfit_rvs = bestfit.rvs(x,params[0],params[1],params[2],params[3],size
if len(params) == 5:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params
if len(params) == 6:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')
plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(800,1400);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')
plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
plt.title('Monthly maxima distribution');
plt.xlim(1200,1450);plt.ylim(0,0.0015);plt.legend(loc='upper right');plt.show()

```

```

plt.figure(figsize=(8,6))
plt.scatter(mmax,occmmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit - '+'"+s
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit - '+'"+str(
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
if len(params) == 5:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param

plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange'
plt.plot(1-scstats.norm.cdf(x,loc_mn,scale_mn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_m,b_m,loc_mb,scale_mb),label='Beta fit',linestyle=':

plt.title('ICDF Monthly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exce
plt.show()

```

-----YEARLY-----

```

In [ ]: yearly = gumbel(x,mu_m,beta_m)
years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12):#1 jaar heeft 12 maanden
        d = np.random.gumbel(loc=mu_m,scale=beta_m)
        ytemp[j] = d
    yrmax[k] = np.max(d)

```

```

In [ ]: ##PLOT##
yrmax[::-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = ((r+1)/(len(yrmax)+1))

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(yrmax,loc=0,scale=1)
c_yr = parameters[0]; loc_yr = parameters[1]; scale_yr = parameters[2]
yearly_weib = weibull(x,c_yr,loc_yr,scale_yr)

parameters = scstats.gumbel_r.fit(yrmax)
mu_yr = parameters[0]
beta_yr = parameters[1]
yearly_gumb = gumbel(x,mu_yr,beta_yr)

parameters = scstats.genextreme.fit(yrmax)
c_yrg = parameters[0]; loc_yrg = parameters[1]; scale_yrg = parameters[2]
yearly_gev = scstats.genextreme.pdf(x,c_yrg,loc_yrg,scale_yrg)

parameters = scstats.norm.fit(yrmax)
loc_yrn = parameters[0]; scale_yrn = parameters[1]
yearly_norm = scstats.norm.pdf(x,loc_yrn,scale_yrn)

parameters = scstats.beta.fit(yrmax)

```

```
a_yr,b_yr,loc_yrb,scale_yrb = parameters
yearly_beta = scstats.beta.pdf(x,a_yr,b_yr,loc_yrb,scale_yrb)
```

```
from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(yrmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))
```

```
In [ ]: if len(params) == 2:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
    if len(params) == 3:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
    if len(params) == 4:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params[
    if len(params) == 6:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params[
```

```
In [ ]: print(mu_m,beta_m)
```

```
In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
if len(params) == 2:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 3:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 4:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 5:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 6:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
#plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(800,1400);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str
if len(params) == 3:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit
if len(params) == 4:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3]),color='Red',labe
if len(params) == 5:
```



```

plt.plot(bestfit.pdf(x, params[0], params[1], params[2], params[3], params[4]), color=
if len(params) == 6:
plt.plot(bestfit.pdf(x, params[0], params[1], params[2], params[3], params[4], params[
plt.plot(x, yearly_weib, label='Weibull fit', color='blue'); plt.hist(yrmax, density=True
plt.plot(x, yearly_gumb, color='orange', label='Gumbel fit')
plt.plot(yearly_norm, label='Normal fit', linestyle='--')
plt.plot(yearly_beta, label='Beta', linestyle=':', color='black')
plt.title('Yearly maxima distribution');
plt.xlim(1200, 1400); plt.ylim(0, 0.003); plt.legend(loc='upper right'); plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(yrmax, occyrmax, color='green', facecolors='none', s=50, label='Data'); plt.ys
if len(params) == 2:
plt.plot(1-bestfit.cdf(x, params[0], params[1]), color='Red', label='Bestfit - '+''+s
plt.plot(bestfit.sf(x, params[0], params[1]), color='Red', label='Bestfit - '+''+str(
if len(params) == 3:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2]), color='Red', label='Bestf
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2]), color='Red', label='Bestf
if len(params) == 4:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3]), color='Red', la
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3]), color='Red', la
if len(params) == 5:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4]), colo
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4]), colo
if len(params) == 6:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4], param
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4], param

plt.plot(x, (1-scstats.weibull_min.cdf(x, c_yr, loc_yr, scale_yr)), label='Weibull fit', c
plt.plot(x, (1-scstats.gumbel_r.cdf(x, mu_yr, beta_yr)), label='Gumbel fit', color='Orang
plt.plot(1-scstats.norm.cdf(x, loc_yrn, scale_yrn), label='Normal fit', linestyle='--')
plt.plot(1-scstats.beta.cdf(x, a_yr, b_yr, loc_yrb, scale_yrb), label='Beta fit', linestyl

plt.title('ICDF Yearly maxima'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Excee
plt.show()

```

```
In [ ]: mu_yr, beta_yr
```

```

In [ ]: plt.figure(figsize=(10, 10)); plt.title('Daily, monthly, yearly maxima distribution');
plt.hist(dmax, density=True, bins=25, color='green', alpha=0.3, ec="k", label='Daily');
plt.plot(daily_bestfit_pdf, color='green', label='Daily fit', lw=2)
plt.hist(mmax, density=True, bins=25, color='orange', alpha=0.3, ec="k", label='Monthly');
plt.plot(monthly_gumb, color='orange', label='Monthly fit', lw=2)
plt.hist(yrmax, density=True, bins=25, color='blue', alpha=0.3, ec="k", label='Yearly');
plt.plot(yearly_bestfit_pdf, color='blue', label='Yearly fit', lw=2, linestyle='--')

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');

plt.xlim(600, 1500); plt.ylim(0, 0.015); plt.legend(loc='upper right'); plt.show()

```

```
In [ ]: ddfd = pd.DataFrame(yrmax)
ddfd.to_csv('YRMAX_FINAL.csv')
```

-----PLOT FOR REPORT-----

```

In [ ]: plt.figure(figsize=(8, 6)); plt.title('Daily maxima distribution'); plt.xlabel('Bending
plt.hist(dmax, density=True, bins=25, color='green', alpha=0.3, ec="k", label='Daily maxim
plt.plot(daily_bestfit_pdf, color='Green', label='Double Weibull')
plt.plot(x, daily_weib, label='Weibull fit', color='blue');
plt.plot(x, daily_gumb, color='orange', label='Gumbel fit')
plt.xlim(650, 1400); plt.ylim(0, 0.008); plt.legend(loc='upper right'); plt.show()

```

```
plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='Green',label='Double Weibull')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(1100,1200);plt.ylim(0,0.0008);

plt.figure(figsize=(12,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
plt.plot(1-daily_bestfit_cdf,color='Green',label='Double Weibull')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange'

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceed
plt.show()
```

### \_d\_\_MONTHLY\_\_

In [ ]:

```
plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');plt.xlabel('Bendi
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='monthly max
plt.plot(monthly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.xlim(1150,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.title('Monthly maxima distribution');plt.legend(loc='upper right')
plt.xlim(1350,1500);plt.ylim(0,0.003);

plt.figure(figsize=(12,6))
plt.scatter(mmax,occmmax,color='green',facecolors='none',s=50,label='Monthly maxima'
plt.plot(1-monthly_bestfit_cdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange'

plt.title('ICDF Monthly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exce
plt.show()
```

### -----YEARLY-----

In [ ]:

```
plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');plt.xlabel('Bendin
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='yearly max
plt.plot(yearly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.xlim(1150,1650);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(yearly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');plt.hist(yrmax,density=True
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.title('Yearly maxima distribution');plt.legend(loc='upper right')
plt.xlim(1350,1650);plt.ylim(0,0.003);

plt.figure(figsize=(12,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Yearly maxima
```



```
plt.ylim(1e-6,2);plt.xlim(1150,1700)
plt.plot(1-yearly_bestfit_cdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_yr,loc_yr,scale_yr)),label='Weibull fit',c
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_yr,beta_yr)),label='Gumbel fit',color='Orang

plt.title('ICDF Yearly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Excee
plt.show()
```

# Case study span length 15 m curve fitting + MC sim

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
from distfit import distfit
```

```
In [ ]: df = pd.read_csv('df_4400C_load_L1_L20.csv', low_memory=False, index_col=0)
x = np.linspace(0, 100e3, 100e3)
dist_names = [ 'alpha', 'anglit', 'arcsine', 'beta', 'betaprime', 'bradford', 'burr'
```

```
In [ ]: df.drop(df.tail(1).index, inplace=True)
L5 = df['L5'].to_numpy()
L15 = df['L15'].to_numpy()
L20 = df['L20'].to_numpy()
```

-----|-----Daily maxima-----|-----

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = L15
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)
dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(dmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));
###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(dmax, loc=0, scale=1)
c_d = parameters[0]; loc_d = parameters[1]; scale_d = parameters[2]
daily_weib = scstats.weibull_min.pdf(x, c_d, loc_d, scale_d)

parameters = scstats.gumbel_r.fit(dmax)
mu_d = parameters[0]
beta_d = parameters[1]
daily_gumb = scstats.gumbel_r.pdf(x, mu_d, beta_d)

parameters = scstats.genextreme.fit(dmax)
c_dg = parameters[0]; loc_dg = parameters[1]; scale_dg = parameters[2]
daily_gev = scstats.genextreme.pdf(x, c_dg, loc_dg, scale_dg)

parameters = scstats.norm.fit(dmax)
loc_dn = parameters[0]; scale_dn = parameters[1]
daily_norm = scstats.norm.pdf(x, loc_dn, scale_dn)
```

```

parameters = scstats.beta.fit(dmax)
a_d,b_d,loc_db,scale_db = parameters
daily_beta = scstats.beta.pdf(x,a_d,b_d,loc_db,scale_db)

```

```

In [ ]: if len(params) == 2:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        if len(params) == 3:
            daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
            daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        if len(params) == 4:
            daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
            daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
        if len(params) == 5:
            daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4])
            daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4])
        if len(params) == 6:
            daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
            daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlabel('Bending moment [kNm]');plt.ylabel('Frequency')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.xlim(1000,2500);plt.ylim(0,0.0040);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(1900,2200);plt.ylim(0,0.002);plt.show()

plt.figure(figsize=(8,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
plt.xlim(1200,2500)
#plt.plot(1-daily_bestfit_cdf,color='Red',label='Bestfit -'+str(name))
plt.plot(bestfit.sf(x,params[0],params[1],params[2]),color='Red',label='Bestfit -'+
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange')
plt.plot(x,(1-scstats.norm.cdf(x,loc_dn,scale_dn)),label='Normal fit',linestyle='--')
plt.plot(x,(1-scstats.beta.cdf(x,a_d,b_d,loc_db,scale_db)),label='Beta fit',linestyle=':

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceed
plt.show()

```

-----|-----Monthly maxima-----|-----

```

In [ ]: maanden = 12*25 #25 jaar simuleren
mmax = np.zeros(maanden)
for k in range(maanden): #12 maanden --> 1 yr
    mtemp = np.zeros(30*150)

```

```

for j in range(30*150): #30 dagen / maand , 150 voertuigen per dag
    d = scstats.norm.rvs(loc_dn,scale_dn,size=1)
    mtemp[j] = d
mmax[k] = np.max(mtemp)

dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(mmax,loc=0,scale=1)
c_m = parameters[0]; loc_m = parameters[1]; scale_m = parameters[2]
monthly_weib = scstats.weibull_min.pdf(x,c_m,loc_m,scale_m)

parameters = scstats.gumbel_r.fit(mmax)
mu_m = parameters[0]
beta_m = parameters[1]
monthly_gumb = scstats.gumbel_r.pdf(x,mu_m,beta_m)

parameters = scstats.genextreme.fit(mmax)
c_mg = parameters[0]; loc_mg = parameters[1]; scale_mg = parameters[2]
monthly_gev = scstats.genextreme.pdf(x,c_mg,loc_mg,scale_mg)

parameters = scstats.norm.fit(mmax)
loc_mn = parameters[0]; scale_mn = parameters[1]
monthly_norm = scstats.norm.pdf(x,loc_mn,scale_mn)

parameters = scstats.beta.fit(mmax)
a_m,b_m,loc_mb,scale_mb = parameters
monthly_beta = scstats.beta.pdf(x,a_m,b_m,loc_mb,scale_mb)

##PLOT##
mmax[:, :-1].sort()
occmmax= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmax[r] = ((r+1)/(len(mmax)+1))

```

```

In [ ]: if len(params) == 2:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    if len(params) == 3:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    if len(params) == 4:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    if len(params) == 6:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')

```

```

plt.plot(monthly_norm, label='Normal fit', linestyle='--')
plt.plot(monthly_beta, label='Beta', linestyle=':', color='black')
plt.xlim(2000, 2500); plt.ylim(0, 0.015); plt.legend(loc='upper right'); plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf, color='Red', label='Bestfit -'+str(name))
plt.plot(x, monthly_weib, label='Weibull fit', color='blue'); plt.hist(mmax, density=True)
plt.plot(x, monthly_gumb, color='orange', label='Gumbel fit')
plt.plot(monthly_norm, label='Normal fit', linestyle='--')
plt.plot(monthly_beta, label='Beta', linestyle=':', color='black')
plt.title('Monthly maxima distribution');
plt.xlim(2300, 2500); plt.ylim(0, 0.0015); plt.legend(loc='upper right'); plt.show()

plt.figure(figsize=(8,6))
plt.scatter(mmax, occmmax, color='green', facecolors='none', s=50, label='Data'); plt.ysca
plt.ylim(1e-6, 2); plt.xlim(2200, 3000)
plt.plot(x, 1-monthly_bestfit_cdf, color='Red', label='Bestfit -'+str(name))
plt.plot(x, (1-scstats.weibull_min.cdf(x, c_m, loc_m, scale_m)), label='Weibull fit', colo
plt.plot(x, (1-scstats.gumbel_r.cdf(x, mu_m, beta_m)), label='Gumbel fit', color='Orange')
plt.plot(1-scstats.norm.cdf(x, loc_mn, scale_mn), label='Normal fit', linestyle='--')
plt.plot(1-scstats.beta.cdf(x, a_m, b_m, loc_mb, scale_mb), label='Beta fit', linestyle=':

plt.title('ICDF Monthly maxima'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Exce
plt.show()

```

-----|-----Yearly maxima|-----

```

In [ ]: #yearly maxima wordt berekend door dist van monthly maxima te simuleren voor 1000 ja
years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12): #1 jaar heeft 12 maanden
        d = bestfit.rvs(params[0], params[1], params[2], size=1)
        ytemp[j] = d
    yrmax[k] = np.max(d)

dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(yrmax, loc=0, scale=1)
c_yr = parameters[0]; loc_yr = parameters[1]; scale_yr = parameters[2]
yearly_weib = scstats.weibull_min.pdf(x, c_yr, loc_yr, scale_yr)

parameters = scstats.gumbel_r.fit(yrmax)
mu_yr = parameters[0]
beta_yr = parameters[1]
yearly_gumb = scstats.gumbel_r.pdf(x, mu_yr, beta_yr)

parameters = scstats.genextreme.fit(yrmax)
c_yrg = parameters[0]; loc_yrg = parameters[1]; scale_yrg = parameters[2]
yearly_gev = scstats.genextreme.pdf(x, c_yrg, loc_yrg, scale_yrg)

parameters = scstats.norm.fit(yrmax)

```

```

loc_yrn = parameters[0]; scale_yrn = parameters[1]
yearly_norm = scstats.norm.pdf(x,loc_yrn,scale_yrn)

parameters = scstats.beta.fit(yrmax)
a_yr,b_yr,loc_yrb,scale_yrb = parameters
yearly_beta = scstats.beta.pdf(x,a_yr,b_yr,loc_yrb,scale_yrb)

##PLOT##
yrmax[:, :-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = ((r+1)/(len(yrmax)+1))

```

```

In [ ]: if len(params) == 2:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    if len(params) == 3:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    if len(params) == 4:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
    if len(params) == 6:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
if len(params) == 2:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 3:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 4:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 5:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 6:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(2000,2500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str
if len(params) == 3:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit
if len(params) == 4:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3]),color='Red',labe
if len(params) == 5:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4]),color=
if len(params) == 6:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[

```

```

plt.plot(x,yearly_weib,label='Weibull fit',color='blue');plt.hist(yrmax,density=True)
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.title('Yearly maxima distribution');
plt.xlim(2300,2500);plt.ylim(0,0.003);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Data');plt.ys
plt.ylim(1e-6,2);plt.xlim(2200,3000)
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1]))
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1]))
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2]))
    plt.plot(bestfit.sf(x,params[0],params[1],params[2]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2]))
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2])+' '+str(params[3]))
    plt.plot(bestfit.sf(x,params[0],params[1],params[2],params[3]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2])+' '+str(params[3]))
if len(params) == 5:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2])+' '+str(params[3])+' '+str(params[4]))
    plt.plot(bestfit.sf(x,params[0],params[1],params[2],params[3],params[4]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2])+' '+str(params[3])+' '+str(params[4]))
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2])+' '+str(params[3])+' '+str(params[4])+' '+str(params[5]))
    plt.plot(bestfit.sf(x,params[0],params[1],params[2],params[3],params[4],params[5]),color='Red',label='Bestfit -'+str(params[0])+' '+str(params[1])+' '+str(params[2])+' '+str(params[3])+' '+str(params[4])+' '+str(params[5]))

plt.plot(x,(1-scstats.weibull_min.cdf(x,c_yr,loc_yr,scale_yr)),label='Weibull fit',color='blue')
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_yr,beta_yr)),label='Gumbel fit',color='Orange')
plt.plot(x,(1-scstats.norm.cdf(x,loc_yrn,scale_yrn)),label='Normal fit',linestyle='--')
plt.plot(x,(1-scstats.beta.cdf(x,a_yr,b_yr,loc_yrb,scale_yrb)),label='Beta fit',linestyle=':')

plt.title('ICDF Yearly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceedance probability');plt.show()

```

## MC-Simulation below

```

In [ ]: ##FUNCTIONS##
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def normal(x,mu,sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x,v):
    return scstats.t.pdf(x,v)
def lognorm_sample(mu,sigma):
    return scstats.lognorm.rvs(s=sigma,scale=np.exp(mu))
def gumbel_sample(mu,beta):
    return scstats.gumbel_r.rvs(loc=mu,scale=beta)
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu,sigma):
    return scstats.norm.rvs(loc=mu,scale=sigma)
def weibull(x,shape,loc,scale):
    return scstats.weibull_min.pdf(x,c=shape,loc=loc,scale=scale)
def gumbel(x,mu,beta):
    z = (x- mu)/beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x,mu,beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))

```

```

In [ ]: ###JCSS CONC###

```



```

x = np.linspace(0,100e3,100000)
m = 3.85;v=10;s=0.09;n=3
mu_y1 = 1; sigma_y1=0.06
e=100000
fc_arr = np.zeros(e);ft_arr=np.zeros(e);ec_arr = np.zeros(e);eps_arr = np.zeros(e);
for r in range(e):
    fc_arr[r] = (np.exp(m+student_t_sample(v)*s*(1+(1/n))**0.5))
    ft_arr[r] = 0.3*(fc_arr[r]**(2/3))
    ec_arr[r] = 10.5*(fc_arr[r]**(1/3))/(1+0.7*2.5)
    eps_arr[r] = 6e-3 * (fc_arr[r]**(-1/6))*(1+0.7*2.5)

##Fitting dist##
fcp = scstats.lognorm.fit(fc_arr,loc=0,scale=1);
ftp =scstats.lognorm.fit(ft_arr,loc=0,scale=1);
ecp = scstats.lognorm.fit(ec_arr,loc=0,scale=1);
epsr = scstats.lognorm.fit(eps_arr);

##JCSS Reinforcing steel + Moment resistance##
def F_steel(Sxx,n,d): #F_steel in kN
    X12 = normal_sample(0,22)
    X13 = normal_sample(0,8)
    mu1 = Sxx + 2*30
    mu11 = mu1 * ((0.87 + 0.13*np.exp(-0.08*d))**-1)
    X11 = normal_sample(mu11,19)
    fyd = X11 + X12 + X13

    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)

    return fyd * As_sample / 1e3 , fyd

def F_conc(Sxx,n,d): #F_conc in kN
    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)
    xc = (As_sample)*F_steel(Sxx,n,d)[1] / (0.85*lognorm_sample(np.log(fcp[2]),(fcp[
    Fcd = 0.85*xc * lognorm_sample(np.log(fcp[2]),fcp[0])
    return Fcd/1e3

def M_res(Sxx,n,d,z): #M_res in kNm
    fs = F_steel(Sxx,n,d)[0]
    fc = F_conc(Sxx,n,d)
    focc = np.minimum(fs,fc)
    mres = focc*z
    return mres/1e3

```

```
In [ ]: sigma_load = lognorm_sample(np.log(1),0.1)
sigma_res = lognorm_sample(np.log(1.2),0.15)
```

```
In [ ]: L=15;
M_G_ED = 1.35 * (1/8) * (0.5*1*25)* (L**2);
print(M_G_ED)
x = np.linspace(0,100e3,100e3)
```

```
In [ ]: def MCsim(Sxx,n,d,z,e):
    print('Running a Monte-Carlo simulation for',e/1e6,'million times.')
    loadarr = np.zeros(e)
    resarr = np.zeros(e)
    zarr = np.zeros(e)
    zarr2 = np.zeros(e)

    with tqdm(total=e, file=sys.stdout) as pbar:
        for r in range(e):
            pbar.set_description('MC-simulation')
            pbar.update(1)
```

```

loadarr[r] = (scstats.genlogistic.rvs(6.386453,2129.676299,34.0627,size=
resarr[r] = M_res(Sxx,n,d,z)*lognorm_sample(np.log(1.2),0.15)
zarr[r] = loadarr[r]/resarr[r]
zarr2[r] = resarr[r] - loadarr[r]

##ICDF plot##
zarr[::-1].sort()
occzarr= np.zeros_like(zarr)
for r in range(len(zarr)):
    occzarr[r] = ((r+1)/(len(zarr)+1))

return (loadarr,resarr,zarr,occzarr,zarr2)

```

```

In [ ]: f= MCsim(500,21,32,567,1000000)
loadarr,resarr,zarr,occzarr,zarr2 = f

```

```

In [ ]: x = np.linspace(0,10e3,100e3)
##PLOTS##
plt.figure(figsize=(16,8))
plt.scatter(resarr,loadarr,facecolors='none',color='green',marker='o',s=1,label='Sim
plt.xlim(0,9000); plt.ylim(0,9000); plt.xlabel('Resistance [kNm]');plt.ylabel('Be

plt.figure(figsize=(16,8))
plt.hist(loadarr,bins=50,alpha=0.5,label='S(x)',density=True,color='green',ec="k") ;
plt.hist(resarr,bins=50,alpha=0.5,label='R(x)',density=True,color='red',ec="k");
plt.hist(zarr2,bins=50,alpha=0.5,label='R(x)-S(x)',density=True,ec="k",color='blue')
plt.legend(loc='upper right');plt.xlabel('Bending moment [kNm]');plt.ylabel('Probabi
plt.show()

plt.figure(figsize=(16,8))
plt.scatter(zarr,occzarr,color='green',facecolors='none',s=50,label='Simulation');pl
plt.ylim(1e-7,2);plt.xlim(0,1.5)
plt.title('ICDF MC-sim');plt.xlabel('Load / Resistance [-]');plt.ylabel('Exceedance
plt.show()

#calculate prob of failure
x_pof = zarr; y_pof = occzarr;
l = len(x_pof[x_pof > 1]) ; ecdf = ECDF(zarr);

print("")
pof = len(zarr[zarr>1])/len(zarr)
#beta = (1-np.mean(zarr))/(scstats.weibull_min.std(c=params[0],loc=params[1],scale=p
print('The probability of failure is','{: .2E}'.format(Decimal(pof)));
#print('The reliability index is', beta) #CIE4130

```

```

In [ ]: zarr

```

```

In [ ]: df3 = pd.DataFrame(f)
df3.to_csv('CASE_STUDY_L15_FINAL.csv')

```

```

In [ ]:

```

# Case study span length 20 m curve fitting + MC sim

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
from distfit import distfit
```

```
In [ ]: df = pd.read_csv('df_4400C_load_L1_L20.csv', low_memory=False, index_col=0)
x = np.linspace(0, 100e3, 100e3)
dist_names = [ 'alpha', 'anglit', 'arcsine', 'beta', 'betaprime', 'bradford', 'burr'
```

```
In [ ]: df.drop(df.tail(1).index, inplace=True)
L5 = df['L5'].to_numpy()
L15 = df['L15'].to_numpy()
L20 = df['L20'].to_numpy()
```

-----|-----Daily maxima-----|-----

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = L20
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)
dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(dmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));
```

```
In [ ]: if len(params) == 2:
    daily_bestfit_pdf = bestfit.pdf(x, params[0], params[1])
    daily_bestfit_cdf = bestfit.cdf(x, params[0], params[1])
if len(params) == 3:
    daily_bestfit_pdf = bestfit.pdf(x, params[0], params[1], params[2])
    daily_bestfit_cdf = bestfit.cdf(x, params[0], params[1], params[2])
if len(params) == 4:
    daily_bestfit_pdf = bestfit.pdf(x, params[0], params[1], params[2], params[3])
    daily_bestfit_cdf = bestfit.cdf(x, params[0], params[1], params[2], params[3])
if len(params) == 5:
    daily_bestfit_pdf = bestfit.pdf(x, params[0], params[1], params[2], params[3], params[4])
    daily_bestfit_cdf = bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4])
if len(params) == 6:
    daily_bestfit_pdf = bestfit.pdf(x, params[0], params[1], params[2], params[3], params[4], params[5])
    daily_bestfit_cdf = bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4], params[5])
print(params)
```

```
In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit - '+'"+str(name))
plt.xlim(0,600);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()
##---##
plt.figure(figsize=(8,6));
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit - '+'"+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(400,500);plt.ylim(0,0.0008);

plt.figure(figsize=(8,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
plt.plot(1-daily_bestfit_cdf,color='Red',label='Bestfit - '+'"+str(name))

plt.title('ICDF Daily maxima');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceedanc
plt.show()
```

-----|-----Monthly maxima|-----

```
In [ ]: maanden = 12*25 #25 jaar simuleren
mmax = np.zeros(maanden)
for k in range(maanden): #12 maanden --> 1 yr
    mtemp = np.zeros(30*150)
    for j in range(30*150): #30 dagen / maand , 150 voertuigen per dag
        d = bestfit.rvs(params[0],params[1],params[2],size=1)
        mtemp[j] = d
    mmax[k] = np.max(mtemp)

dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

##PLOT##
mmax[:, :-1].sort()
occmmax= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmax[r] = (r/(len(mmax)+1))
```

```
In [ ]: if len(params) == 2:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
if len(params) == 3:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
if len(params) == 4:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
if len(params) == 5:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
if len(params) == 6:
    monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
    monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
print(params)
```

```
In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
```

```

plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlim(300,600);plt.ylim(0,0.035);plt.legend(loc='upper right');plt.show()
##---##
plt.figure(figsize=(8,6));
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.title('Monthly maxima distribution');plt.legend(loc='upper right')
plt.xlim(500,550);plt.ylim(0,0.001);

plt.figure(figsize=(8,6))
plt.scatter(mmax,occmmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
plt.plot(1-monthly_bestfit_cdf,color='Red',label='Bestfit -'+str(name))

plt.title('ICDF Monthly maxima');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceeda
plt.show()

```

-----|-----Yearly maxima|-----

```

In [ ]: #yearly maxima wordt berekend door dist van monthly maxima te simuleren voor 1000 ja
years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12):#1 jaar heeft 12 maanden
        d = bestfit.rvs(params[0],params[1],params[2],size=1)
        ytemp[j] = d
    yrmax[k] = np.max(d)

dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

##PLOT##
yrmax[:,::-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = (r/(len(yrmax)+1))

```

```

In [ ]: if len(params) == 2:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    if len(params) == 3:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    if len(params) == 4:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
    if len(params) == 6:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))

```

```
plt.xlim(300,600);plt.ylim(0,0.035);plt.legend(loc='upper right');plt.show()
##---##
plt.figure(figsize=(8,6));
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit - '+'"+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.title('Yearly maxima distribution');plt.legend(loc='upper right')
plt.xlim(500,550);plt.ylim(0,0.0008);

plt.figure(figsize=(8,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Data');plt.ys
plt.xlim(300,600);plt.plot(1-yearly_bestfit_cdf,color='Red',label='Bestfit - '+'"+str

plt.title('ICDF Yearly maxima');plt.xlabel('Load effect [kNm]');plt.ylabel('Exceedan
plt.show()
```

## MC-Simulation below

```
In [ ]: ##FUNCTIONS##
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def normal(x,mu,sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x,v):
    return scstats.t.pdf(x,v)
def lognorm_sample(mu,sigma):
    return scstats.lognorm.rvs(s=sigma,scale=np.exp(mu))
def gumbel_sample(mu,beta):
    return scstats.gumbel_r.rvs(loc=mu,scale=beta)
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu,sigma):
    return scstats.norm.rvs(loc=mu,scale=sigma)
def weibull(x,shape,loc,scale):
    return scstats.weibull_min.pdf(x,c=shape,loc=loc,scale=scale)
def gumbel(x,mu,beta):
    z = (x- mu)/beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x,mu,beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))
```

```
In [ ]: ###JCSS CONC###
x = np.linspace(0,100e3,100000)
m = 3.85;v=10;s=0.09;n=3
mu_y1 = 1; sigma_y1=0.06
e=100000
fc_arr = np.zeros(e);ft_arr=np.zeros(e);ec_arr = np.zeros(e);eps_arr = np.zeros(e);
for r in range(e):
    fc_arr[r] = (np.exp(m+student_t_sample(v)*s*(1+(1/n))**0.5))
    ft_arr[r] = 0.3*(fc_arr[r]**(2/3))
    ec_arr[r] = 10.5*(fc_arr[r]**(1/3))/(1+0.7*2.5)
    eps_arr[r] = 6e-3 * (fc_arr[r]**(-1/6))*(1+0.7*2.5)

##Fitting dist##
fcp = scstats.lognorm.fit(fc_arr,loc=0,scale=1);
ftp =scstats.lognorm.fit(ft_arr,loc=0,scale=1);
ecp = scstats.lognorm.fit(ec_arr,loc=0,scale=1);
eps_p = scstats.lognorm.fit(eps_arr);

##JCSS Reinforcing steel + Moment resistance##
```

```

def F_steel(Sxx,n,d): #F_steel in kN
    X12 = normal_sample(0,22)
    X13 = normal_sample(0,8)
    mu1 = Sxx + 2*30
    mu11 = mu1 * ((0.87 + 0.13*np.exp(-0.08*d))**-1)
    X11 = normal_sample(mu11,19)
    fyd = X11 + X12 + X13

    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)

    return fyd * As_sample / 1e3 , fyd

def F_conc(Sxx,n,d): #F_conc in kN
    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)
    xc = (As_sample)*F_steel(Sxx,n,d)[1] / (0.85*lognorm_sample(np.log(fcp[2]),(fcp[
    Fcd = 0.85*xc * lognorm_sample(np.log(fcp[2]),fcp[0])
    return Fcd/1e3

def M_res(Sxx,n,d,z): #M_res in kNm
    fs = F_steel(Sxx,n,d)[0]
    fc = F_conc(Sxx,n,d)
    focc = np.minimum(fs,fc)
    mres = focc*z
    return mres/1e3

```

```

In [ ]: sigma_load = lognorm_sample(np.log(1),0.1)
        sigma_res = lognorm_sample(np.log(1.2),0.15)

```

```

In [ ]: L=20;
        M_G_ED = 1.35 * (1/8) * (0.5*1*25)* (L**2);
        print(M_G_ED)
        x = np.linspace(0,100e3,100e3)

```

```

In [ ]: def MCsim(Sxx,n,d,z,e):
        print('Running a Monte-Carlo simulation for',e/1e6,'million times.')
        loadarr = np.zeros(e)
        resarr = np.zeros(e)
        zarr = np.zeros(e)
        zarr2 = np.zeros(e)

        with tqdm(total=e, file=sys.stdout) as pbar:
            for r in range(e):
                pbar.set_description('MC-simulation')
                pbar.update(1)
                loadarr[r] = (scstats.gumbel_r.rvs(3365.841449,63.268484,size=1)+M_G_ED)
                resarr[r] = M_res(Sxx,n,d,z)*lognorm_sample(np.log(1.2),0.15)
                zarr[r] = loadarr[r]/resarr[r]
                zarr2[r] = resarr[r] - loadarr[r]

        ##ICDF plot##
        zarr[::-1].sort()
        occzarr= np.zeros_like(zarr)
        for r in range(len(zarr)):
            occzarr[r] = ((r+1)/(len(zarr)+1))

        return (loadarr,resarr,zarr,occzarr,zarr2)

```

```

In [ ]: f= MCsim(500,32,32,567,1000000)
        loadarr,resarr,zarr,occzarr,zarr2 = f

```



```

In [ ]: x = np.linspace(0,10e3,100e3)
        ##PLOTS##
        plt.figure(figsize=(16,8))
        plt.scatter(resarr,loadarr,facecolors='none',color='green',marker='o',s=1,label='Sim
        plt.xlim(0,3000); plt.ylim(0,3000); plt.xlabel('Resistance [kNm]');plt.ylabel('Be

        plt.figure(figsize=(16,8))
        plt.hist(loadarr,bins=50,alpha=0.5,label='S(x)',density=True,color='green',ec="k") ;
        plt.hist(resarr,bins=50,alpha=0.5,label='R(x)',density=True,color='red',ec="k");
        plt.hist(zarr,bins=50,alpha=0.5,label='R(x)-S(x)',density=True,ec="k",color='blue')
        plt.legend(loc='upper right');plt.xlabel('Bending moment [kNm]');plt.ylabel('Probabi
        plt.show()

        plt.figure(figsize=(16,8))
        plt.scatter(zarr,occzarr,color='green',facecolors='none',s=50,label='Simulation');pl
        plt.ylim(5e-6,2);plt.xlim(0,1.5)
        plt.title('ICDF MC-sim');plt.xlabel('Load / Resistance [-]');plt.ylabel('Exceedance
        plt.show()

        #calculate prob of failure
        x_pof = zarr; y_pof = occzarr;
        l = len(x_pof[x_pof > 1]) ; ecdf = ECDF(zarr);

        print("")
        pof = y_pof[l]
        #beta = (1-np.mean(zarr))/(scstats.weibull_min.std(c=params[0],loc=params[1],scale=p
        print('The probability of failure is',"{:0.2E}".format(Decimal(pof)));
        #print('The reliability index is', beta) #CIE4130

```

```

In [ ]: df3 = pd.DataFrame(f);
        df3.to_csv('CASE_STUDY_L20_FINAL.csv')

```

```

In [ ]:

```

# Curve fitting span modified LP data length 5 m + MC sim

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
from distfit import distfit
```

```
In [ ]: df = pd.read_csv('df_4400C_load_L1_L20.csv', low_memory=False, index_col=0)
x = np.linspace(0, 100e3, 100e3)
dist_names = [ 'alpha', 'anglit', 'arcsine', 'beta', 'betaprime', 'bradford', 'burr'
```

```
In [ ]: df.drop(df.tail(1).index, inplace=True)
L5 = df['L5'].to_numpy()
L15 = df['L15'].to_numpy()
L20 = df['L20'].to_numpy()
```

-----|-----Daily maxima-----|-----

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = L5
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)
dmax[:, -1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = ((r+1)/(len(dmax)+1))
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(dmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(dmax, loc=0, scale=1)
c_d = parameters[0]; loc_d = parameters[1]; scale_d = parameters[2]
daily_weib = scstats.weibull_min.pdf(x, c_d, loc_d, scale_d)

parameters = scstats.gumbel_r.fit(dmax)
mu_d = parameters[0]
beta_d = parameters[1]
daily_gumb = scstats.gumbel_r.pdf(x, mu_d, beta_d)

parameters = scstats.genextreme.fit(dmax)
c_dg = parameters[0]; loc_dg = parameters[1]; scale_dg = parameters[2]
daily_gev = scstats.genextreme.pdf(x, c_dg, loc_dg, scale_dg)

parameters = scstats.norm.fit(dmax)
loc_dn = parameters[0]; scale_dn = parameters[1]
```

```

daily_norm = scstats.norm.pdf(x,loc_dn,scale_dn)

parameters = scstats.beta.fit(dmax)
a_d,b_d,loc_db,scale_db = parameters
daily_beta = scstats.beta.pdf(x,a_d,b_d,loc_db,scale_db)

```

```

In [ ]: if len(params) == 2:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    if len(params) == 3:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    if len(params) == 4:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4])
    if len(params) == 6:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlabel('Bending moment [kNm]');plt.ylabel('Frequency')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.xlim(100,700);plt.ylim(0,0.020);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(350,600);plt.ylim(0,0.002);plt.show()

plt.figure(figsize=(8,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
plt.xlim(0,800)
#plt.plot(1-daily_bestfit_cdf,color='Red',label='Bestfit -'+str(name))
plt.plot(bestfit.sf(x,params[0],params[1],params[2]),color='Red',label='Bestfit -'+
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange')
plt.plot(1-scstats.norm.cdf(x,loc_dn,scale_dn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_d,b_d,loc_db,scale_db),label='Beta fit',linestyle=':

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceed
plt.show()

```

-----|-----Monthly maxima|-----

```

In [ ]: maanden = 12*25 #25 jaar simuleren
mmax = np.zeros(maanden)
for k in range(maanden): #12 maanden --> 1 yr

```

```

mtemp = np.zeros(30*150)
for j in range(30*150): #30 dagen / maand , 150 voertuigen per dag
    d = scstats.gumbel_r.rvs(mu_d,beta_d,size=1)
    mtemp[j] = d
mmax[k] = np.max(mtemp)

dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(mmax,loc=0,scale=1)
c_m = parameters[0]; loc_m = parameters[1]; scale_m = parameters[2]
monthly_weib = scstats.weibull_min.pdf(x,c_m,loc_m,scale_m)

parameters = scstats.gumbel_r.fit(mmax)
mu_m = parameters[0]
beta_m = parameters[1]
monthly_gumb = scstats.gumbel_r.pdf(x,mu_m,beta_m)

parameters = scstats.genextreme.fit(mmax)
c_mg = parameters[0]; loc_mg = parameters[1]; scale_mg = parameters[2]
monthly_gev = scstats.genextreme.pdf(x,c_mg,loc_mg,scale_mg)

parameters = scstats.norm.fit(mmax)
loc_mn = parameters[0]; scale_mn = parameters[1]
monthly_norm = scstats.norm.pdf(x,loc_mn,scale_mn)

parameters = scstats.beta.fit(mmax)
a_m,b_m,loc_mb,scale_mb = parameters
monthly_beta = scstats.beta.pdf(x,a_m,b_m,loc_mb,scale_mb)

##PLOT##
mmax[:, :-1].sort()
occmmax= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmax[r] = ((r+1)/(len(mmax)+1))

```

```

In [ ]: if len(params) == 2:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
    if len(params) == 3:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
    if len(params) == 4:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    if len(params) == 6:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit - '+'"+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');

```

```

plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')
plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(400,700);plt.ylim(0,0.025);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True)
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')
plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
plt.title('Monthly maxima distribution');
plt.xlim(600,700);plt.ylim(0,0.0015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6))
plt.scatter(mmax,occmmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
#plt.plot(1-monthly_bestfit_cdf(color='Red',label='Bestfit -'+str(name)))
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange')
plt.plot(1-scstats.norm.cdf(x,loc_mn,scale_mn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_m,b_m,loc_mb,scale_mb),label='Beta fit',linestyle=':

plt.title('ICDF Monthly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exce
plt.show()

```

-----|Yearly maxima|-----

```

In [ ]: #yearly maxima wordt berekend door dist van monthly maxima te simuleren voor 1000 ja
years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12):#1 jaar heeft 12 maanden
        d = scstats.gumbel_r.rvs(mu_m,beta_m,size=1)
        ytemp[j] = d
    yrmax[k] = np.max(d)

dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'));
params = model.get('params');
bestfit = (model.get('distr'));

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(yrmax,loc=0,scale=1)
c_yr = parameters[0]; loc_yr = parameters[1]; scale_yr = parameters[2]
yearly_weib = scstats.weibull_min.pdf(x,c_yr,loc_yr,scale_yr)

parameters = scstats.gumbel_r.fit(yrmax)
mu_yr = parameters[0]
beta_yr = parameters[1]
yearly_gumb = scstats.gumbel_r.pdf(x,mu_yr,beta_yr)

parameters = scstats.genextreme.fit(yrmax)
c_yrg = parameters[0]; loc_yrg = parameters[1]; scale_yrg = parameters[2]
yearly_gev = scstats.genextreme.pdf(x,c_yrg,loc_yrg,scale_yrg)

parameters = scstats.norm.fit(yrmax)
loc_yrn = parameters[0]; scale_yrn = parameters[1]

```

```

yearly_norm = scstats.norm.pdf(x,loc_yrn,scale_yrn)

parameters = scstats.beta.fit(yrmax)
a_yr,b_yr,loc_yrb,scale_yrb = parameters
yearly_beta = scstats.beta.pdf(x,a_yr,b_yr,loc_yrb,scale_yrb)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(yrmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

##PLOT##
yrmax[:-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = ((r+1)/(len(yrmax)+1))

```

```

In [ ]: if len(params) == 2:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
if len(params) == 3:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
if len(params) == 4:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
if len(params) == 5:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
if len(params) == 6:
    yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
    yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
if len(params) == 2:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 3:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 4:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 5:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 6:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(400,800);plt.ylim(0,0.02);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str
if len(params) == 3:

```



```

plt.plot(bestfit.pdf(x, params[0], params[1], params[2]), color='Red', label='Bestfit
if len(params) == 4:
plt.plot(bestfit.pdf(x, params[0], params[1], params[2], params[3]), color='Red', labe
if len(params) == 5:
plt.plot(bestfit.pdf(x, params[0], params[1], params[2], params[3], params[4]), color=
if len(params) == 6:
plt.plot(bestfit.pdf(x, params[0], params[1], params[2], params[3], params[4], params[
plt.plot(x, yearly_weib, label='Weibull fit', color='blue'); plt.hist(yrmax, density=True
plt.plot(x, yearly_gumb, color='orange', label='Gumbel fit')
plt.plot(yearly_norm, label='Normal fit', linestyle='--')
plt.plot(yearly_beta, label='Beta', linestyle=':', color='black')
plt.title('Yearly maxima distribution');
plt.xlim(600, 800); plt.ylim(0, 0.003); plt.legend(loc='upper right'); plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(yrmax, occyrmax, color='green', facecolors='none', s=50, label='Data'); plt.ys
plt.ylim(1e-6, 2); plt.xlim(400, 1000)
if len(params) == 2:
plt.plot(1-bestfit.cdf(x, params[0], params[1]), color='Red', label='Bestfit - '+'"+s
plt.plot(bestfit.sf(x, params[0], params[1]), color='Red', label='Bestfit - '+'"+str(
if len(params) == 3:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2]), color='Red', label='Bestf
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2]), color='Red', label='Bestf
if len(params) == 4:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3]), color='Red', la
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3]), color='Red', la
if len(params) == 5:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4]), colo
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4]), colo
if len(params) == 6:
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4], param
plt.plot(1-bestfit.cdf(x, params[0], params[1], params[2], params[3], params[4], param

plt.plot(x, (1-scstats.weibull_min.cdf(x, c_yr, loc_yr, scale_yr)), label='Weibull fit', c
plt.plot(x, (1-scstats.gumbel_r.cdf(x, mu_yr, beta_yr)), label='Gumbel fit', color='Orang
plt.plot(1-scstats.norm.cdf(x, loc_yrn, scale_yrn), label='Normal fit', linestyle='--')
plt.plot(1-scstats.beta.cdf(x, a_yr, b_yr, loc_yrb, scale_yrb), label='Beta fit', linestyl

plt.title('ICDF Yearly maxima'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Excee
plt.show()

```

In [ ]: mu\_yr, beta\_yr

## MC-Simulation below

```

In [ ]: ##FUNCTIONS##
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def normal(x, mu, sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x, v):
    return scstats.t.pdf(x, v)
def lognorm_sample(mu, sigma):
    return scstats.lognorm.rvs(s=sigma, scale=np.exp(mu))
def gumbel_sample(mu, beta):
    return scstats.gumbel_r.rvs(loc=mu, scale=beta)
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu, sigma):
    return scstats.norm.rvs(loc=mu, scale=sigma)

```



```

def weibull(x,shape,loc,scale):
    return scstats.weibull_min.pdf(x,c=shape,loc=loc,scale=scale)
def gumbel(x,mu,beta):
    z = (x- mu)/beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x,mu,beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))

```

```

In [ ]: ###JCSS CONC###
x = np.linspace(0,100e3,100000)
m = 3.85;v=10;s=0.09;n=3
mu_y1 = 1; sigma_y1=0.06
e=100000
fc_arr = np.zeros(e);ft_arr=np.zeros(e);ec_arr = np.zeros(e);eps_arr = np.zeros(e);
for r in range(e):
    fc_arr[r] = (np.exp(m+student_t_sample(v)*s*(1+(1/n)**0.5))
    ft_arr[r] = 0.3*(fc_arr[r]**(2/3))
    ec_arr[r] = 10.5*(fc_arr[r]**(1/3))/(1+0.7*2.5)
    eps_arr[r] = 6e-3 * (fc_arr[r]**(-1/6))*(1+0.7*2.5)

##Fitting dist##
fcp = scstats.lognorm.fit(fc_arr,loc=0,scale=1);
ftp =scstats.lognorm.fit(ft_arr,loc=0,scale=1);
ecp = scstats.lognorm.fit(ec_arr,loc=0,scale=1);
epsf = scstats.lognorm.fit(eps_arr);

##JCSS Reinforcing steel + Moment resistance##
def F_steel(Sxx,n,d): #F_steel in kN
    X12 = normal_sample(0,22)
    X13 = normal_sample(0,8)
    mu1 = Sxx + 2*30
    mu11 = mu1 * ((0.87 + 0.13*np.exp(-0.08*d))**-1)
    X11 = normal_sample(mu11,19)
    fyd = X11 + X12 + X13

    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)

    return fyd * As_sample / 1e3 , fyd

def F_conc(Sxx,n,d): #F_conc in kN
    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)
    xc = (As_sample)*F_steel(Sxx,n,d)[1] / (0.85*lognorm_sample(np.log(fcp[2]),(fcp[
    Fcd = 0.85*xc * lognorm_sample(np.log(fcp[2]),fcp[0])
    return Fcd/1e3

def M_res(Sxx,n,d,z): #M_res in kNm
    fs = F_steel(Sxx,n,d)[0]
    fc = F_conc(Sxx,n,d)
    focc = np.minimum(fs,fc)
    mres = focc*z
    return mres/1e3

```

```

In [ ]: sigma_load = lognorm_sample(np.log(1),0.1)
sigma_res = lognorm_sample(np.log(1.2),0.15)

```

```

In [ ]: L=5;
M_G_ED = 1.35 * (1/8) * (0.5*1*25)* (L**2);
print(M_G_ED)
x = np.linspace(0,100e3,100e3)

```

```

In [ ]: def MCsim(Sxx,n,d,z,e):

```

```

print('Running a Monte-Carlo simulation for',e/1e6,'million times.')
loadarr = np.zeros(e)
resarr = np.zeros(e)
zarr = np.zeros(e)
zarr2 = np.zeros(e)

with tqdm(total=e, file=sys.stdout) as pbar:
    for r in range(e):
        pbar.set_description('MC-simulation')
        pbar.update(1)
        loadarr[r] = (scstats.gumbel_r.rvs(523.200131759064,26.70816300782262,si
resarr[r] = M_res(Sxx,n,d,z)*lognorm_sample(np.log(1.2),0.15)
zarr[r] = loadarr[r]/resarr[r]
zarr2[r] = resarr[r] - loadarr[r]

##ICDF plot##
zarr[::-1].sort()
occzarr= np.zeros_like(zarr)
for r in range(len(zarr)):
    occzarr[r] = ((r+1)/(len(zarr)+1))

return (loadarr,resarr,zarr,occzarr,zarr2)

```

```
In [ ]: f= MCsim(500,7,32,405,1000000)
loadarr,resarr,zarr,occzarr,zarr2 = f
```

```
In [ ]: df3 = pd.DataFrame(f)
df3.to_csv('CASE_STUDY_L5_FINAL.csv')
```

```
In [ ]: x = np.linspace(0,10e3,100e3)
##PLOTS##
plt.figure(figsize=(16,8))
plt.scatter(resarr,loadarr,facecolors='none',color='green',marker='o',s=1,label='Sim
plt.xlim(0,3000); plt.ylim(0,3000); plt.xlabel('Resistance [kNm]');plt.ylabel('Be

plt.figure(figsize=(16,8))
plt.hist(loadarr,bins=50,alpha=0.5,label='S(x)',density=True,color='green',ec="k") ;
plt.hist(resarr,bins=50,alpha=0.5,label='R(x)',density=True,color='red',ec="k");
plt.hist(zarr2,bins=50,alpha=0.5,label='R(x)-S(x)',density=True,ec="k",color='blue')
plt.legend(loc='upper right');plt.xlabel('Bending moment [kNm]');plt.ylabel('Probabi
plt.show()

plt.figure(figsize=(16,8))
plt.scatter(zarr,occzarr,color='green',facecolors='none',s=50,label='Simulation');pl
plt.ylim(5e-6,2);plt.xlim(0,1.5)
plt.title('ICDF MC-sim');plt.xlabel('Load / Resistance [-]');plt.ylabel('Exceedance
plt.show()

#calculate prob of failure
x_pof = zarr; y_pof = occzarr;
l = len(x_pof[x_pof > 1]) ; ecdf = ECDF(zarr);

print("")
pof = y_pof[l]
#beta = (1-np.mean(zarr))/(scstats.weibull_min.std(c=params[0],loc=params[1],scale=p
print('The probability of failure is',"{: .2E}".format(Decimal(pof)));
#print('The reliability index is', beta) #CIE4130

```

# Curve fitting WIM data

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
#pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

```
In [ ]: df = pd.read_csv('WIM.csv', low_memory=False)
df = df[df['GVW'] > 120]
x = np.linspace(0, 100e3, 100e3)
dist_names = [ 'alpha', 'anglit', 'arcsine', 'beta', 'betaprime', 'bradford', 'burr'
```

```
In [ ]: ##FUNCTIONS##
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def normal(x, mu, sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x, v):
    return scstats.t.pdf(x, v)
def lognorm_sample(mu, sigma):
    return scstats.lognorm.rvs(s=sigma, scale=np.exp(mu))
def gumbel_sample(mu, beta):
    return scstats.gumbel_r.rvs(loc=mu, scale=beta)
def lognorm(x, mu, sigma):
    return scstats.lognorm.pdf(x=x, s=sigma, scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu, sigma):
    return scstats.norm.rvs(loc=mu, scale=sigma)
def weibull(x, shape, loc, scale):
    return scstats.weibull_min.pdf(x, c=shape, loc=loc, scale=scale)
def gumbel(x, mu, beta):
    z = (x - mu)/beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x, mu, beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))
```

```
In [ ]: dagen = 365
dmax = np.zeros(dagen)
subset = df['McMax [kNm]'].to_numpy()
for k in range(dagen): #aantal dagen
    d = random.choices(subset, k=150)
    dmax[k] = np.max(d)
dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = (r/(len(dmax)+1))
```

```
In [ ]: dmax[::-1].sort()
occdmax = np.zeros_like(dmax)
for r in range(len(dmax)):
    occdmax[r] = ((r+1)/(len(dmax)+1))
```

```

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(dmax,loc=0,scale=1)
c_d = parameters[0]; loc_d = parameters[1]; scale_d = parameters[2]
daily_weib = weibull(x,c_d,loc_d,scale_d)

parameters = scstats.gumbel_r.fit(dmax)
mu_d = parameters[0]
beta_d = parameters[1]
daily_gumb = gumbel(x,mu_d,beta_d)

parameters = scstats.genextreme.fit(dmax)
c_dg = parameters[0]; loc_dg = parameters[1]; scale_dg = parameters[2]
daily_gev = scstats.genextreme.pdf(x,c_dg,loc_dg,scale_dg)

parameters = scstats.norm.fit(dmax)
loc_dn = parameters[0]; scale_dn = parameters[1]
daily_norm = scstats.norm.pdf(x,loc_dn,scale_dn)

parameters = scstats.beta.fit(dmax)
a_d,b_d,loc_db,scale_db = parameters
daily_beta = scstats.beta.pdf(x,a_d,b_d,loc_db,scale_db)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(dmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

```

```

In [ ]: if len(params) == 2:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
    if len(params) == 3:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
    if len(params) == 4:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],size=1)
    if len(params) == 5:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4])
        daily_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params[4])
    if len(params) == 6:
        daily_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],params[5])
        daily_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params[4],params[5])
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency')
plt.xlim(400,1300);plt.ylim(0,0.005);plt.legend(loc='upper right');plt.show()

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins')
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.xlabel('Bending moment [kNm]');plt.ylabel('Frequency')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')

```

```

plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.xlim(400,1300);plt.ylim(0,0.005);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.plot(daily_norm,label='Normal fit',linestyle='--')
plt.plot(daily_beta,label='Beta',linestyle=':',color='black')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(800,1300);plt.ylim(0,0.0025);plt.show()

plt.figure(figsize=(8,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
#plt.plot(1-daily_bestfit_cdf,color='Red',label='Bestfit -'+str(name))
plt.plot(bestfit.sf(x,params[0],params[1],params[2]),color='Red',label='Bestfit -'+
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange'
plt.plot(1-scstats.norm.cdf(x,loc_dn,scale_dn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_d,b_d,loc_db,scale_db),label='Beta fit',linestyle=':

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceed
plt.show()

```

-----MONTHLY-----

Please note: daily\_bestfit\_rvs does not properly work, so copy the function and let the code run.

```

In [ ]: parameters = scstats.norm.fit(dmax)
loc_dn,scale_dn = parameters ;daily_norm = scstats.norm.pdf(x,loc_dn,scale_dn)
print(loc_dn,scale_dn)

```

```

In [ ]: daily = daily_weib
maanden = 12*25 #15 jaar simuleren
mmax = np.zeros(maanden)
for k in range(maanden): #12 maanden --> 1 yr
    mtemp = np.zeros(30*150)
    for j in range(30*150): #30 dagen / maand , 150 voertuigen per dag
        d = scstats.norm.rvs(loc_dn,scale_dn,size=1)
        mtemp[j] = d
    mmax[k] = np.max(mtemp)

```

```

In [ ]: ###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(mmax,loc=0,scale=1)
c_m = parameters[0]; loc_m = parameters[1]; scale_m = parameters[2]
monthly_weib = weibull(x,c_m,loc_m,scale_m)

parameters = scstats.gumbel_r.fit(mmax)
mu_m = parameters[0]
beta_m = parameters[1]
monthly_gumb = gumbel(x,mu_m,beta_m)

parameters = scstats.genextreme.fit(mmax)
c_mg = parameters[0]; loc_mg = parameters[1]; scale_mg = parameters[2]
monthly_gev = scstats.genextreme.pdf(x,c_mg,loc_mg,scale_mg)

parameters = scstats.norm.fit(mmax)
loc_mn = parameters[0]; scale_mn = parameters[1]
monthly_norm = scstats.norm.pdf(x,loc_mn,scale_mn)

```

```

parameters = scstats.beta.fit(mmax)
a_m,b_m,loc_mb,scale_mb = parameters
monthly_beta = scstats.beta.pdf(x,a_m,b_m,loc_mb,scale_mb)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(mmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

##PLOT##
mmax[:, :-1].sort()
occmmax= np.zeros_like(mmax)
for r in range(len(mmax)):
    occmmax[r] = ((r+1)/(len(mmax)+1))

```

```

In [ ]: if len(params) == 2:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
    if len(params) == 3:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
    if len(params) == 4:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
        monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],size=1)
    if len(params) == 5:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params
    if len(params) == 6:
        monthly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],para
        monthly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params
    print(params)

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
        plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
        plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
        plt.xlim(1250,1700);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');
        plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bins
        plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
        plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
        plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
        plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
        plt.plot(monthly_norm,label='Normal fit',linestyle='--')
        plt.plot(monthly_beta,label='Beta',linestyle=':',color='black')
        plt.xlim(1250,1700);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf,color='Red',label='Bestfit -'+str(name))
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.plot(monthly_norm,label='Normal fit',linestyle='--')

```



```

plt.plot(monthly_beta, label='Beta', linestyle=':', color='black')
plt.title('Monthly maxima distribution');
plt.xlim(1400,1800);plt.ylim(0,0.0015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6))
plt.scatter(mmax, occmmax, color='green', facecolors='none', s=50, label='Data');plt.ysca
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+s
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str(
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
if len(params) == 5:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param

plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange'
plt.plot(1-scstats.norm.cdf(x,loc_mn,scale_mn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_m,b_m,loc_mb,scale_mb),label='Beta fit',linestyle=':

plt.title('ICDF Monthly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exce
plt.show()

```

```
In [ ]: dfdf = pd.DataFrame(mmax)
dfdf.to_csv('MMAX_FINAL.csv')
```

```
In [ ]: mu_m,beta_m
```

-----YEARLY-----

```
In [ ]: years = 1000
yrmax = np.zeros(years)

for k in range(years):
    ytemp = np.zeros(12) #1jaar heeft 12 maanden
    for j in range(12):#1 jaar heeft 12 maanden
        d = np.random.gumbel(mu_m,beta_m,size=1)
        ytemp[j] = d
    yrmax[k] = np.max(d)
```

```
In [ ]: ##PLOT##
yrmax[:, :-1].sort()
occyrmax= np.zeros_like(yrmax)
for r in range(len(yrmax)):
    occyrmax[r] = ((r+1)/(len(yrmax)+1))

###Berekenen parameters GumbelR + Weib + genextreme + normal###
parameters = scstats.weibull_min.fit(yrmax,loc=0,scale=1)
c_yr = parameters[0]; loc_yr = parameters[1]; scale_yr = parameters[2]
yearly_weib = weibull(x,c_yr,loc_yr,scale_yr)

parameters = scstats.gumbel_r.fit(yrmax)
mu_yr = parameters[0]
beta_yr = parameters[1]
yearly_gumb = gumbel(x,mu_yr,beta_yr)
```



```

parameters = scstats.genextreme.fit(yrmax)
c_yrg = parameters[0]; loc_yrg = parameters[1]; scale_yrg = parameters[2]
yearly_gev = scstats.genextreme.pdf(x,c_yrg,loc_yrg,scale_yrg)

parameters = scstats.norm.fit(yrmax)
loc_yrn = parameters[0]; scale_yrn = parameters[1]
yearly_norm = scstats.norm.pdf(x,loc_yrn,scale_yrn)

parameters = scstats.beta.fit(yrmax)
a_yr,b_yr,loc_yrb,scale_yrb = parameters
yearly_beta = scstats.beta.pdf(x,a_yr,b_yr,loc_yrb,scale_yrb)

from distfit import distfit
dist = distfit(distr=dist_names);
bestfit = dist.fit_transform(yrmax);
model = (bestfit.get('model'))
name = (model.get('name'))
params = model.get('params')
bestfit = (model.get('distr'))

```

```

In [ ]: if len(params) == 2:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1])
        yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],size=1)
    if len(params) == 3:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2])
        yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],size=1)
    if len(params) == 4:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3])
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3])
    if len(params) == 5:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_rvs = bestfit.pdf(params[0],params[1],params[2],params[3],params[
    if len(params) == 6:
        yearly_bestfit_pdf = bestfit.pdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_cdf = bestfit.cdf(x,params[0],params[1],params[2],params[3],param
        yearly_bestfit_rvs = bestfit.rvs(params[0],params[1],params[2],params[3],params[

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.xlim(1200,2000);plt.ylim(0,0.01);plt.legend(loc='upper right');plt.show()

```

```

In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Data - bin
if len(params) == 2:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 3:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 4:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 5:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
if len(params) == 6:
    plt.plot(yearly_bestfit_pdf,color='Red',label='Bestfit -'+""+str(name))
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.xlim(1200,2000);plt.ylim(0,0.01);plt.legend(loc='upper right');plt.show()

```

```

##---##

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
if len(params) == 2:
    plt.plot(bestfit.pdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str
if len(params) == 3:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2]),color='Red',label='Bestfit
if len(params) == 4:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3]),color='Red',labe
if len(params) == 5:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4]),color=
if len(params) == 6:
    plt.plot(bestfit.pdf(x,params[0],params[1],params[2],params[3],params[4],params[
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');plt.hist(yrmax,density=True
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.plot(yearly_norm,label='Normal fit',linestyle='--')
plt.plot(yearly_beta,label='Beta',linestyle=':',color='black')
plt.title('Yearly maxima distribution');
plt.xlim(1650,1800);plt.ylim(0,0.003);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Data');plt.ys
if len(params) == 2:
    plt.plot(1-bestfit.cdf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+s
    plt.plot(bestfit.sf(x,params[0],params[1]),color='Red',label='Bestfit -'+""+str(
if len(params) == 3:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2]),color='Red',label='Bestf
if len(params) == 4:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3]),color='Red',la
if len(params) == 5:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4]),colo
if len(params) == 6:
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param
    plt.plot(1-bestfit.cdf(x,params[0],params[1],params[2],params[3],params[4],param

plt.plot(x,(1-scstats.weibull_min.cdf(x,c_yr,loc_yr,scale_yr)),label='Weibull fit',c
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_yr,beta_yr)),label='Gumbel fit',color='Orang
plt.plot(1-scstats.norm.cdf(x,loc_yrn,scale_yrn),label='Normal fit',linestyle='--')
plt.plot(1-scstats.beta.cdf(x,a_yr,b_yr,loc_yrb,scale_yrb),label='Beta fit',linestyl

plt.title('ICDF Yearly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Excee
plt.show()

```

```
In [ ]: mu_yr,beta_yr
```

```
In [ ]: plt.figure(figsize=(10,10));plt.title('Daily, monthly, yearly maxima distribution');
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Daily');
plt.plot(daily_bestfit_pdf,color='green',label='Daily fit',lw=2)
plt.hist(mmax,density=True,bins=25,color='orange',alpha=0.3,ec="k",label='Monthly');
plt.plot(monthly_gumb,color='orange',label='Monthly fit',lw=2)
plt.hist(yrmax,density=True,bins=25,color='blue',alpha=0.3,ec="k",label='Yearly');
plt.plot(yearly_bestfit_pdf,color='blue',label='Yearly fit',lw=2,linestyle='--')

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');

plt.xlim(600,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

```

```
In [ ]: plt.hist(yrmax,density=True)
plt.plot(x,yearly_norm);plt.xlim(1000,1500);
```

```
In [ ]: dfdf = pd.DataFrame(yrmax)
dfdf.to_csv('YRMAX_FINAL.csv')
```

-----PLOT FOR REPORT-----

```
In [ ]: plt.figure(figsize=(8,6));plt.title('Daily maxima distribution');plt.xlabel('Bending
plt.hist(dmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='Daily maxim
plt.plot(daily_bestfit_pdf,color='Green',label='Double Weibull')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.xlim(650,1400);plt.ylim(0,0.008);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));

plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(daily_bestfit_pdf,color='Green',label='Double Weibull')
plt.plot(x,daily_weib,label='Weibull fit',color='blue');plt.hist(dmax,density=True,b
plt.plot(x,daily_gumb,color='orange',label='Gumbel fit')
plt.title('Daily maxima distribution');plt.legend(loc='upper right')
plt.xlim(1100,1200);plt.ylim(0,0.0008);

plt.figure(figsize=(12,6))
plt.scatter(dmax,occdmax,color='green',facecolors='none',s=50,label='Data');plt.ysca
plt.plot(1-daily_bestfit_cdf,color='Green',label='Double Weibull')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_d,loc_d,scale_d)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_d,beta_d)),label='Gumbel fit',color='Orange'

plt.title('ICDF Daily maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceed
plt.show()
```

\_d\_\_MONTHLY\_\_

```
In [ ]: plt.figure(figsize=(8,6));plt.title('Monthly maxima distribution');plt.xlabel('Bendi
plt.hist(mmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='monthly max
plt.plot(monthly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.xlim(1150,1500);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(monthly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,monthly_weib,label='Weibull fit',color='blue');plt.hist(mmax,density=True
plt.plot(x,monthly_gumb,color='orange',label='Gumbel fit')
plt.title('Monthly maxima distribution');plt.legend(loc='upper right')
plt.xlim(1350,1500);plt.ylim(0,0.003);

plt.figure(figsize=(12,6))
plt.scatter(mmax,occmmax,color='green',facecolors='none',s=50,label='Monthly maxima'
plt.plot(1-monthly_bestfit_cdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_m,loc_m,scale_m)),label='Weibull fit',colo
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_m,beta_m)),label='Gumbel fit',color='Orange'

plt.title('ICDF Monthly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exce
plt.show()
```

-----YEARLY-----

```
In [ ]: plt.figure(figsize=(8,6));plt.title('Yearly maxima distribution');plt.xlabel('Bendin
plt.hist(yrmax,density=True,bins=25,color='green',alpha=0.3,ec="k",label='yearly max
plt.plot(yearly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
```

```

plt.xlim(1150,1650);plt.ylim(0,0.015);plt.legend(loc='upper right');plt.show()

plt.figure(figsize=(8,6));
plt.xlabel('Bending moment [kNm]'); plt.ylabel('Frequency');
plt.plot(yearly_bestfit_pdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,yearly_weib,label='Weibull fit',color='blue');plt.hist(yrmax,density=True)
plt.plot(x,yearly_gumb,color='orange',label='Gumbel fit')
plt.title('Yearly maxima distribution');plt.legend(loc='upper right')
plt.xlim(1350,1650);plt.ylim(0,0.003);

plt.figure(figsize=(12,6))
plt.scatter(yrmax,occyrmax,color='green',facecolors='none',s=50,label='Yearly maxima')
plt.ylim(1e-6,2);plt.xlim(1150,1700)
plt.plot(1-yearly_bestfit_cdf,color='Green',label='Gumbel fit',linestyle='--')
plt.plot(x,(1-scstats.weibull_min.cdf(x,c_yr,loc_yr,scale_yr)),label='Weibull fit',color='blue')
plt.plot(x,(1-scstats.gumbel_r.cdf(x,mu_yr,beta_yr)),label='Gumbel fit',color='Orange')

plt.title('ICDF Yearly maxima');plt.xlabel('Bending moment [kNm]');plt.ylabel('Exceedance')
plt.show()

```

Naar aanleiding van een korte studie blijkt dat voor de daily maxima de weibull fit goed is, voor de monthly maxima de gumbel fit en voor de yearly maxima ook de gumbel fit.

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

Reading databases. Drop LM1 value.

```
In [ ]: L=10;
M_G_ED = 1.35 * (1/8) * (0.5*1*25)* (L**2);
print(M_G_ED)
#df['McMax [kNm]'] = df['McMax [kNm]'] + M_G_ED
x = np.linspace(0,100e3,100e3)
```

```
In [ ]: ##FUNCTIONS##
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def normal(x,mu,sigma):
    return (1 / (sigma*np.sqrt(2*np.pi))) * np.exp(-0.5*((x-mu)/sigma)**2)
def student_t(x,v):
    return scstats.t.pdf(x,v)
def lognorm_sample(mu,sigma):
    return scstats.lognorm.rvs(s=sigma,scale=np.exp(mu))
def gumbel_sample(mu,beta):
    return scstats.gumbel_r.rvs(loc=mu,scale=beta)
def lognorm(x,mu,sigma):
    return scstats.lognorm.pdf(x=x,s=sigma,scale=np.exp(mu))
def student_t_sample(v):
    return scstats.t.rvs(v)
def normal_sample(mu,sigma):
    return scstats.norm.rvs(loc=mu,scale=sigma)
def weibull(x,shape,loc,scale):
    return scstats.weibull_min.pdf(x,c=shape,loc=loc,scale=scale)
def gumbel(x,mu,beta):
    z = (x- mu)/beta
    return (1/beta)*np.exp(-(z+np.exp(-z)))
def invgumbelpdf(x,mu,beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))
```

Based on notebook 3: gumbel\_r is the best fitted distribution.

```
In [ ]: yrmax = pd.read_csv('YRMAX_FINAL.csv',index_col=0,low_memory=False)
yrmax = yrmax.to_numpy();parameters = scstats.gumbel_r.fit(yrmax)
mu_yr,beta_yr = parameters;
```

The part below shows the resistance of a concrete beam.

```
In [ ]: ###JCSS CONC###
x = np.linspace(0,100e3,100000)
m = 3.85;v=10;s=0.09;n=3
mu_y1 = 1; sigma_y1=0.06
e=100000
fc_arr = np.zeros(e);ft_arr=np.zeros(e);ec_arr = np.zeros(e);eps_arr = np.zeros(e);
```

```

for r in range(e):
    fc_arr[r] = (np.exp(m+student_t_sample(v)*s*(1+(1/n))**0.5))
    ft_arr[r] = 0.3*(fc_arr[r]**(2/3))
    ec_arr[r] = 10.5*(fc_arr[r]**(1/3))/(1+0.7*2.5)
    eps_arr[r] = 6e-3 * (fc_arr[r]**(-1/6))*(1+0.7*2.5)

##Fitting dist##
fcp = scstats.lognorm.fit(fc_arr,loc=0,scale=1);
ftp =scstats.lognorm.fit(ft_arr,loc=0,scale=1);
ecp = scstats.lognorm.fit(ec_arr,loc=0,scale=1);
eps_p = scstats.lognorm.fit(eps_arr);

##JCSS Reinforcing steel + Moment resistance##
def F_steel(Sxx,n,d): #F_steel in kN
    X12 = normal_sample(0,22)
    X13 = normal_sample(0,8)
    mu1 = Sxx + 2*30
    mu11 = mu1 * ((0.87 + 0.13*np.exp(-0.08*d))**-1)
    X11 = normal_sample(mu11,19)
    fyd = X11 + X12 + X13

    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)

    return fyd * As_sample / 1e3 , fyd

def F_conc(Sxx,n,d): #F_conc in kN
    As = n * (np.pi*(d**2) /4)
    As_sample = normal_sample(As,0.06*As)
    xc = (As_sample)*F_steel(Sxx,n,d)[1] / (0.85*lognorm_sample(np.log(fcp[2]),(fcp[
    Fcd = 0.85*xc * lognorm_sample(np.log(fcp[2]),fcp[0])
    return Fcd/1e3

def M_res(Sxx,n,d,z): #M_res in kNm
    fs = F_steel(Sxx,n,d)[0]
    fc = F_conc(Sxx,n,d)
    focc = np.minimum(fs,fc)
    mres = focc*z
    return mres/1e3

```

The part below describes the model uncertainties. Based on JCSS PMC3

```

In [ ]: sigma_load = lognorm_sample(np.log(1),0.1)
        sigma_res = lognorm_sample(np.log(1.2),0.15)

```

Part below is the MC simulation. Please note: this *does* include M\_G\_ED.

```

In [ ]: def MCsim(Sxx,n,d,z,e):
        print('Running a Monte-Carlo simulation for',e/1e6,'million times.')
        loadarr = np.zeros(e)
        resarr = np.zeros(e)
        zarr = np.zeros(e)
        zarr2 = np.zeros(e)

        with tqdm(total=e, file=sys.stdout) as pbar:
            for r in range(e):
                pbar.set_description('MC-simulation')
                pbar.update(1)
                loadarr[r] = (scstats.gumbel_r.rvs(mu_yr,beta_yr,size=1)+M_G_ED)*lognorm
                resarr[r] = M_res(Sxx,n,d,z)*lognorm_sample(np.log(1.2),0.15)
                zarr[r] = loadarr[r]/resarr[r]
                zarr2[r] = resarr[r] - loadarr[r]

```

```
##ICDF plot##  
zarr[::-1].sort()  
occzarr= np.zeros_like(zarr)  
for r in range(len(zarr)):  
    occzarr[r] = ((r+1)/(len(zarr)+1))  
  
return (loadarr, resarr, zarr, occzarr, zarr2)
```

```
In [ ]: f= Msim(500,17,32,405,1000000)
```

```
In [ ]: dfdf = pd.DataFrame(f)  
dfdf.to_csv('CASE_STUDY_WIM_FINAL.csv')
```

```
In [ ]: zarr = f[2]
```

```
In [ ]: len(zarr[zarr>1])
```

```
In [ ]:
```



## File 5 - Case study analysis

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
#pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

```
In [ ]: df = pd.read_csv('CASE_STUDY_L10_FINAL_V2.csv', low_memory=False, index_col=0)
df = df.to_numpy()
loadarr, resarr, zarr, occzarr, zarr2 = df
```

```
In [ ]: x = np.linspace(0, 10e3, 100e3)
##PLOTS##
plt.figure(figsize=(16,8))
plt.scatter(resarr, loadarr, facecolors='none', color='green', marker='o', s=1, label='Sim')
plt.xlim(0, 8000); plt.ylim(0, 4000); plt.xlabel('Resistance [kNm]'); plt.ylabel('Be

plt.figure(figsize=(16,8))
plt.hist(loadarr, bins=50, alpha=0.5, label='S(x)', density=True, color='green', ec="k") ;
plt.hist(resarr, bins=50, alpha=0.5, label='R(x)', density=True, color='red', ec="k");
plt.hist(zarr2, bins=50, alpha=0.5, label='R(x)-S(x)', density=True, ec="k", color='blue')
plt.legend(loc='upper right'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Probabi
plt.show()

plt.figure(figsize=(16,8))
plt.scatter(zarr, occzarr, color='green', facecolors='none', s=50, label='Simulation'); pl
plt.title('ICDF MC-sim'); plt.xlabel('Load / Resistance [-]'); plt.ylabel('Exceedance
plt.show()

#calculate prob of failure
x_pof = zarr; y_pof = occzarr;
l = len(x_pof[x_pof > 1]) ; ecdf = ECDF(zarr);

print("")
pof = (len(zarr[zarr>1]))/len(zarr)
#beta = (1-np.mean(zarr))/(scstats.weibull_min.std(c=params[0], loc=params[1], scale=p
print('The probability of failure is', "{:.2E}".format(Decimal(pof)));
#print('The reliability index is', beta) #CIE4130
```

```
In [ ]: plt.figure(figsize=(16,8))
plt.hist(loadarr, bins=50, alpha=0.5, label='S(x)', density=True, color='green', ec="k") ;
plt.hist(resarr, bins=50, alpha=0.5, label='R(x)', density=True, color='red', ec="k");
#plt.hist(zarr2, bins=50, alpha=0.5, label='R(x)-S(x)', density=True, ec="k", color='blue')
plt.legend(loc='upper right'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Probabi
plt.title('Distributions for S(x) and R(x) with uncertainty factor')
plt.show()
```

```
In [ ]: def phi(pof):
    factor = 1 / (np.sqrt(2*np.pi))
    return np.exp(-(pof**2)/2)*factor
def integrate(pof):
    return sc.integrate.quad(phi, float("-inf"), pof)[0]
def poffunc(beta):
```

```
    return integrate(-beta) #gives Prob of failure
for r in range(1,100000):
    step = 1/10000
    pof1 = poffunc(5-r*step)
    if 5 - r*step > 0:
        if 0.99990 < (pof1/pof) < 1.0005:
            print('beta:',5-r*step,'||','pof',pof1)
            break
```

```
In [ ]: print(len(zarr[zarr>1]))
```

# Case study analysis LP data

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
#pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats

In [ ]: df = pd.read_csv('CASE_STUDY_LP_FINAL.csv', low_memory=False, index_col=0)
df = df.to_numpy()

In [ ]: loadarr, resarr, zarr, occzarr, zarr2 = df
zarr[:, -1].sort()
occzarr = np.zeros_like(zarr)
for r in range(len(zarr)):
    occzarr[r] = ((r+1)/(len(zarr)+1))

In [ ]: df2 = zarr, occzarr
np.savetxt('LP_icdf.csv', df2)

In [ ]: x = np.linspace(0, 10e3, 100e3)
##PLOTS##
plt.figure(figsize=(16, 8))
plt.scatter(resarr, loadarr, facecolors='none', color='green', marker='o', s=1, label='Sim')
plt.xlim(0, 8000); plt.ylim(0, 4000); plt.xlabel('Resistance [kNm]'); plt.ylabel('Be

plt.figure(figsize=(16, 8))
plt.hist(loadarr, bins=50, alpha=0.5, label='S(x)', density=True, color='green', ec="k") ;
plt.hist(resarr, bins=50, alpha=0.5, label='R(x)', density=True, color='red', ec="k");
#plt.hist(zarr2, bins=50, alpha=0.5, label='R(x)-S(x)', density=True, ec="k", color='blue')
plt.legend(loc='upper right'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Probabi
plt.title('Monte-Carlo simulation $1^6$ simulations')
plt.show()

plt.figure(figsize=(16, 8))
plt.scatter(zarr, occzarr, color='green', facecolors='none', s=50, label='Simulation'); pl
plt.title('ICDF MC-sim'); plt.xlabel('Load / Resistance [-]'); plt.ylabel('Exceedance
plt.show()

print("")
pof = (len(zarr[zarr>1]))/len(zarr)
#beta = (1-np.mean(zarr))/(scstats.weibull_min.std(c=params[0], loc=params[1], scale=p
print('The probability of failure is', "{:.2E}".format(Decimal(pof)));
#print('The reliability index is', beta) #CIE4130

In [ ]: def phi(pof):
    factor = 1 / (np.sqrt(2*np.pi))
    return np.exp(-(pof**2)/2)*factor
def integrate(pof):
    return sc.integrate.quad(phi, float("-inf"), pof)[0]
def poffunc(beta):
    return integrate(-beta) #gives Prob of failure
for r in range(1, 100000):
```

```
step = 1/10000
pof1 = poffunc(5-r*step)
if 5 - r*step > 0:
    if 0.99990 < (pof1/pof) < 1.0005:
        print('beta:',5-r*step,'||','pof',pof1)
else:
    break
```

# Case study WIM analysis

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
import random as random
import matplotlib.pyplot as plt
#pd.set_option('display.max_columns', None)
from decimal import Decimal
from statsmodels.distributions.empirical_distribution import ECDF
import sys
from tqdm import tqdm
import scipy.stats as scstats
```

```
In [ ]: df = pd.read_csv('CASE_STUDY_WIM_FINAL.csv', low_memory=False, index_col=0)
df = df.to_numpy()
```

```
In [ ]: loadarr, resarr, zarr, occzarr, zarr2 = df
zarr[:, -1].sort()
occzarr = np.zeros_like(zarr)
for r in range(len(zarr)):
    occzarr[r] = ((r+1)/(len(zarr)+1))
```

```
In [ ]: x = np.linspace(0, 10e3, 100e3)
##PLOTS##
plt.figure(figsize=(16,8))
plt.scatter(resarr, loadarr, facecolors='none', color='green', marker='o', s=1, label='Sim')
plt.xlim(0, 8000); plt.ylim(0, 4000); plt.xlabel('Resistance [kNm]'); plt.ylabel('Be

plt.figure(figsize=(16,8))
plt.hist(loadarr, bins=50, alpha=0.5, label='S(x)', density=True, color='green', ec="k") ;
plt.hist(resarr, bins=50, alpha=0.5, label='R(x)', density=True, color='red', ec="k");
#plt.hist(zarr2, bins=50, alpha=0.5, label='R(x)-S(x)', density=True, ec="k", color='blue')
plt.legend(loc='upper right'); plt.xlabel('Bending moment [kNm]'); plt.ylabel('Probabi
plt.title('Monte-Carlo simulation $1^6$ simulations')
plt.show()

plt.figure(figsize=(16,8))
plt.scatter(zarr, occzarr, color='green', facecolors='none', s=50, label='Simulation'); pl
plt.title('ICDF MC-sim'); plt.xlabel('Load / Resistance [-]'); plt.ylabel('Exceedance
plt.show()

print("")
pof = (len(zarr[zarr>1]))/len(zarr)
#beta = (1-np.mean(zarr))/(scstats.weibull_min.std(c=params[0], loc=params[1], scale=p
print('The probability of failure is', "{:.2E}".format(Decimal(pof)));
#print('The reliability index is', beta) #CIE4130
```

```
In [ ]: def phi(pof):
    factor = 1 / (np.sqrt(2*np.pi))
    return np.exp(-(pof**2)/2)*factor
def integrate(pof):
    return sc.integrate.quad(phi, float("-inf"), pof)[0]
def poffunc(beta):
    return integrate(-beta) #gives Prob of failure
for r in range(1, 100000):
    step = 1/10000
    pof1 = poffunc(5-r*step)
    if 5 - r*step > 0:
```

```
if 0.99990 < (pof1/pof) < 1.0005:  
    print('beta:',5-r*step,'||','pof',pof1)  
else:  
    break
```

```
In [ ]: print(len(zarr[zarr>1]))
```

```
In [ ]: import pandas as pd
import numpy as np
import scipy as sc
pd.set_option('display.max_columns', None)
import random
import math as math
```

In onderstaande cellen wordt  $M_{Ed}$  tgv WIM berekend adv hetzelfde script dat ik heb gebruikt om de  $M_{Ed}$  tgv Theoretical te bepalen.

```
In [ ]: df_wim = pd.read_csv('WIM - OA_FIL_ROTT_DATA.csv')
df_wim2 = pd.read_csv('WIM - BK_FIL_ROTT_DATA.csv');df_wim2 = df_wim2.fillna(0)
df_wim = df_wim.fillna(0)
```

```
In [ ]: def Y_WIM(L): #gebruik j om len(df) aan te duiden
x = np.linspace(0,L+25,(1+(L+25)*10)) #Lengtes toevoegen om alle aslasten op te
y = np.zeros(len(x)) #Lege array maken
y1 = x # y1 = x
y2 = y1 - df_wim.iloc[j,16]/100 # y2 = y1 - w1. Wordt gedaan om de onderstaande
y = np.vstack((y1,y2))
for r in range(9): #10 w's vullen. Eindigt op kolom 26
c = y[r+1] - df_wim.iloc[j,r+17]/100
y = np.vstack((y,c))
y = np.where(y<0,0,y) #negatieve waarden verwijderen
y = np.where(y>L,0,y) #waarden groter dan L verwijderen
return y
```

In onderstaande cel wordt  $M_{Ed}$  berekend voor WIM-data. **Let op!** Het script is aangepast op de WIM-data.

```
In [ ]: L=10
```

```
In [ ]: McMax = np.zeros(len(df_wim))
F_matrix = (df_wim.iloc[:, 3: 14]*1e2*(10/1000)).to_numpy() #selecteren van alle bel

for j in range(len(df_wim)): #Len (df_wim)
y_matrix = Y_WIM(L) #y1 = y_matrix[1]

Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

#Av
for r in range(11): #in totaal 11 F's, dus 11 Av.
Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
#Mc
for r in range(11):
Mc[r] = np.where(y_matrix[r] < (L/2),
(Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
Av[r]*(L/2))
Mc[r] = np.where(Mc[r]<0,0,Mc[r])

Mc = Mc.sum(axis=0) #opsommen
Mc = np.amax(Mc)

McMax[j] = Mc
df_wim['McMax [kNm]'] = McMax
```

```
In [ ]: def Y_WIM2(L): #gebruik j om len(df) aan te duiden
x = np.linspace(0,L+25,(1+(L+25)*10)) #Lengtes toevoegen om alle aslasten op te
```



```

y = np.zeros(len(x)) #Lege array maken
y1 = x # y1 = x
y2 = y1 - df_wim2.iloc[j,16]/100 # y2 = y1 - w1. Wordt gedaan om de onderstaande
y = np.vstack((y1,y2))
for r in range(9): #10 w's vullen. Eindigt op kolom 26
    c = y[r+1] - df_wim2.iloc[j,r+17]/100
    y = np.vstack((y,c))
y = np.where(y<0,0,y) #negatieve waarden verwijderen
y = np.where(y>L,0,y) #waarden groter dan L verwijderen
return y

```

```

In [ ]: McMax = np.zeros(len(df_wim2))
F_matrix = (df_wim2.iloc[:, 3: 14]*1e2*(10/1000)).to_numpy() #selecteren van alle be

for j in range(len(df_wim2)): #Len (df_wim)
    y_matrix = Y_WIM2(L) #y1 = y_matrix[1]

    Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
    Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

    #Av
    for r in range(11): #in totaal 11 F's, dus 11 Av.
        Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
    #Mc
    for r in range(11):
        Mc[r] = np.where(y_matrix[r] < (L/2),
                        (Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
                        Av[r]*(L/2))
        Mc[r] = np.where(Mc[r]<0,0,Mc[r])

    Mc = Mc.sum(axis=0) #opsommen
    Mc = np.amax(Mc)

    McMax[j] = Mc
df_wim2['McMax [kNm]'] = McMax

```

```

In [ ]: df_C = pd.read_csv('df_4400C_load.csv', sep=',',low_memory = False)
df_tot = pd.read_csv('df_4400C_load.csv', sep=',',low_memory = False)

```

```

In [ ]: #Verwijderen van LM1 tandemstelsel
indexNames = df_tot[ (df_tot['Soort'] == 'TS')].index
df_tot.drop(indexNames , inplace=True)

```

```

In [ ]: import matplotlib.pyplot as plt
plt.figure(figsize=(15,5), dpi=80)

#plot_wim = plt.scatter(df_wim['GVW']*1e2,df_wim['McMax [kNm]'],alpha = 0.5, label='
plot_theo = plt.scatter(df_C['Max Axle Load [kg]',df_C['McMax [kNm]'],alpha = 0.5,f
plot_wim = plt.scatter(df_wim2['GVW']*1e2,df_wim2['McMax [kNm]'],alpha = 1, facecolo

plt.xlim(10e3,110e3)
plt.ylim(0e3,1.5e3)
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [kg]')
plt.ylabel('Bending moment [kNm]')
plt.title('Bending moment vs GVW for a bridge of 10 meters')
plt.show()

```

```

In [ ]: import matplotlib.pyplot as plt

```

```
plt.figure(figsize=(15,5), dpi=80)

plot_wim = plt.scatter(df_wim['GVW']*1e2,df_wim['McMax [kNm]'],alpha = 0.5, label='W')
plot_theo = plt.scatter(df_C['Max Axle Load [kg]'],df_C['McMax [kNm]'],alpha = 0.5, f

plt.xlim(10e3,110e3)
plt.ylim(0e3,1.5e3)
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [kg]')
plt.ylabel('Bending moment [kNm]')
plt.title('Bending moment vs GVW for a bridge of 10 meters')
plt.show()
```

```
In [ ]: plt.figure(figsize=(5,5), dpi=80)

#plot_wim = plt.scatter(df_wim['GVW']*1e2,df_wim['McMax [kNm]'],alpha = 0.5, label='
plot_theo = plt.scatter(df_C['Max Axle Load [kg]'],df_C['McMax [kNm]'],alpha = 0.5, f
plot_wim = plt.scatter(df_wim2['GVW']*1e2,df_wim2['McMax [kNm]'],alpha = 1, facecolo

plt.xlim(50e3,110e3)
plt.ylim(0e3,1.5e3)
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [kg]')
plt.ylabel('Bending moment [kNm]')
plt.title('Bending moment vs GVW for a bridge of 10 meters')
plt.show()
```

```
In [ ]: mcwim = df_wim['McMax [kNm]']
mcwim = mcwim.drop_duplicates()
mcwim = mcwim.to_numpy();mcwim[:,::-1].sort()
mctheo = df_tot['McMax [kNm]']
mctheo= mctheo.drop_duplicates()
mctheo = mctheo.to_numpy();mctheo[:,::-1].sort()
```

```
In [ ]: occtheo = np.zeros_like(mctheo)
for r in range(len(mctheo)):
    occtheo[r] = (r/(len(mctheo)+1))

occwim = np.zeros_like(mcwim)
for r in range(len(mcwim)):
    occwim[r] = (r/(len(mcwim)+1))
```

```
In [ ]: import scipy.stats as scstats
#Berekenen parameters Weibull
parameters = scstats.exponweib.fit(mctheo,floc=0,f0=1)
nTHEO = parameters[3]
aTHEO = parameters[1]

#Berekenen parameters Weibull
parameters = scstats.exponweib.fit(mcwim,floc=0,f0=1)
nWIM = parameters[3]
aWIM = parameters[1]

#Berekenen parameters GumbelR
parameters = scstats.gumbel_r.fit(mctheo)
muTHEO = parameters[0]
betaTHEO = parameters[1]
```

```
#Berekenen parameters GEV
parameters = scstats.genextreme.fit(mctheo)
```

```
In [ ]: def invweibullpdf(x,n,a):
        return (1-(1 - np.exp(-(x/n)**a)))

def invgumbelpdf(x,mu,beta):
    return (1-(np.exp(-np.exp(-(x-mu)/beta))))
```

```
In [ ]: plt.figure(figsize=(15,5))
plt.scatter(mcwim,occwim, alpha=0.5, label='WIM', c='green')
plt.scatter(mctheo,occtheo,alpha=0.5,label='MEA',c='blue')
plt.plot(np.linspace(0,150e3,10000),invweibullpdf(np.linspace(0,120e3,10000),nTHEO,a)
#plt.plot(np.linspace(0,150e3,10000),invgumbelpdf(np.linspace(0,120e3,10000),muTHEO,
plt.plot(np.linspace(0,150e3,10000),invweibullpdf(np.linspace(0,120e3,10000),nWIM,aw)
#plt.plot(np.linspace(0,150e3,10000),invgumbelpdf(np.linspace(0,120e3,10000),muWIM,b

plt.xlim(0,1.50e3)
plt.ylim(1e-6,1.5)
plt.ylabel('Exceedance frequency')
plt.xlabel('Bending moment [kNm]')
plt.title('Inverted CDF for locations 4400')
plt.yscale("log")
plt.grid(True,which="both",ls="-")
plt.legend(loc="upper right")
plt.show()
```

```
In [ ]: from statsmodels.distributions.empirical_distribution import ECDF
```

```
In [ ]: #ecdf = ECDF(df_B['McMax [kNm]'])
ecdfC = ECDF(df_C['McMax [kNm]'])
#ecdfD = ECDF(df_D['McMax [kNm]'])
#ecdfA = ECDF(df_A['McMax [kNm]'])
ecdfWIM= ECDF(df_wim['McMax [kNm]'])
ecdfWIM2 =ECDF(df_wim2['McMax [kNm]'])
```

```
In [ ]: # plot the cdf
plt.figure(figsize=(12,5))
#plt.plot(ecdfA.x,ecdfA.y,label='4400A')
#plt.plot(ecdf.x, ecdf.y,label='4400B')
plt.plot(ecdfC.x,ecdfC.y,label='4400C')
#plt.plot(ecdfD.x,ecdfD.y,label='4400D')
plt.plot(ecdfWIM.x,ecdfWIM.y,label='WIM OA-Brug')
plt.plot(ecdfWIM2.x,ecdfWIM2.y,label='WIM BK-Brug')
plt.xlabel('Bending moment [kNm]')
plt.ylabel('P(x)')
plt.title('Cumulative distribution function for different locations')
plt.legend()
plt.xlim(0,1500)
plt.ylim(0,1.05)
plt.show()
```

```
In [ ]: df_OA = df_wim
df_BK = df_wim2
plt.figure(figsize=(15,6))
config = np.unique(df_OA['CONFIG'])

for r in range(len(config)):
    l = df_OA[(df_OA['CONFIG'] == config[r]) & (df_OA['GVW'] > 400) & (df_OA['McMax
mmax = np.max(l['McMax [kNm]'])
gvw = l[l['McMax [kNm]'] == mmax]
```

```

gvwmax = np.max(gvw['GVW'])*1e2
if gvwmax > 40e3:

    plt.scatter(gvwmax,mmax,alpha = 1,label=config[r])
    plt.xlim(40e3,110e3);plt.ylim(.75e3,1.4e3);leg = plt.legend(loc='upper right')
    for lh in leg.legendHandles:
        lh.set_alpha(1)

    plt.title('Bending moment vs GVW for a bridge of 10 meters -- WIM_OA')
plt.show()

```

```
In [ ]: df_OA[:2]
```

```
In [ ]: print(len(config))
```

```

In [ ]: #####OA-Bridge#####
config = np.unique(df_OA['CONFIG'])
for r in range(len(config)):
    l = df_OA[(df_OA['CONFIG'] == config[r]) & (df_OA['GVW'] > 400) & (df_OA['McMax']
mmax = np.max(l['McMax [kNm]'])
gvw = l[l['McMax [kNm]'] == mmax]
if mmax > 0:
    gvwmax = np.max(gvw['GVW'])
    f_matrix = gvw.iloc[:,3:14].to_numpy()
    w_matrix = gvw.iloc[:,16:26].to_numpy()
    assen = np.count_nonzero(f_matrix,axis=0).sum() #bepalen van aantal assen
    wsum = w_matrix.sum()
    plt.figure(figsize=(5,5))
    for k in range(assen):
        plt.vlines(x=20+(wsum-w_matrix[0,:k].sum()),ymin=0,ymax=f_matrix[0,k],co
plt.title('||Config'+ " "+str(config[r])+"|| GVW max="+str(gvwmax*1e2)+"
        + "|| McMax="+str(math.ceil(mmax))+ "||")

        plt.ylim(0,260);plt.ylabel('Axle load [kN]')
        plt.xlim(-1,2000);plt.xlabel('Total length [cm]')
    plt.legend(loc='upper right')
plt.show()

```

```

In [ ]: #####BK-Bridge#####
config = np.unique(df_BK['CONFIG'])
for r in range(len(config)):
    l = df_BK[(df_BK['CONFIG'] == config[r]) & (df_BK['GVW'] > 400) & (df_BK['McMax']
mmax = np.max(l['McMax [kNm]'])
gvw = l[l['McMax [kNm]'] == mmax]
if mmax > 0:
    gvwmax = np.max(gvw['GVW'])
    f_matrix = gvw.iloc[:,3:14].to_numpy()
    w_matrix = gvw.iloc[:,16:26].to_numpy()
    assen = np.count_nonzero(f_matrix,axis=0).sum() #bepalen van aantal assen
    wsum = w_matrix.sum()
    plt.figure(figsize=(5,5))
    for k in range(assen):
        plt.vlines(x=20+(wsum-w_matrix[0,:k].sum()),ymin=0,ymax=f_matrix[0,k],co
plt.title('||Config'+ " "+str(config[r])+"|| GVW max="+str(gvwmax*1e2)+"
        + "|| McMax="+str(math.ceil(mmax))+ "||")

        plt.ylim(0,260);plt.ylabel('Axle load [kN]')
        plt.xlim(-1,2000);plt.xlabel('Total length [cm]')
    plt.legend(loc='upper right')
plt.show()

```

```
In [ ]: aa = df_C[(df_C['Soort'] == 'kipper') & (df_C['Axles'] == '5') & (df_C['McMax [kNm]'
```

```
In [ ]: ##PLOT VOOR LP DATA##
f_matrix = aa.iloc[:,0:18].to_numpy()/100
w_matrix = aa.iloc[:,18:36].to_numpy()*100
assen = np.count_nonzero(f_matrix,axis=0).sum() #bepalen van aantal assen
wsum = w_matrix.sum()
plt.figure(figsize=(5,5))
gvwmax = np.max(aa['Max Axle Load [kg]'])
mmax = np.max(aa['McMax [kNm]'])

for k in range(assen):
    plt.vlines(x=20+(wsum-w_matrix[0,:k].sum()),ymin=0,ymax=f_matrix[0,k],color='g',
    plt.title("GVW max="+str(gvwmax)+" kg"+" || l_tot="+str(wsum/100)+"m"
    +" || McMax="+str(math.ceil(mmax))+" ||")

    plt.ylim(0,260);plt.ylabel('Axle load [kN]')
    plt.xlim(-1,2000);plt.xlabel('Total length [cm]')
plt.legend(loc='upper right')
plt.show()
```

```
In [ ]: #df_BK['CONFIG'].value_counts()
print(np.unique(df_BK['CONFIG']))
```

```
In [ ]: #####BK-Bridge#####
config = np.unique(df_BK['CONFIG'])
for r in range(len(config)):
    l = df_BK[(df_BK['CONFIG'] == config[r]) & (df_BK['GVW'] > 400) & (df_BK['McMax']
    mmax = np.min(l['McMax [kNm]'])
    gvw = l[l['McMax [kNm]'] == mmax]
    if mmax > 0:
        gvwmax = np.min(gvw['GVW'])
        f_matrix = gvw.iloc[:,3:14].to_numpy()
        w_matrix = gvw.iloc[:,16:26].to_numpy()
        assen = np.count_nonzero(f_matrix,axis=0).sum() #bepalen van aantal assen
        wsum = w_matrix.sum()
        plt.figure(figsize=(5,5))
        for k in range(assen):
            plt.vlines(x=20+(wsum-w_matrix[0,:k].sum()),ymin=0,ymax=f_matrix[0,k],co
            plt.title(' || Config'+ " "+str(config[r])+" || GVW max="+str(gvwmax*1e2)+"
            +" || McMax="+str(math.ceil(mmax))+" ||")

            plt.ylim(0,260);plt.ylabel('Axle load [kN]')
            plt.xlim(-1,2000);plt.xlabel('Total length [cm]')
            plt.legend(loc='upper right')
            plt.show()
```

```
In [ ]: df_wim = pd.concat([df_BK,df_OA])
df_C['Axle 04'] = df_C['Axles']
df_C.drop(df_C.tail(1).index,inplace=True)
```

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111) | (df_wim['CONFIG'] == 211) | (df_wim['CONFIG']
(df_wim['CONFIG'] == 1111) | (df_wim['CONFIG'] == 1211 ) | (df_wim['CONFIG'
```

```
In [ ]: ##Bepalen van 04 axles##
df_C['Axles'] = df_C['Axles'].astype(int)
for r in range(len(df_C)):
    df_C.iloc[r,48] = np.count_nonzero(df_C.iloc[r,:18].to_numpy()) - df_C.iloc[r,40
```

```
In [ ]: wim2semilist = np.array([111,211,311,411,1111,1211,11111,111111])
```

## Configuratie 111

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 1) & (df_C['Axles'] == 2) & (df_C['Soort'] == 'oplegge
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[0]
aa = df_wim[df_wim['CONFIG'] == config]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX1 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[0]
aa = df_wim[df_wim['CONFIG'] == config]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[0]
aa = df_wim[df_wim['CONFIG'] == config]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config))
plt.show()
```

## Configuraties 211 & 1111

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 2) & (df_C['Axles'] == 2) & (df_C['Soort'] == 'oplegge
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX1 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
```

```

config = wim2semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config))
plt.show()

```

```

In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(con
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX4 for axle configuration'+ " "+str(config))
plt.show()

```

## Configuration 311 & 1121 & 11111

```

In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==2) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim2semilist[2]
config1 = wim2semilist[5]
config2 = wim2semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX1 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()

```

```

In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[2]
config1 = wim2semilist[5]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()

```

```

In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[2]
config1 = wim2semilist[5]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()

```

```

In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[2]
config1 = wim2semilist[5]

```



```
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(con
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX4 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[2]
config1 = wim2semilist[5]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW5'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW5'+ " " + str(con
plt.hist(bb['F5 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW5'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX5 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()
```

## Configuration 11111 & 411

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==2) & (df_C['Soort'] == 'oplegger
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[3]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX5 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[3]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[3]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[3]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(con
```

```
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX4 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[3]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW5'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW5'+ " " + str(con
plt.hist(bb['F5 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW5'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX5 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim2semilist[3]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW6'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW6'+ " " + str(con
plt.hist(bb['F6 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW6'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX6 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

## 3 axled trucks

### Configuration 121

```
In [ ]: wim3semilist = np.array([121,221,321,421,1121,1221,11121,111121,2111])
bb = df_C[(df_C['Axle 04'] == 1) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim3semilist[0]
config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX1 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim3semilist[0]
#config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim3semilist[0]
#config1 = wim2semilist[7]
```

```
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bins=np.arange(0,250,5)
config = wim3semilist[0]
#config1 = wim2semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(con
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX4 for axle configuration'+ " "+str(config))
plt.show()
```

## Configuration 221 & 1121

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 2) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX1 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 2) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 2) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 2) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[1]
config1 = wim2semilist[4]
```

```

aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(con
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX4 for axle configuration'+ " "+str(config))
plt.show()

```

```

In [ ]: bb = df_C[(df_C['Axle 04'] == 2) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[1]
config1 = wim2semilist[4]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW5'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW5'+ " " + str(con
plt.hist(bb['F5 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW5'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX5 for axle configuration'+ " "+str(config))
plt.show()

```

## Configuration 321 & 1221 & 11121

```

In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[2]
config1 = wim3semilist[5]
config2 = wim3semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(con
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX1 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()

```

```

In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[2]
config1 = wim3semilist[5]
config2 = wim3semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(con
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX2 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()

```

```

In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger
bins=np.arange(0,250,5)
config = wim3semilist[2]
config1 = wim3semilist[5]
config2 = wim3semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX3 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()

```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[2]
config1 = wim3semilist[5]
config2 = wim3semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(config)
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX4 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[2]
config1 = wim3semilist[5]
config2 = wim3semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW5'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW5'+ " " + str(config)
plt.hist(bb['F5 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW5'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AX5 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 3) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[2]
config1 = wim3semilist[5]
config2 = wim3semilist[6]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW6'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW6'+ " " + str(config)
plt.hist(bb['F6 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW6'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW6 for axle configuration'+ " "+str(config)+"&"+str(config1)+"&
plt.show()
```

## Configuration 421 & 11121

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW1'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW1'+ " " + str(config)
plt.hist(bb['F1 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW1'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW1 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW2'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW2'+ " " + str(config)
plt.hist(bb['F2 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW2'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
```



```
plt.title('Axle load AXW2 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW3'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW3'+ " " + str(con
plt.hist(bb['F3 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW3'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW3 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW4'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW4'+ " " + str(con
plt.hist(bb['F4 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW4'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW4 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW5'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW5'+ " " + str(con
plt.hist(bb['F5 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW5'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW5 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW6'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW6'+ " " + str(con
plt.hist(bb['F6 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW6'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW6 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: bb = df_C[(df_C['Axle 04'] == 4) & (df_C['Axles'] ==3) & (df_C['Soort'] == 'oplegger')
bins=np.arange(0,250,5)
config = wim3semilist[3]
config1 = wim3semilist[7]
aa = df_wim[(df_wim['CONFIG'] == config) | (df_wim['CONFIG'] == config1)]
plt.hist(aa['AXW7'],bins=bins,color='blue',alpha=0.7,label=('WIM AXW7'+ " " + str(con
plt.hist(bb['F7 [kg]']/100, bins=bins, color='red',alpha=0.7,label=('LP AXW7'))
plt.legend(loc='upper right');
plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle load AXW7 for axle configuration'+ " "+str(config)+"&"+str(config1))
plt.show()
```

```
In [ ]: #create df column with total amount of axles per vehicle#
axles = df_wim.iloc[:,3:14].to_numpy()
axarr = np.zeros(len(df_wim))
for r in range(len(df_wim)):
    axarr[r] = np.count_nonzero(axles[r])
df_wim['Axles'] = axarr
```

## 2-AXLED SEMI-TRUCKS +1-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111) | (df_wim['CONFIG'] == 1121) |
                  (df_wim['CONFIG'] == 311) | (df_wim['CONFIG'] == 11111) |
                  (df_wim['CONFIG'] == 411) | (df_wim['CONFIG'] == 111111) |
                  (df_wim['CONFIG'] == 1111)]
bb = df_C[(df_C['Soort'] == 'opleggetrekket') & (df_C['Axles'] == 2)]
aa=aa[aa['Axles'] ==3]
bb=bb[bb['Axle 04'] == 1]
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]'])/100;mean2 = np.mean(bb['F2 [kg]'])/100;mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');plt
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))
```

```
In [ ]: plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 1-axled O4')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()
```

## 2-AXLED SEMI-TRUCK +2-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111) | (df_wim['CONFIG'] == 1121) |
                  (df_wim['CONFIG'] == 311) | (df_wim['CONFIG'] == 11111) |
                  (df_wim['CONFIG'] == 411) | (df_wim['CONFIG'] == 111111) |
                  (df_wim['CONFIG'] == 1111)]
bb = df_C[(df_C['Soort'] == 'opleggetrekket') & (df_C['Axles'] == 2)]
aa=aa[aa['Axles'] ==4]
bb=bb[bb['Axle 04'] == 2]
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]'])/100;mean2 = np.mean(bb['F2 [kg]'])/100;mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
```



```
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 2-axled O4')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 2-axled O4')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 2-axled O4')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()
```

## 2-AXLED SEMI-TRUCK +3-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111) | (df_wim['CONFIG'] == 1121) |
                 (df_wim['CONFIG'] == 311) | (df_wim['CONFIG'] == 11111) |
                 (df_wim['CONFIG'] == 411) | (df_wim['CONFIG'] == 111111) |
                 (df_wim['CONFIG'] == 1111)]
bb = df_C[(df_C['Soort'] == 'opleggertrekker') & (df_C['Axles'] == 2)]
aa=aa[aa['Axles'] ==5]
bb=bb[bb['Axle O4'] == 3]
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 3-axled O4')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 3-axled O4')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 3-axled O4')
```

```

print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-DT45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 3-axled 04')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

```

## 2-AXLED SEMI-TRUCKS + 4-O4

```

In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111) | (df_wim['CONFIG'] == 1121) |
                (df_wim['CONFIG'] == 311) | (df_wim['CONFIG'] == 11111) |
                (df_wim['CONFIG'] == 411) | (df_wim['CONFIG'] == 111111) |
                (df_wim['CONFIG'] == 1111)]
bb = df_C[(df_C['Soort'] == 'opleggertrekker') & (df_C['Axles'] == 2)]
aa=aa[aa['Axles'] ==6]
bb=bb[bb['Axle 04'] == 4]

```

```

In [ ]: mean1 = np.mean(aa['AXW1'])
mean2 = np.mean(aa['AXW2'])
mean3 = np.mean(aa['AXW3'])
mean4 = np.mean(aa['AXW4'])
mean5 = np.mean(aa['AXW5'])
mean6 = np.mean(aa['AXW6'])

print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)
print((aa['AXW6'] < mean6).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))

```

```

In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX5'));
plt.hist(aa['AXW6'],bins=bins,alpha=0.7,label=('WIM-AX6'));

plt.legend(loc='upper right');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 2-axled semi-trucks, 4-axled 04');plt.xlim(0,200);#plt.yli
plt.show()

```

```

In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 2-axled 04')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');

```

```

plt.title('IAD for 2-axled ST, 2-axled O4')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 2-axled O4')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-DT45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 3-axled O4')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

plt.hist(aa['DT56']/100, bins=20,alpha=0.7,label='WIM-DT56');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 2-axled ST, 3-axled O4')
print(np.mean(aa['DT56']/100),np.min(aa['DT56']/100),np.max(aa['DT56']/100))
plt.show()

```

## 3-AXLED SEMI-TRUCKS + 1-O4

```

In [ ]: aa = df_wim[(df_wim['CONFIG'] == 121) | (df_wim['CONFIG'] == 122) |
                (df_wim['CONFIG'] == 123) | (df_wim['CONFIG'] == 124) |
                (df_wim['CONFIG'] == 1211) | (df_wim['CONFIG'] == 1221) |
                (df_wim['CONFIG'] == 12111) | (df_wim['CONFIG'] == 121111)]
bb = df_C[(df_C['Soort'] == 'opleggertrekker') & (df_C['Axles'] == 3)]

```

```

In [ ]: plt.figure(figsize=(6,6))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax3=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 3-axled semi-trucks');plt.xlim(0,250);plt.ylim(0,100)
plt.show()

```

```

In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 1-axled O4')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 1-axled O4')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

```

```
plt.hist(aa['DT34']/100, bins=20, alpha=0.7, label='WIM-DT34');
plt.hist(bb['w3 [m]'], bins=20, alpha=0.7, label='LP-DT34');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 1-axled 04')
print(np.mean(bb['w3 [m]']), np.min(bb['w3 [m]']), np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100), np.min(aa['DT34']/100), np.max(aa['DT34']/100))
plt.show()
```

## 3-AXLED SEMI-TRUCKS + 1-04

```
In [ ]: aa=aa[aa['Axles'] ==4]
bb=bb[bb['Axle 04'] == 1]
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]'])/100; mean2 = np.mean(bb['F2 [kg]'])/100; mean3 = np.me
print(mean1, mean2, mean3, mean4, mean5, mean6, mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print(len(aa), len(bb))
```

## 3-AXLED SEMI-TRUCKS +2-04

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 121) | (df_wim['CONFIG'] == 122) |
                    (df_wim['CONFIG'] == 123) | (df_wim['CONFIG'] == 124) |
                    (df_wim['CONFIG'] == 1211) | (df_wim['CONFIG'] == 1221) |
                    (df_wim['CONFIG'] == 12111) | (df_wim['CONFIG'] == 121111)]
bb = df_C[(df_C['Soort'] == 'opleggertrekker') & (df_C['Axles'] == 3)]
aa=aa[aa['Axles'] ==5]
bb=bb[bb['Axle 04'] == 2]
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]'])/100; mean2 = np.mean(bb['F2 [kg]'])/100; mean3 = np.me
print(mean1, mean2, mean3, mean4, mean5, mean6, mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)

print(len(aa), len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20, alpha=0.7, label='WIM-DT12');
plt.hist(bb['w1 [m]'], bins=20, alpha=0.7, label='LP-DT12');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 2-axled 04')
plt.show()
print(np.mean(bb['w1 [m]']), np.min(bb['w1 [m]']), np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100), np.min(aa['DT12']/100), np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20, alpha=0.7, label='WIM-DT23');
plt.hist(bb['w2 [m]'], bins=20, alpha=0.7, label='LP-DT23');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 2-axled 04')
print(np.mean(bb['w2 [m]']), np.min(bb['w2 [m]']), np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100), np.min(aa['DT23']/100), np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20, alpha=0.7, label='WIM-DT34');
plt.hist(bb['w3 [m]'], bins=20, alpha=0.7, label='LP-DT34');
```

```
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 2-axled O4')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-D45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 2-axled O4')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()
```

## 3-AXLED SEMI-TRUCKS +3-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 121) | (df_wim['CONFIG'] == 122) |
                  (df_wim['CONFIG'] == 123) | (df_wim['CONFIG'] == 124) |
                  (df_wim['CONFIG'] == 1211) | (df_wim['CONFIG'] == 1221) |
                  (df_wim['CONFIG'] == 12111) | (df_wim['CONFIG'] == 121111)]
bb = df_C[(df_C['Soort'] == 'opleggertrekker') & (df_C['Axles'] == 3)]
aa=aa[aa['Axles'] ==6]
bb=bb[bb['Axle O4'] == 3]
```

```
In [ ]: aa
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
#print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
#print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
#print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)
#print((aa['AXW6'] < mean6).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 3-axled O4')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 3-axled O4')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 3-axled O4')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-DT45');
```



```
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 3-axled O4')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

plt.hist(aa['DT56']/100, bins=20,alpha=0.7,label='WIM-DT56');
plt.hist(bb['w5 [m]'],bins=20,alpha=0.7,label='LP-DT56');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 3-axled ST, 3-axled O4')
print(np.mean(bb['w5 [m]']),np.min(bb['w5 [m]']),np.max(bb['w5 [m]']))
print(np.mean(aa['DT56']/100),np.min(aa['DT56']/100),np.max(aa['DT56']/100))
plt.show()
```

## 4-axled tipper trucks

```
In [ ]: aa = df_wim[(df_wim['Axles'] == 4) & (df_wim['DT12'] < 200) & (df_wim['DT23'] < 200)
                & (df_wim['DT34'] < 200) & (df_wim['DT45'] < 200)]
```

```
In [ ]: bb = df_C[(df_C['Soort'] == 'kipper') & (df_C['Axles'] == 4)]
```

```
In [ ]: plt.figure(figsize=(6,6))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax3=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax3=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 4-axled tipper-trucks');plt.xlim(0,160);plt.ylim(0,30)
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 4-axled TT')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 4-axled TT')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()
```

```
plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 4-axled TT')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()
```

## 5-axled tipper trucks

```
In [ ]: aa = df_wim[(df_wim['Axles'] == 5) & (df_wim['DT12'] < 200) & (df_wim['DT23'] < 200)
              & (df_wim['DT34'] < 200) & (df_wim['DT45'] < 200)]
bb = df_C[(df_C['Soort'] == 'kipper') & (df_C['Axles'] == 5)]
```

```
In [ ]: plt.figure(figsize=(6,6))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax3=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax3=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
ax3=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 5-axled tipper-trucks');plt.xlim(0,160);plt.ylim(0,30)
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled TT')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled TT')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled TT')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
```



```
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-D45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled TT')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()
```

## 4-axled mobile cranes & 5-axled mobile cranes

```
In [ ]: aa = df_wim[(df_wim['Axles'] == 4) & (df_wim['DT12'] < 200) & (df_wim['DT23'] < 200)
                & (df_wim['DT34'] < 200) & (df_wim['DT45'] < 200)]
bb = df_C[(df_C['Soort'] == 'mobiele kraan') & (df_C['Axles'] == 4)]
```

```
In [ ]: plt.figure(figsize=(6,6))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
#plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax3=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax3=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
#ax3=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 4-axled mobile cranes');plt.xlim(0,160);plt.ylim(0,30)
plt.show()
```

```
In [ ]: dd = df_C[df_C['Soort'] == 'mobiele kraan']
ee = df_C[(df_C['Soort'] == 'kipper') & (df_C['Axles'] == 4)]
```

```
In [ ]: lengte = bb['Length N3 [m]']
lengte2 = ee['Length N3 [m]']
lengte = pd.to_numeric(lengte)
lengte2 = pd.to_numeric(lengte2)
plt.hist(lengte,bins=25,alpha=0.7,label='LP-Length of mobile cranes');
plt.hist(lengte2,bins=25,alpha=0.7,label='LP-Length of tipper trucks');plt.legend(lo
plt.ylabel('Frequency');plt.title('Lengths for 4-axled vehicles'); plt.xlim(0,17);pl
```

```
In [ ]: lengte = bb['Wheelbase N3 [m]']
lengte2 = ee['Wheelbase N3 [m]']
lengte = pd.to_numeric(lengte)
lengte2 = pd.to_numeric(lengte2)
plt.hist(lengte,bins=25,alpha=0.7,label='LP-Wheelbase of mobile cranes');
plt.hist(lengte2,bins=25,alpha=0.7,label='LP-Wheelbase of tipper trucks');plt.legend
plt.ylabel('Frequency');plt.title('Wheelbase for 4-axled vehicles'); plt.xlim(4,8);p
```

```
In [ ]: aa = df_wim[(df_wim['Axles'] == 4) & (df_wim['DT12'] < 300) & (df_wim['DT23'] < 300)
                & (df_wim['DT34'] < 300) & (df_wim['DT45'] < 300) & (df_wim['LENGTH'] >
bb = df_C[(df_C['Soort'] == 'mobiele kraan') & (df_C['Axles'] == 4)]
```

```
In [ ]: plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
```

```
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
#plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax3=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax3=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
#ax3=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 4-axled mobile cranes');plt.xlim(0,160);#plt.ylim(0,30)
plt.show()
```

```
In [ ]: dd = aa[['McMax [kNm]', 'GVW']]
ee = bb[['McMax [kNm]', 'Max Axle Load [kg]']]
plt.scatter(dd['GVW']*100,dd['McMax [kNm]'],alpha = 1, facecolor='none',color='blue')
plt.scatter(ee['Max Axle Load [kg]',ee['McMax [kNm]'],alpha = 1, facecolor='none',c
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [kg]');plt.ylabel('Bending moment [kNm]');plt.xlim(35e3,65e3);plt.yl
plt.title('Bending moment vs GVW for a bridge of 10 meters')
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.mea
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]',bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 4-axled TT')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]',bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 4-axled TT')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]',bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 4-axled TT')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()
```

## 5-AXLED MOBILE CRANES

```
In [ ]: aa = df_wim[(df_wim['Axles'] == 5) & (df_wim['DT12'] < 300) & (df_wim['DT23'] < 300)
```

```

        & (df_wim['DT34'] < 300) & (df_wim['DT45'] < 300) & (df_wim['LENGTH'] >
bb = df_C[(df_C['Soort'] == 'mobiele kraan') & (df_C['Axles'] == 5)]

```

```

In [ ]: plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax3=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax3=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
ax3=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 5-axled mobile cranes');plt.xlim(0,200);#plt.ylim(0,30)
plt.show()

In [ ]: dd = aa[['McMax [kNm]', 'GVW']]
ee = bb[['McMax [kNm]', 'Max Axle Load [kg]']]
plt.scatter(dd['GVW']*100,dd['McMax [kNm]'],alpha = 1, facecolor='none',color='blue')
plt.scatter(ee['Max Axle Load [kg]'],ee['McMax [kNm]'],alpha = 1, facecolor='none',c
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [kg]');plt.ylabel('Bending moment [kNm]');plt.xlim(40e3,100e3);plt.y
plt.title('Bending moment vs GVW for a bridge of 10 meters')
plt.show()

In [ ]: print(len(aa),len(bb))
mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
#print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
#print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
#print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)

In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

```

```
plt.hist(aa['DT45']/100, bins=20, alpha=0.7, label='WIM-D45');
plt.hist(bb['w4 [m]'], bins=20, alpha=0.7, label='LP-DT45');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w4 [m]']), np.min(bb['w4 [m]']), np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100), np.min(aa['DT45']/100), np.max(aa['DT45']/100))
plt.show()
```

## FULL TRUCKS 2-axled (cat. 11)

```
In [ ]: lijst = ['gesloten opbouw', 'gecond. met temperatuurreg.', 'geconditioneerd voertuig',
                'huifopbouw', 'voor vervoer voertuigen']
```

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 11)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
        (df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
        & ((df_C['Axles'] == 2) & (df_C['Axle 04'] == 0)))]
```

```
In [ ]: plt.hist(aa['AXW1'], bins=bins, alpha=0.7, label=('WIM-AX1'));
plt.hist(aa['AXW2'], bins=bins, alpha=0.7, label=('WIM-AX2'));

ax1 = plt.hist(bb['F1 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX1'))
ax2 = plt.hist(bb['F2 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX2'))

plt.legend(loc='upper right'); plt.xlabel('Axle load [kN]'); plt.ylabel('Frequency')
plt.title('Axle loads for 2-axled full-trucks'); plt.xlim(0, 160); plt.ylim(0, 4000)
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100)
mean2 = np.mean(bb['F2 [kg]']/100)
(aa['AXW1'] < mean1).value_counts()
(aa['AXW2'] < mean2).value_counts()
```

```
In [ ]: print(mean1, mean2)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20, alpha=0.7, label='WIM-DT12');
plt.hist(bb['w1 [m]'], bins=20, alpha=0.7, label='LP-DT12');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 2-axled FT')
plt.show()
print(np.mean(bb['w1 [m]']), np.min(bb['w1 [m]']), np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100), np.min(aa['DT12']/100), np.max(aa['DT12']/100))
```

## FULL TRUCKS 2-axled+1 O4 (cat. 111, 112, 1111, 1112)

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 111)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
        (df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
        & ((df_C['Axles'] == 2) & (df_C['Axle 04'] == 1)))]
```

```
In [ ]: bb
```

```
In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))

plt.legend(loc='upper right');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 2-axled full-trucks, 1-axled O4');plt.xlim(0,250);#plt.yli
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
(aa['AXW1'] < mean1).value_counts();(aa['AXW2'] < mean2).value_counts();(aa['AXW3']
print(mean1,mean2,mean3)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()
```

## FULL TRUCKS 2-axled +2-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 211) |(df_wim['CONFIG'] == 1111)]
bb = df_C[(df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 2) & (df_C['Axle O4']==2))]
```

```
In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 2-axled full-trucks, 2-axled O4');plt.xlim(0,150);#plt.yli
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
(aa['AXW1'] < mean1).value_counts();(aa['AXW2'] < mean2).value_counts();(aa['AXW3']
```



```
print(mean1,mean2,mean3,mean4)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()
```

## FULL TRUCKS 2-axled +3-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 2111)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 2) & (df_C['Axle 04']==3))]
```

```
In [ ]: plt.figure(figsize=(9,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX5'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
ax2=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))

plt.legend(loc='upper right');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 2-axled full-trucks, 3-axled O4');plt.xlim(0,150);#plt.yli
plt.show()
```

```
In [ ]: #(aa['AXW3'] < mean3).value_counts()[0]
mean1 = np.mean(bb['F1 [kg]']/100);
mean2 = np.mean(bb['F2 [kg]']/100);
mean3 = np.mean(bb['F3 [kg]']/100);
mean4 = np.mean(bb['F4 [kg]']/100);
mean5 = np.mean(bb['F5 [kg]']/100);
```

```

print(mean1,mean2,mean3,mean4,mean5)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100) #deze is 100% overLoaded
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean5).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))

```

```

In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-D45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

```

## FULL TRUCKS 3-AXLED +0-04

```

In [ ]: aa = df_wim[(df_wim['CONFIG'] == 12) | (df_wim['CONFIG'] == 21)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 3) & (df_C['Axle 04']==0)))

```

```

In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 3-axled full-trucks');plt.xlim(0,200);plt.ylim(0,400)
plt.show()

```

```

In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.meaa

```



```
print(mean1,mean2,mean3)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()
```

## FULL TRUCKS 3-AXLED + 2-O4 (cat 221, 1121, 2121)

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 221) | (df_wim['CONFIG'] == 1121)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 3) & (df_C['Axle 04']==2))]
```

```
In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX5'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
ax2=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))

plt.legend(loc='upper left');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 3-axled full-trucks, 2-axled O4');plt.xlim(0,200);#plt.yLi
plt.show()
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
```

```
plt.show()

plt.hist(aa['DT34']/100, bins=20, alpha=0.7, label='WIM-DT34');
plt.hist(bb['w3 [m]'], bins=20, alpha=0.7, label='LP-DT34');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']), np.min(bb['w3 [m]']), np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100), np.min(aa['DT34']/100), np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20, alpha=0.7, label='WIM-D45');
plt.hist(bb['w4 [m]'], bins=20, alpha=0.7, label='LP-DT45');
plt.xlabel('Distance [m]'); plt.ylabel('Frequency'); plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w4 [m]']), np.min(bb['w4 [m]']), np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100), np.min(aa['DT45']/100), np.max(aa['DT45']/100))
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100); mean2 = np.mean(bb['F2 [kg]']/100); mean3 = np.me
print(mean1, mean2, mean3, mean4, mean5)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)

print(len(aa), len(bb))
```

```
In [ ]: bb
```

## FULL TRUCKS 3-AXLED +3-04

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 2121)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 3) & (df_C['Axle 04'] == 3))]
```

```
In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'], bins=bins, alpha=0.7, label=('WIM-AX1'));
plt.hist(aa['AXW2'], bins=bins, alpha=0.7, label=('WIM-AX2'));
plt.hist(aa['AXW3'], bins=bins, alpha=0.7, label=('WIM-AX3'));
plt.hist(aa['AXW4'], bins=bins, alpha=0.7, label=('WIM-AX4'));
plt.hist(aa['AXW5'], bins=bins, alpha=0.7, label=('WIM-AX5'));
plt.hist(aa['AXW6'], bins=bins, alpha=0.7, label=('WIM-AX6'));

ax1 = plt.hist(bb['F1 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX4'))
ax2=plt.hist(bb['F5 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX5'))
ax2=plt.hist(bb['F6 [kg]']/100, bins=bins, alpha=0.7, label=('LP-AX6'))

plt.legend(loc='upper right'); plt.xlabel('Axle load [kN]'); plt.ylabel('Frequency')
plt.title('Axle loads for 3-axled full-trucks, 3-axled 04'); plt.xlim(0,200); #plt.yli
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100); mean2 = np.mean(bb['F2 [kg]']/100); mean3 = np.me
print(mean1, mean2, mean3, mean4, mean5, mean6)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
```

```
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)
print((aa['AXW6'] < mean6).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-D45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

plt.hist(aa['DT56']/100, bins=20,alpha=0.7,label='WIM-DT56');
plt.hist(bb['w5 [m]'],bins=20,alpha=0.7,label='LP-DT56');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w5 [m]']),np.min(bb['w5 [m]']),np.max(bb['w5 [m]']))
print(np.mean(aa['DT56']/100),np.min(aa['DT56']/100),np.max(aa['DT56']/100))
plt.show()
```

## FULL TRUCKS 4-AXLED +0-04

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 31) |(df_wim['CONFIG'] == 22) ]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 4) & (df_C['Axle 04']==0))]
```

```
In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
```

```
plt.legend(loc='upper right');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 2-axled full-trucks, 2-axled O4');plt.xlim(0,200);#plt.yli
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.me
print(mean1,mean2,mean3,mean4)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()
```

## FULL TRUCKS 4-AXLED +2-O4 (cat 222,1122,2122)

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 222) |(df_wim['CONFIG'] == 1122) ]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 4) & (df_C['Axle O4']==2))]
```

```
In [ ]: plt.figure(figsize=(7,4))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX5'));
plt.hist(aa['AXW6'],bins=bins,alpha=0.7,label=('WIM-AX6'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
ax2=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))
ax2=plt.hist(bb['F6 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX6'))

plt.legend(loc='upper right');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 4-axled full-trucks, 2-axled O4');plt.xlim(0,250);plt.ylim
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]'])/100;mean2 = np.mean(bb['F2 [kg]'])/100;mean3 = np.me
print(mean1,mean2,mean3,mean4,mean5,mean6)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)
print((aa['AXW6'] < mean6).value_counts()[1]/len(aa)*100)
print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-D45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

plt.hist(aa['DT56']/100, bins=20,alpha=0.7,label='WIM-DT56');
plt.hist(bb['w5 [m]'],bins=20,alpha=0.7,label='LP-DT56');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w5 [m]']),np.min(bb['w5 [m]']),np.max(bb['w5 [m]']))
print(np.mean(aa['DT56']/100),np.min(aa['DT56']/100),np.max(aa['DT56']/100))
plt.show()
```

## FULL TRUCKS 4-AXLED + 3-O4

```
In [ ]: aa = df_wim[(df_wim['CONFIG'] == 2122)]
bb = df_C[((df_C['Soort'] == lijst[0]) | (df_C['Soort'] == lijst[1]) | (df_C['Soort']
(df_C['Soort'] == lijst[3]) | (df_C['Soort'] == lijst[4]) | (df_C['Soort']
& ((df_C['Axles'] == 4) & (df_C['Axle 04']==3))]
```

```
In [ ]: plt.figure(figsize=(7,5))
plt.hist(aa['AXW1'],bins=bins,alpha=0.7,label=('WIM-AX1'));
plt.hist(aa['AXW2'],bins=bins,alpha=0.7,label=('WIM-AX2'));
plt.hist(aa['AXW3'],bins=bins,alpha=0.7,label=('WIM-AX3'));
```



```
plt.hist(aa['AXW4'],bins=bins,alpha=0.7,label=('WIM-AX4'));
plt.hist(aa['AXW5'],bins=bins,alpha=0.7,label=('WIM-AX5'));
plt.hist(aa['AXW6'],bins=bins,alpha=0.7,label=('WIM-AX6'));
plt.hist(aa['AXW7'],bins=bins,alpha=0.7,label=('WIM-AX7'));

ax1 = plt.hist(bb['F1 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX1'))
ax2=plt.hist(bb['F2 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX2'))
ax2=plt.hist(bb['F3 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX3'))
ax2=plt.hist(bb['F4 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX4'))
ax2=plt.hist(bb['F5 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX5'))
ax2=plt.hist(bb['F6 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX6'))
ax2=plt.hist(bb['F7 [kg]']/100,bins=bins,alpha=0.7,label=('LP-AX7'))

plt.legend(loc='upper right');plt.xlabel('Axle load [kN]');plt.ylabel('Frequency')
plt.title('Axle loads for 4-axled full-trucks, 3-axled O4');plt.xlim(0,170);#plt.yli
plt.show()
```

```
In [ ]: mean1 = np.mean(bb['F1 [kg]']/100);mean2 = np.mean(bb['F2 [kg]']/100);mean3 = np.mea
print(mean1,mean2,mean3,mean4,mean5,mean6,mean7)
print((aa['AXW1'] < mean1).value_counts()[1]/len(aa)*100)
print((aa['AXW2'] < mean2).value_counts()[1]/len(aa)*100)
print((aa['AXW3'] < mean3).value_counts()[1]/len(aa)*100)
print((aa['AXW4'] < mean4).value_counts()[1]/len(aa)*100)
print((aa['AXW5'] < mean5).value_counts()[1]/len(aa)*100)
print((aa['AXW6'] < mean6).value_counts()[1]/len(aa)*100)
print((aa['AXW7'] < mean7).value_counts()[1]/len(aa)*100)

print(len(aa),len(bb))
```

```
In [ ]: plt.hist(aa['DT12']/100, bins=20,alpha=0.7,label='WIM-DT12');
plt.hist(bb['w1 [m]'],bins=20,alpha=0.7,label='LP-DT12');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
plt.show()
print(np.mean(bb['w1 [m]']),np.min(bb['w1 [m]']),np.max(bb['w1 [m]']))
print(np.mean(aa['DT12']/100),np.min(aa['DT12']/100),np.max(aa['DT12']/100))

plt.hist(aa['DT23']/100, bins=20,alpha=0.7,label='WIM-DT23');
plt.hist(bb['w2 [m]'],bins=20,alpha=0.7,label='LP-DT23');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w2 [m]']),np.min(bb['w2 [m]']),np.max(bb['w2 [m]']))
print(np.mean(aa['DT23']/100),np.min(aa['DT23']/100),np.max(aa['DT23']/100))
plt.show()

plt.hist(aa['DT34']/100, bins=20,alpha=0.7,label='WIM-DT34');
plt.hist(bb['w3 [m]'],bins=20,alpha=0.7,label='LP-DT34');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w3 [m]']),np.min(bb['w3 [m]']),np.max(bb['w3 [m]']))
print(np.mean(aa['DT34']/100),np.min(aa['DT34']/100),np.max(aa['DT34']/100))
plt.show()

plt.hist(aa['DT45']/100, bins=20,alpha=0.7,label='WIM-D45');
plt.hist(bb['w4 [m]'],bins=20,alpha=0.7,label='LP-DT45');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w4 [m]']),np.min(bb['w4 [m]']),np.max(bb['w4 [m]']))
print(np.mean(aa['DT45']/100),np.min(aa['DT45']/100),np.max(aa['DT45']/100))
plt.show()

plt.hist(aa['DT56']/100, bins=20,alpha=0.7,label='WIM-DT56');
plt.hist(bb['w5 [m]'],bins=20,alpha=0.7,label='LP-DT56');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
```

```
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w5 [m]']),np.min(bb['w5 [m]']),np.max(bb['w5 [m]']))
print(np.mean(aa['DT56']/100),np.min(aa['DT56']/100),np.max(aa['DT56']/100))
plt.show()

plt.hist(aa['DT67']/100, bins=20,alpha=0.7,label='WIM-DT67');
plt.hist(bb['w6 [m]'],bins=20,alpha=0.7,label='LP-DT67');
plt.xlabel('Distance [m]');plt.ylabel('Frequency');plt.legend(loc='upper right');
plt.title('IAD for 5-axled MB')
print(np.mean(bb['w6 [m]']),np.min(bb['w6 [m]']),np.max(bb['w6 [m]']))
print(np.mean(aa['DT67']/100),np.min(aa['DT67']/100),np.max(aa['DT67']/100))
plt.show()
```

## Extremely Heavy Combinations (EHC)

```
In [ ]: aa = df_wim[(df_wim['Axles'] > 7) & (df_wim['GVW'] < 820)]
cc = df_wim[(df_wim['GVW'] > 820) & (df_wim['Axles'] < 7)]
bb = df_C[df_C['Soort'] == 'Extreme']
kk = df_wim[(df_wim['GVW'] > 820) & (df_wim['Axles'] > 7)]
```

```
In [ ]: dd = aa[['McMax [kNm]', 'GVW']]
ee = bb[['McMax [kNm]', 'Max Axle Load [kg]']]
ff = cc[['McMax [kNm]', 'GVW']]
plt.scatter(dd['GVW']*100,dd['McMax [kNm]'],alpha = 1, facecolor='none',color='blue')
plt.scatter(ff['GVW']*100,ff['McMax [kNm]'],alpha = 1, facecolor='none',color='green')
plt.scatter(kk['GVW']*100,kk['McMax [kNm]'],alpha = 1, facecolor='none',color='black')
plt.scatter(ee['Max Axle Load [kg]'],ee['McMax [kNm]'],alpha = 1, facecolor='none',c
leg = plt.legend(loc='upper left')
for lh in leg.legendHandles:
    lh.set_alpha(1)

plt.xlabel('GVW [kg]');plt.ylabel('Bending moment [kNm]');plt.xlim(0,110e3);plt.ylim
plt.title('Bending moment vs GVW - EHC')
plt.show()
```

```
In [ ]: print('Mean of WIM 7+ ax EHC is',np.mean(aa['McMax [kNm]']))
print('Mean of WIM >82t EHC is',np.mean(cc['McMax [kNm]']))
print('Mean of WIM (>82t & >7 AX) EHC is',np.mean(kk['McMax [kNm]']))
print('Mean of LP EHC is',np.mean(bb['McMax [kNm]']))
print('Max of WIM 7+ ax EHC is',np.max(aa['McMax [kNm]']))
print('Max of WIM >82t EHC is',np.max(cc['McMax [kNm]']))
print('Max of WIM (>82t & >7 AX) EHC is',np.max(kk['McMax [kNm]']))
print('Max of LP EHC is',np.max(bb['McMax [kNm]']))
print('Min of WIM 7+ ax EHC is',np.min(aa['McMax [kNm]']))
print('Min of WIM >82t EHC is',np.min(cc['McMax [kNm]']))
print('Min of WIM (>82t & >7 AX) EHC is',np.min(kk['McMax [kNm]']))
print('Min of LP EHC is',np.min(bb['McMax [kNm]']))
```

```
In [ ]: print('Mean of WIM 7+ ax EHC is',np.mean(aa['GVW']))
print('Mean of WIM >82t EHC is',np.mean(cc['GVW']))
print('Mean of WIM (>82t & >7 AX) EHC is',np.mean(kk['GVW']))
print('Mean of LP EHC is',np.mean(bb['Max Axle Load [kg]']/100))
print('Max of WIM 7+ ax EHC is',np.max(aa['GVW']))
print('Max of WIM >82t EHC is',np.max(cc['GVW']))
print('Max of WIM (>82t & >7 AX) EHC is',np.max(kk['GVW']))
print('Max of LP EHC is',np.max(bb['Max Axle Load [kg]']/100))
print('Min of WIM 7+ ax EHC is',np.min(aa['GVW']))
print('Min of WIM >82t EHC is',np.min(cc['GVW']))
print('Min of WIM (>82t & >7 AX) EHC is',np.min(kk['GVW']))
print('Min of LP EHC is',np.min(bb['Max Axle Load [kg]']/100))
```



In [ ]: df\_C

In [ ]:

# Axle contribution LP data

In this notebook for each passing vehicle the share of each individual axle for  $M_{Max}$  will be calculated. In this way, insight is obtained to which (group of) axle(s) have the highest share in  $M_{Max}$ .

```
In [ ]: import sys
import pandas as pd
import numpy as np
import scipy as sc
pd.set_option('display.max_columns', None)
import random
import math as math
import matplotlib.pyplot as plt
from tqdm import tqdm
#np.set_printoptions(threshold=sys.maxsize)
```

```
In [ ]: df = pd.read_csv('df_4400C_load.csv', low_memory=False) ##LET OP. ZONDER ALPHA WAARD
df = df.drop(columns=['McMax [kNm]'])
```

```
In [ ]: def Y(L): #gebruik j om Len(df) aan te duiden
x = np.linspace(0,L+25,(L+25)*10) #lengtes toevoegen om alle aslasten op te vang
y = np.zeros(len(x)) #lege array maken
y1 = x # y1 = x
y2 = y1 - df.iloc[j,18] # y2 = y1 - w1. Wordt gedaan om de onderstaande loop te
y = np.vstack((y1,y2))
for r in range(16): #17 w's vullen. Eindigt op kolom 35
c = y[r+1] - df.iloc[j,r+19]
y = np.vstack((y,c))
y = np.where(y<0,0,y) #negatieve waarden verwijderen
y = np.where(y>L,0,y) #waarden groter dan L verwijderen
return y
```

```
In [ ]: L=10
```

In the cell below, a temporary database is created to determine the influence per axle for  $M_{Max}$ . This is done by determining the location of all axles whenever  $M_{Max}$  is occurring. To achieve this, a new variable called *locid* is introduced which is the location of the bridge where the maximum bending moment is occurring (note. This does not have to be at midspan, due to the influence of other axles). Then, all **relative** locations of the individual axles are determined and their influence on the bending moment is determined. These influences are called  $\eta_{F1}$ . . .  $\eta_{F11}$  and are determined for each axle and for each passing vehicle.

For example, a single axle,  $F_1$  loads a simply supported bridge. Then at midspan, the bending moment is  $\frac{FL}{4}$  and the share of  $F_1$  is 100%. Now, a tandem load of  $F_1 = F_2$  and  $w_1 \ll L$ . The bending moment at midspan becomes  $FL$  and the share of each axle is 50%.

```
In [ ]: collist = ['nF1', 'nF2', 'nF3', 'nF4', 'nF5', 'nF6', 'nF7', 'nF8', 'nF9', 'nF10', 'nF11']
df_y = np.zeros((len(df),len(collist)))

df1y = np.zeros(len(df))
```

```
In [ ]: McMax = np.zeros(len(df))
F_matrix = (df.iloc[:, : 18]*(10/1000)).to_numpy() #selecteren van alle belastingen
for j in range(len(df)): #Len (df)
```

```

y_matrix = Y(L) #y1 = y_matrix[1]

Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

#Av
for r in range(18): #in totaal 18 F's, dus 18 Av.
    Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
#Mc
for r in range(18):
    Mc[r] = np.where(y_matrix[r] < (L/2),
                    (Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
                    Av[r]*(L/2))
    Mc[r] = np.where(Mc[r]<0,0,Mc[r])

Mcmayi = Mc.sum(axis=0) #opsommen
loc_id = (Mcmayi.argmax())

for r in range(10):
    df_y[j,r] = Mc[r,loc_id] / np.amax(Mcmayi)
axno_max_le = np.argmax(df_y[j])

df1y[j] = (loc_id/10) - np.sum(df.iloc[j,18:18+axno_max_le])

Mcmayi = np.amax(Mcmayi)
McMax[j] = Mcmayi

df['McMax [kNm]'] = McMax

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

In [ ]: plt.hist(df1y,bins=100,alpha=0.7,color='blue', label='Yearly data 4400C'); plt.xlabel(
plt.ylabel('Frequency');plt.ylim(0,14500); plt.legend(loc='upper left')
plt.xlim(2.5 , 6.5); plt.title('Distribution location maximum load effect ');

```

```

In [ ]: df_y = pd.DataFrame(data=df_y,columns=collist)
df = pd.concat([df,df_y], axis=1)

```

```

In [ ]: #create df column with total amount of axles per vehicle#
axles = df.iloc[:, :18].to_numpy()
axarr = np.zeros(len(df))
for r in range(len(df)):
    axarr[r] = np.count_nonzero(axles[r])
df['Axles'] = axarr

```

Now the idea is to plot per passing vehicle the total amount of axles and the influence of each axle per passing vehicle.

```

In [ ]: #create df column with total amount of axles that load the vehicle per axle group# T
#Of an n-axled vehicle, all n axles have a 1/n share to M_Max. So: for a 2-axled veh
#contribution to M_max. Otherwise it is considered as SINGLE axled vehicle.
#for 3 axled vehicle, each axle is assumed to contribute 33%. So only the the axles

```

```

loadax = np.zeros(len(df))
nfi = df.iloc[:,48:59].to_numpy()
axarr = axarr
axles = df.iloc[:,40].to_numpy()
print(len(df))
with tqdm(total=len(df), file=sys.stdout) as pbar:
    for j in range(len(df)):
        loadax[j] = len(np.where(nfi[j] > (1/axles[j]))[0])
        pbar.set_description('Calculating nF,i')
        pbar.update(1)
df['AXLOAD'] = loadax

```

In [ ]: df

```

plt.scatter(df['Axles'][:10],df['AXLOAD'][:10],facecolor='none',edgecolor='blue', al
plt.xlim(1,15);plt.ylim(0,11); plt.xlabel('Number of axles'); plt.ylabel('Number of
plt.title('Distribution of axles contributing to maximum load effect')
plt.show()

```

```

In [ ]: from mpl_toolkits.mplot3d import Axes3D
%matplotlib notebook
# Creating figure
fig = plt.figure(figsize = (16,9))
ax = plt.axes(projection = "3d")

# Add x, y gridlines
ax.grid(b = True, color = 'grey',
        linestyle = '-.', linewidth = 0.3,
        alpha = 0.2)

# Creating color map
my_cmap = plt.get_cmap('hsv')
x = df['Axles'].to_numpy();y = df['AXLOAD'].to_numpy();z = df['McMax [kNm]'].to_numpy()
# Creating plot
sctt = ax.scatter3D(x, y, z,
                   alpha = 0.8,
                   c = (x + y + z),
                   cmap = my_cmap,
                   marker = 'o')

plt.title("Contribution of axles of to maximum load effect")
ax.set_xlabel('Number of axles', fontweight = 'bold')
ax.set_ylabel('Contribution to $M_{Max}$', fontweight = 'bold')
ax.set_zlabel('Load effect [kNm]', fontweight = 'bold')
fig.colorbar(sctt, ax = ax, shrink = 0.5, aspect = 5);

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

In [ ]: freq = np.zeros((14,14))
for r in range(1,14):
    for j in range(1,14):
        freqnr = df[(df['Axles'] == r) & (df['AXLOAD'] == j)]
        freq[r,j] = len(freqnr)

```

```
In [ ]: #freq = pd.DataFrame(freq)
        #freq.to_excel('df_4400C_freq_group_ax.xlsx')
```

```
In [ ]:
```

# Axle contribution WIM data

In this notebook for each passing vehicle the share of each individual axle for  $M_{Max}$  will be calculated. In this way, insight is obtained to which (group of) axle(s) have the highest share in  $M_{Max}$ .

```
In [ ]: import sys
import pandas as pd
import numpy as np
import scipy as sc
pd.set_option('display.max_columns', None)
import random
import math as math
import matplotlib.pyplot as plt
from tqdm import tqdm
#np.set_printoptions(threshold=sys.maxsize)
```

```
In [ ]: df_wim = pd.read_csv('WIM - OA_FIL_ROTT_DATA.csv');df_wim = df_wim.fillna(0)
df_wim2 = pd.read_csv('WIM - BK_FIL_ROTT_DATA.csv');df_wim2 = df_wim2.fillna(0)
```

```
In [ ]: def Y_WIM(L): #gebruik j om len(df) aan te duiden
x = np.linspace(0,L+25,(1+(L+25)*10)) #Lengtes toevoegen om alle aslasten op te
y = np.zeros(len(x)) #Lege array maken
y1 = x # y1 = x
y2 = y1 - df_wim.iloc[j,16]/100 # y2 = y1 - w1. Wordt gedaan om de onderstaande
y = np.vstack((y1,y2))
for r in range(9): #10 w's vullen. Eindigt op kolom 26
c = y[r+1] - df_wim.iloc[j,r+17]/100
y = np.vstack((y,c))
y = np.where(y<0,0,y) #negatieve waarden verwijderen
y = np.where(y>L,0,y) #waarden groter dan L verwijderen
return y
```

```
In [ ]: L=10
```

In the cell below, a temporary database is created to determine the influence per axle for  $M_{Max}$ . This is done by determining the location of all axles whenever  $M_{Max}$  is occurring. To achieve this, a new variable called *locid* is introduced which is the location of the bridge where the maximum bending moment is occurring (note. This does not have to be at midspan, due to the influence of other axles). Then, all **relative** locations of the individual axles are determined and their influence on the bending moment is determined. These influences are called  $\eta_{F1}$ . . .  $\eta_{F11}$  and are determined for each axle and for each passing vehicle.

For example, a single axle,  $F_1$  loads a simply supported bridge. Then at midspan, the bending moment is  $\frac{FL}{4}$  and the share of  $F_1$  is 100%. Now, a tandem load of  $F_1 = F_2$  and  $w_1 \ll L$ . The bending moment at midspan becomes  $FL$  and the share of each axle is 50%.

```
In [ ]: collist = ['nF1', 'nF2', 'nF3', 'nF4', 'nF5', 'nF6', 'nF7', 'nF8', 'nF9', 'nF10', 'nF11']
df_y = np.zeros((len(df_wim),len(collist)))
df_y2 = np.zeros((len(df_wim2),len(collist)))

wim1y = np.zeros(len(df_wim))
wim2y = np.zeros(len(df_wim2))
```

```
In [ ]: McMax = np.zeros(len(df_wim))
F_matrix = (df_wim.iloc[:, 3: 14]*1e2*(10/1000)).to_numpy() #selecteren van alle bel
```

```

for j in range(len(df_wim)): #Len (df_wim)

    y_matrix = Y_WIM(L) #y1 = y_matrix[1]

    Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
    Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

    #Av
    for r in range(11): #in totaal 11 F's, dus 11 Av.
        Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
    #Mc
    for r in range(11):
        Mc[r] = np.where(y_matrix[r] < (L/2),
                        (Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
                        Av[r]*(L/2))
        Mc[r] = np.where(Mc[r]<0,0,Mc[r])

    Mcmaxi = Mc.sum(axis=0) #opsommen
    loc_id = (Mcmaxi.argmax())

    for r in range(10):
        df_y[j,r] = Mc[r,loc_id] / np.amax(Mcmaxi)
    axno_max_le = np.argmax(df_y[j])

    wim1y[j] = (loc_id/10) - np.sum(df_wim.iloc[j,16:16+axno_max_le]/100)

    Mcmaxi = np.amax(Mcmaxi)
    McMax[j] = Mcmaxi
df_wim['McMax [kNm]'] = McMax

```

In [ ]:

```

In [ ]: def Y_WIM2(L): #gebruik j om len(df) aan te duiden
x = np.linspace(0,L+25,(1+(L+25)*10)) #Lengtes toevoegen om alle aslasten op te
y = np.zeros(len(x)) #lege array maken
y1 = x # y1 = x
y2 = y1 - df_wim2.iloc[j,16]/100 # y2 = y1 - w1. Wordt gedaan om de onderstaande
y = np.vstack((y1,y2))
for r in range(9): #10 w's vullen. Eindigt op kolom 26
    c = y[r+1] - df_wim2.iloc[j,r+17]/100
    y = np.vstack((y,c))
y = np.where(y<0,0,y) #negatieve waarden verwijderen
y = np.where(y>L,0,y) #waarden groter dan L verwijderen
return y

```

```

In [ ]: McMax = np.zeros(len(df_wim2))
F_matrix = (df_wim2.iloc[:, 3: 14]*1e2*(10/1000)).to_numpy() #selecteren van alle be

for j in range(len(df_wim2)): #Len (df_wim)
    y_matrix = Y_WIM2(L) #y1 = y_matrix[1]

    Av = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))
    Mc = np.zeros((np.shape(y_matrix)[0],np.shape(y_matrix)[1]))

    #Av
    for r in range(11): #in totaal 11 F's, dus 11 Av.
        Av[r] = np.where(y_matrix[r]!=0, F_matrix[j,r]/L*(L-y_matrix[r]),y_matrix[r])
    #Mc
    for r in range(11):
        Mc[r] = np.where(y_matrix[r] < (L/2),
                        (Av[r]*(L/2) - (F_matrix[j,r]*(L/2 - y_matrix[r]))),
                        Av[r]*(L/2))

```



```

                Av[r]*(L/2))
    Mc[r] = np.where(Mc[r]<0,0,Mc[r])

    Mcmaxi = Mc.sum(axis=0) #opsommen
    loc_id = (Mcmaxi.argmax())

    for r in range(10):
        df_y2[j,r] = Mc[r,loc_id] / np.amax(Mcmaxi)
    axno_max_le = np.argmax(df_y2[j])

    wim2y[j] = (loc_id/10) - np.sum(df_wim2.iloc[j,16:16+axno_max_le]/100)
    Mcmaxi = np.amax(Mcmaxi)

    McMax[j] = Mcmaxi
df_wim2['McMax [kNm]'] = McMax

```

```
In [ ]: wim1y
```

```
In [ ]:
```

```
In [ ]: df_y = pd.DataFrame(data=df_y,columns=collist)
df_y2 = pd.DataFrame(data=df_y2,columns=collist)
df_y = pd.concat([df_y,df_y2])
df_C = pd.read_csv('df_4400C_load.csv', sep=',',low_memory = False)
df_wim = pd.concat([df_wim,df_wim2])
df_wim = pd.concat([df_wim,df_y], axis=1)
df_wim = df_wim.dropna()

```

```
In [ ]: #create df column with total amount of axles per vehicle#
axles = df_wim.iloc[:,3:14].to_numpy()
axarr = np.zeros(len(df_wim))
for r in range(len(df_wim)):
    axarr[r] = np.count_nonzero(axles[r])
df_wim['Axles'] = axarr

```

Now the idea is to plot per passing vehicle the total amount of axles and the influence of each axle per passing vehicle.

```
In [ ]: #create df column with total amount of axles that load the vehicle per axle group# T
#Of an n-axled vehicle, all n axles have a 1/n share to M_Max. So: for a 2-axled veh
#contribution to M_max. Otherwise it is considered as SINGLE axled vehicle.
#for 3 axled vehicle, each axle is assumed to contribute 33%. So only the the axles

loadax = np.zeros(len(df_wim))
nfi = df_wim.iloc[:,27:38].to_numpy()
axarr = axarr
axles = df_wim.iloc[:,38].to_numpy()
print(len(df_wim))
with tqdm(total=len(df_wim), file=sys.stdout) as pbar:
    for j in range(len(df_wim)):
        loadax[j] = len(np.where(nfi[j] > (1/axles[j]))[0])
        pbar.set_description('Calculating nF,i')
        pbar.update(1)
df_wim['AXLOAD'] = loadax

```

```
In [ ]: plt.scatter(df_wim['Axles'],df_wim['AXLOAD'],facecolor='none',edgecolor='blue', alph
plt.xlim(1,12);plt.ylim(0,12); plt.xlabel('Number of axles'); plt.ylabel('Number of
plt.show()

```

```
In [ ]: df_wim
```

In [ ]:

In [ ]: `df_wim[5:6][['CONFIG', 'GVW', 'AXW1', 'AXW2', 'AXW3', 'LENGTH', 'DT12', 'DT23', 'McMax [kNm]`In [ ]: `df_wim[141:142][['CONFIG', 'GVW', 'AXW1', 'AXW2', 'AXW3', 'AXW4', 'AXW5', 'AXW6', 'LENGTH', 'DT34', 'DT45', 'DT56', 'McMax [kNm]', 'nF1', 'nF2', 'nF3', 'AXLOAD']]`In [ ]: `df_wim[8217:8218][['CONFIG', 'GVW', 'AXW1', 'AXW2', 'AXW3', 'AXW4', 'AXW5', 'AXW6', 'AXW7', 'DT34', 'DT45', 'DT56', 'DT67', 'DT78', 'McMax [kNm]', 'nF4', 'nF5', 'nF6'`In [ ]: `plt.scatter(df_wim['Axles'], df_wim['AXLOAD'], facecolor='none', edgecolor='blue', alpha=0.8, plt.xlim(1,12); plt.ylim(0,12); plt.xlabel('Number of axles'); plt.ylabel('Number of axles'); plt.title('Contribution of axles to maximum load effect') plt.show()`In [ ]: 

```
freq = np.zeros((13,13))
for r in range(1,13):
    for j in range(1,13):
        freqnr = df_wim[(df_wim['Axles'] == r) & (df_wim['AXLOAD'] == j)]
        freq[r,j] = len(freqnr)
```

In [ ]: 

```
from mpl_toolkits.mplot3d import Axes3D

# Creating figure
fig = plt.figure(figsize = (16,9))
ax = plt.axes(projection = "3d")

# Add x, y gridlines
ax.grid(b = True, color = 'grey',
        linestyle = '-.-', linewidth = 0.3,
        alpha = 0.2)

# Creating color map
my_cmap = plt.get_cmap('hsv')
x = df_wim['Axles'].to_numpy(); y = df_wim['AXLOAD'].to_numpy(); z = df_wim['McMax [kNm]']

# Creating plot
sctt = ax.scatter3D(x, y, z,
                   alpha = 0.8,
                   c = (x + y + z),
                   cmap = my_cmap,
                   marker = 'o')

plt.title("Contribution of axles of to maximum load effect")
ax.set_xlabel('Number of axles', fontweight = 'bold')
ax.set_ylabel('Contribution to $M_{Max}$', fontweight = 'bold')
ax.set_zlabel('Load effect [kNm]', fontweight = 'bold')
fig.colorbar(sctt, ax = ax, shrink = 0.5, aspect = 5);
```

In [ ]: 

```
#for ii in range(0,360,1):
#    ax.view_init(elev=25., azimuth=ii)
#    plt.savefig("movie%d.png" % ii)
```

In [ ]: `plt.hist(np.append(wim1y,wim2y),bins=100,alpha=0.7,color='green', label='WIM data'); plt.xlabel('$x_{max,LE}$ [m]', fontsize=12); plt.ylabel('Frequency'); plt.ylim(0,20000); plt.legend(loc='upper left'); plt.xlim(2.5 , 8.5); plt.title('Distribution location maximum load effect ');`In [ ]: `yy = np.append(wim1y,wim2y)`

```
In [ ]: np.max(yy)
```

```
In [ ]: np.sort(yy)
```

```
In [ ]:
```