Dortmund University

Faculty of Statistics

Summer term 2023

Master's Thesis

An analysis of municipal vehicle data using classification methods and kernel density estimation

Author:

Daniel Schürmann

April 30, 2023

Contents

1 Introduction

This paper describes the complete process of analysing vehicle data from the City of Dortmund. It is meant as a showcase for advanced / state-of-the-art techniques that could be used by authorities dealing with statistical data. Some ideas shown can be used to automate cumbersome procedures. Dealing with official statistics has some unique demands, but the text is written to ensure easy transfer to other areas where possible.

The paper is organized as follows. Chapter 2 shows the needed data cleaning techniques used to do address matching and missing data imputation. Chapter 3 continues with the task of classifying cars into segments. Visualization techniques based on kernel density estimation are described in chapter 4. Chapter 5 provides a proposal how to publish high resolution spatial data in a privacy preserving manner. Spatial analysis by the means of the Kantorovich distance can be found in chapter 6. Chapter 7 concludes the analysis.

Special thanks should be given to the City of Dortmund's employees for providing the required data access. All data processing has been done with the statistical software `R` (R Core Team, 2023) unless stated otherwise.

2 Treating data

This chapter provides some background information to understand the available data as well as needed data cleaning steps. Vehicle data is available from the City of Dortmund's database systems. Dortmund is a metropolitan city in the western part of Germany that has currently (2023) a population of about 600,000 people and 300,000 cars. The city's extend is about 20km in north-south as well as east-west direction. Vehicle administration in Germany is primarily done by each of the about 11,000 municipalities separately. In addition, municipal data is shared with the federal vehicle office (KBA). In Germany car insurance is mandatory. Insurance companies report their policies to the federal office, which in return forwards the information back to the municipalities.

This decentralized structure implies that every issue raised in this paper is only valid for the City of Dortmund, though other municipalities may share similar problems. The first problem is undocumented legacy software systems, which somehow work, but it is unknown how. This requires "reverse engineering" the data generating process to understand the database tables. The records are transaction based. Every transaction (e.g. change of insurance, ownership, registration plate, theft, etc.) yields a new record. Every record has a validity date attached (e.g. valid till "01.01.2099" means currently valid record). This implies that old records are retroactively changed. A typical record has columns about the owner's name, address, date of birth, owner type (private, or non-private entity), the car's manufacturer ID, model ID, vehicle identification number (VIN), mass, date of first registration, top speed, vehicle type (i.e. car, motorbike, lorry, trailer), date of record creation as well as validity date. The process of identifying vehicles registered at a cut-off date is constructed as finding all records for a specific vehicle (identified by VIN) and then looking if a record exists where the date of record creation is before the cut-off date and the validity date is after the cut-off date. A VIN is a in ISO 3779 standardized 17-digit alphanumeric code uniquely identifying every vehicle since the 1980s. The VIN is engraved into a vehicle's chassis and can not be changed. Strangely, the dataset contains records with invalid VINs (e.g. "0001"). Those records (about 2,000 to 3,000 at every single point in time) have been discarded from the analysis as there is no way to reliably tell those vehicles apart. Querying the database for past data returned plausible results back to the year 2014. Counts prior to 2014 are incorrect. Obviously old records are purged form the database, but there is no documentation available on when records are deleted.

In addition, there is a second database table with technical information (i.e. length, width, and height of vehicle), which is merged into the dataset. The technical database sometimes includes multiple records per vehicle. In those cases the record having the least amount of missing values is selected. The federal vehicle office provides supplementary data (KBA, 2023b) on vehicle mass, number of seats and axles, engine output, energy source and model name as text. The KBA notoriously changes their website links without redirections. To prevent link rot, all references have been changed to the Internet Archive's cached versions. Unfortunately the KBA's data is only available as a huge 870 page long pdf-file. In 2012 legal procedures based on the freedom of information act were initialized to get access to a machine-readable version of the pdf (Frag den Staat, 2012). The KBA's on first sight absurd rebuttal claims that the raw data is not available, but they are willing to sell them for a price between €300-€400 anyway. This makes sense when considering that the KBA does not actually have the data (confirmed via phone call). All data processing steps have been outsourced to an external service provider. The pdf is processed by text extraction tools and the information is merged into the dataset on manufacturer and model ID. In case data differs, the KBA data is preferred. The municipalities' vehicle database is maintained manually (e.g. there is no auto-complete on technical data), which makes mistakes more likely.

2.1 Address matching

Having covered the temporal and technical aspects, the next problem is the spatial location of vehicles defined by the owner's address. There is a general caveat with non-privately registered vehicles. For example, there are three (IT-) consultancy contractors head-quartered in Dortmund. Those companies have subsidiaries all over Germany and many of their employees have company cars, which are all registered at a single address in Dortmund. Therefore, when analysing spatial distributions, all non-private vehicles are omitted, because those vehicles most likely never have been to Dortmund and the number of vehicles is mainly a count of the companies' country-wide employees.

Spatial information is provided by text strings for city, street and house number. The dataset has a column called street number, which could provide the street's ID (each street has a unique ID) and making the following steps unnecessary. Unfortunately the column is empty and the feature was never implemented. As already mentioned, there is no autofill feature ensuring correct addresses. Instead, the addresses have to be matched against a database of all addresses with their corresponding coordinates in ETRS89 / UTM zone 32N coordinate system, which is a lambert azimuthal equal-area projection (angles are distorted but areas are correct due to the Earth's sphere projected onto a plane). Values are measured in meters making the coordinates easy interpretable. An addresses' coordinate is actually the building's entrance, not its centroid. Plotting maps is done with the R-package `sf` (see Pebesma, 2018 and Pebesma and Bivand, 2023).

Matching addresses is a cumbersome but needed procedure, because failed matches do not appear completely-at-random or at-random. Directly matching fails at difficult to write street names (missing-not-at-random). For example, the correct street name "JOSEPH-VON-FRAUNHOFER-STRAßE" is written as "JOS-V.-FRAUNHOFER-S", "J.-V.-FRAUENHOFER-STR", "JOSEF-V.-FRAUNH.-STR.", "JOSEPH-V-FRAUNHOFER-S.", "JOSEPH-V-FRAUENH.-STR.", "JOS-V-FRAUENHOFER-STRAßE", "JOS.-V.-FRAUNHOF.STR.", "J.-V.-FRAUNHOFER-STR.". A multi-stage matching procedure based on string distances is used. The first step is matching the city, the second is matching the street and the last one is matching the house number. Matching the city is necessary, because the dataset includes addresses in other cities and even in Poland, Spain and The Netherlands. It is unclear why this is the case. Some examples of misspelling DORTMUND are "DORMTUND, DORMUND, DORMTMUND, DO, DORTNMUND". The special case "DO" as abbreviation for DORTMUND is handled by a special rule.

Any distance function d measuring the similarity between two strings s, t and u should satisfy the following axioms (van der Loo, 2014):

- non negativity $d(s,t) \geq 0$
- identity $d(s,t) = 0$ iff. $s = t$
- symmetry* $d(s,t) = d(t,s)$
- triangle inequality* $d(s,u) \leq d(s,t) + d(t,u)$

In this case, the "optimal string alignment" method proved to be the most useful for addresses. It is based on the edit distance, which counts the number of needed operations to turn one string into another, i.e. insertion, deletion, substitution and transpositions (e.g. transpose "M" and "T" in "DORMTUND"). Each operation gets a weight $\omega_i, \omega_d, \omega_s$ and ω_t attached to it. Computing the optimal string distance is done recursively (van der Loo, 2014):

$$d(s,t) = \min \begin{cases} 0 & \text{if } |t|, |s| = 0, \\ d(s, t_{1:(|t|-1)}) + \omega_d & \text{if } |t| \geq 1, \\ d(s_{1:(|s|-1)}, t) + \omega_i & \text{if } |s| \geq 1, \\ d(s_{1:(|s|-1)}, t_{1:(|t|-1)}) + \omega_s I(s_{|s|} \neq t_{|t|}) & \text{if } |s|, |t| \geq 1, \\ d(s_{1:(|s|-2)}, t_{1:(|t|-2)}) + \omega_t I(s_{|s|} \neq t_{|t|}) & \text{if } s_{|s|} = t_{|t|-1} \wedge s_{|s|-1} = t_{|t|} \wedge |s|, |t| \geq 2, \end{cases}$$

where for a string $s =$ "DORTMUND" $|s| = 8$ is the number of characters, s_i the i-th character in s (e.g. $s_2 =$ "O"), $s_{1:i}$ is the sub string of the first i characters (e.g. $s_{1:4} =$ "DORT") and I is the indicator function. Depending on the weights ω chosen, d might not meet the symmetry and triangle inequality demands (e.g. $\omega_i \neq \omega_d$ violates symmetry). Before the actual string matching is done, all non-characters are removed from the strings, special characters are replaced (i.e. ä = ae, ü = ue, ö = oe, ß = ss), the abbreviation "str" is expanded to "strasse" and all strings are converted to lower case. Matching the city is done with weights $\omega_d = 1, \omega_i = 1, \omega_s = 0$ and $\omega_t = 0$. For street name matching $\omega_d = 1, \omega_i = 0.5, \omega_s = 1$ and $\omega_t = 1$ is used because street names are usually abbreviated, which implies weighing insertions with a lower penalty. String matching is implemented in (van der Loo, 2014)'s `stringdist` R-package. After all non-Dortmund addresses have been removed and the streets are matched on a threshold of the normalised distance $\frac{d}{|s|} > 0.9$, all valid house numbers for the matched street are selected. If no direct match on the numbers can be found, but one within an absolute distance of four, the best matching number measured by absolute distance is selected. If no such number can be found, a house number is selected randomly from all valid house numbers for that street. This is violating procedures used e.g. by the German federal statistics office, which uses the median house number, but has the advantage of being unbiased. Using the median house number might cause density spikes at the middle of long streets, which do not actually exist. For low resolution statistics provided by the federal office, this is not an issue, but for high resolution methods used in this paper, the issue is real. Table 2.1 shows the matching results. Direct match refers to matching addresses directly without any data processing steps. Sanitized matches are done on the sanitized strings. Those types of matches can be considered to be correct in all cases. The category fuzzy match uses the string distance based matching procedure described above and are highly likely to be correct, but some addresses are certainly wrongly matched. The last column "no match" refers to addresses, where the City successfully matched as Dortmund, but the street fuzzy matching score is below the threshold of 0.9 and thus no coordinates can be allocated. They are omitted from all spatial analysis in this paper. This should not be an issue as they can be regarded as missing-completely-at-random and only make up less than 0.5% of all vehicles. Table 2.1 implies a better data quality for the years 2014, 2015 and 2016. This is not the case, as additional manual address correction was used for those years. For data after 2017, this has been omitted because it was not feasible to do so.

Table 2.1: Address qualities in %

year	direct match	sanitized match	fuzzy match	no match
2014	99.00	0.10	0.80	0.10
2015	98.90	0.10	0.90	0.10
2016	98.90	0.10	0.90	0.10
2017	94.30	3.80	1.50	0.40
2018	94.30	3.90	1.50	0.40
2019	94.30	4.00	1.40	0.30
2020	94.10	4.30	1.30	0.30
2021	94.20	4.20	1.20	0.30
2022	94.20	4.30	1.20	0.30
2023	94.20	4.40	1.20	0.30

2.2 Imputation

This section deals with imputing the remaining missing values. Chapter 3 tries to classify cars into segments by analysing their technical data which requires a complete dataset. To get some idea of the problem's scope, tables 2.2 and 2.3 show the proportions of missing data. Most problematic is the share of about 15% to 30% missing data in vehicle length, width and height. This information is crucial in classifying cars. All other features are included in the imputation process as well, though imputated data is not used in any further analysis in this paper except indirectly due to the inferred vehicle segment. Imputation models are estimated for each year separately, because vehicle's technical data obviously change over time (e.g. bigger, heavier, etc.).

Table 2.2: Missing values in %

year	pollution class	top speed	co_2 emissions	length	width	height	axle 1 load
2014	0.10	1.50	34.50	15.90	28.30	16.80	1.60
2015	0.00	1.60	29.70	17.10	29.50	18.10	1.70
2016	0.00	1.60	25.50	18.00	30.30	19.00	1.60
2017	0.00	0.80	21.00	18.10	30.50	19.10	0.80
2018	0.00	0.80	18.20	18.80	31.00	19.80	0.80
2019	0.00	0.70	15.70	19.20	31.20	20.40	0.80
2020	0.00	0.70	13.50	19.50	31.20	20.90	0.70
2021	0.00	0.60	11.70	18.80	30.00	20.10	0.60
2022	0.00	0.40	10.50	18.10	28.80	19.40	0.40
2023	0.00	0.30	10.70	17.40	27.70	18.60	0.30

Table 2.3: Missing values in % (cont.)

year	owner's sex	owner's age	engine output	engine size	no. of axles	no. of drive axles	no of seats
2014	7.40	7.40	0.10	3.10	3.10	3.10	0.10
2015	7.40	7.40	0.10	2.90	2.90	2.90	0.10
2016	7.60	7.60	0.10	2.80	2.80	2.80	0.10
2017	8.00	8.00	0.00	2.70	2.70	2.70	0.00
2018	8.00	8.00	0.00	2.60	2.60	2.60	0.00
2019	8.20	8.20	0.00	2.40	2.40	2.40	0.00
2020	8.80	8.80	0.00	2.40	2.40	2.40	0.00
2021	8.70	8.70	0.00	2.30	2.30	2.30	0.00
2022	8.90	8.90	0.00	2.30	2.30	2.30	0.00
2023	9.20	9.20	0.00	2.30	2.30	2.30	0.00

Many different methods exist to impute missing values. Currently, the most powerful ones seem to be "MICE" and "MIDAS" with "MIDAS" having better performance (Gondara and Wang, 2018). "MIDAS" is short for multiple imputation using de-noising auto-encoders. An auto-encoder belongs to the class of neural networks. Before going into MIDAS' details, a basic understanding of neural networks is required. A simple feed forward neural network is a supervised learning method matching input data x to the desired output data y. It consists of an input layer, an output layer and multiple hidden layers between them. In mathematical terms (Goodfellow et al., 2016, pp. 168)

$$y = \phi_\dagger(W_\dagger\phi_A(W_A \dots \phi_1(W_1\phi_*(W_*x + b_*) + b_1) \dots + b_A) + b_\dagger),$$

the network is a composition of the input layer function ϕ_*, a total of 1 to A hidden layer functions $\phi_1, \dots, \phi_A$ and a final output layer function $\phi_\dagger$. Each layer has multiple nodes which take in multiple inputs from previous layers. In a fully connected network, each node has all previous outputs as inputs available. The number of nodes within a layer is referred to as the layer's size. Each node then applies the layer function ϕ to its scaled (by a matrix W) and shifted (by a vector b) input values. Choosing the ϕ, the number of layers and the layer's size has to be done beforehand. The benefits of neural networks can be exploited by choosing the ϕ as non-linear functions (otherwise it is just a simple linear model).

The difficult task is optimizing the W and the b. Stochastic gradient descent is the algorithm used for this step. As the name already implies, the gradient of a loss function is needed. The gradient is computed by the efficient back-propagation algorithm. But first, the loss function needs to be considered. Let y be the desired output and $\hat{y}$ the network's actual output. If both are continuous, the loss is calculated as $l(\hat{y}, y) = \sqrt{\frac{1}{n}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$, the root mean squared error (RSME). If the variables are binary or categorical, the loss function becomes $l(\hat{y}, y) = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\ln(\hat{y}_{ij})$, where $y_i = (0, \dots, 0, 1, 0, \dots 0)$ indicates which of K class y_i belongs to and $\hat{y}_{ij}$ is the models estimated probability of y_i belonging to class $1 \leq j \leq K$(Lall and Robinson, 2023). This is similar to the concepts in logistic regression and is referred to by cross-entropy.

After the models first run (called epoch) with random weights, the resulting loss is calculated as $l_\theta(\hat{y}, y)$, where θ is the set of all parameters (the W and b) used to calculate $\hat{y}$. Recall that $\hat{y}$ is the result of composing the ϕ. Optimizing the weights by minimizing the loss requires calculating the loss function's gradient. The obvious solution is to iteratively apply the chain rule of calculus. This is the basic idea behind the back-propagation algorithm. Consider the function composition $f(g(x))$ with notation $g(x) = y$ and $f(y) = z$, the gradient becomes (Goodfellow et al., 2016, p. 207)

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^T \nabla_y z,$$

with $\frac{\partial y}{\partial x}$ being the hessian matrix. The back-propagation algorithm calculates the gradients by storing and reusing intermediate results. This speeds up the computation by avoiding duplicate calculations (Goodfellow et al., 2016, p. 210).

The learning (i.e. modifying θ) stochastic gradient descent does is simply setting

$$\theta_{\text{new}} = \theta_{\text{old}} - \varepsilon \nabla_\theta l,$$

where ε is called the learning rate and $\nabla_\theta l$ is the loss function's gradient with regard to θ (Goodfellow et al., 2016, p. 152). As only the first derivative is used, it is a simpler method than e.g. Newton–Raphson. The complexity is reduced even further by not evaluating $\nabla_\theta l$ for all observations. Instead, only a random sample called mini-batch is used (hence the name stochastic gradient descent). This does not lead to optimal results, but often to good enough results. Those are the basic building blocks for a feed forward neural network.

"MIDAS" uses auto-encoders, which can best be understood as being two feed forward neural networks chained together. The first one, called "encoder" takes the input values and is forced to output a representation in a lower dimensional space. The second network, the "decoder" takes this dimensionality reduced input and tries to recreate the original output. This is an unsupervised learning method, as the target output is the same as the input (Goodfellow et al., 2016, p. 396). In this context, missing data can be thought of data being reduced in dimensionality. The obvious solution for the network is to learn the identity function. To avoid this outcome, "MIDAS" needs to deploy some tweaks to prevent over fitting. The first measure is to corrupt the input by introducing an additional 20% of missing values (encoded as zeros) at each training epoch (not just once at the beginning of the training). The second measure is to randomly drop some nodes in the hidden layer of the encoder (default probability for a drop-out is 50%) (Lall and Robinson, 2022). For the encoder part of the auto-encoder, the l-th hidden layer function becomes $W_l v \phi_l(x^{(l-1)} + b_l)$, where $v \sim \text{Bern}(p = 0.5)$.

The non-linear function ϕ MIDAS uses is called an exponential linear unit (ELU) given by (Lall and Robinson, 2023)

$$\phi(x) = \begin{cases} \alpha(\exp(x) - 1) & x \leq 0 \\ x & x > 0 \end{cases},$$

where $\alpha > 0$ is added to the parameters to be optimized (θ). The output layer function $\phi_\dagger$ depends on the type of variable (Lall and Robinson, 2023)

$$\phi_\dagger(x) = \begin{cases} x & \text{if } x \text{ continuous} \\ \frac{1}{1+\exp(-x)} & \text{if } x \text{ binary} \\ \left(\frac{\exp(x_1)}{\sum_{i=1}^{K} \exp(x_i)}, \dots, \frac{\exp(x_K)}{\sum_{i=1}^{K} \exp(x_i)} \right)^T & \text{if } x \text{ categorical} \end{cases}$$

The output function for categorical data is the softmax function, which scales an n-dimensional input vector to a vector that sums up to 1 and can be interpreted as the probabilities of a multinomial distribution. The loss is calculated between the original input (not the corrupted one that gets fed into the network) and only for observations that were originally not corrupted. The loss function is furthermore modified to include an extra loss for having big weighs in θ by adding $\lambda ||\theta||^2$ to l. Similar to the idea employed in stochastic gradient descent, only a small mini-batch sample is used to train the model. MIDAS is implemented in the `rMIDAS` package (Robinson et al., 2022), which uses a python interface to the Alphabet / Google Inc. sponsored TensorFlow library.

For the vehicle dataset a network structure of two hidden layers, each one with 50 nodes proved to be satisfactory. The number of epochs (learning iterations) is determined by splitting the complete data into a training and evaluation set and comparing imputation losses. Figure 2.1 shows the error scores at different training epochs. Each reporting interval is equal to three training epochs. The red crosses indicate the minimum loss for each variable type. A good choice seems to reached at reporting interval 6 (i.e. 18 training epochs).

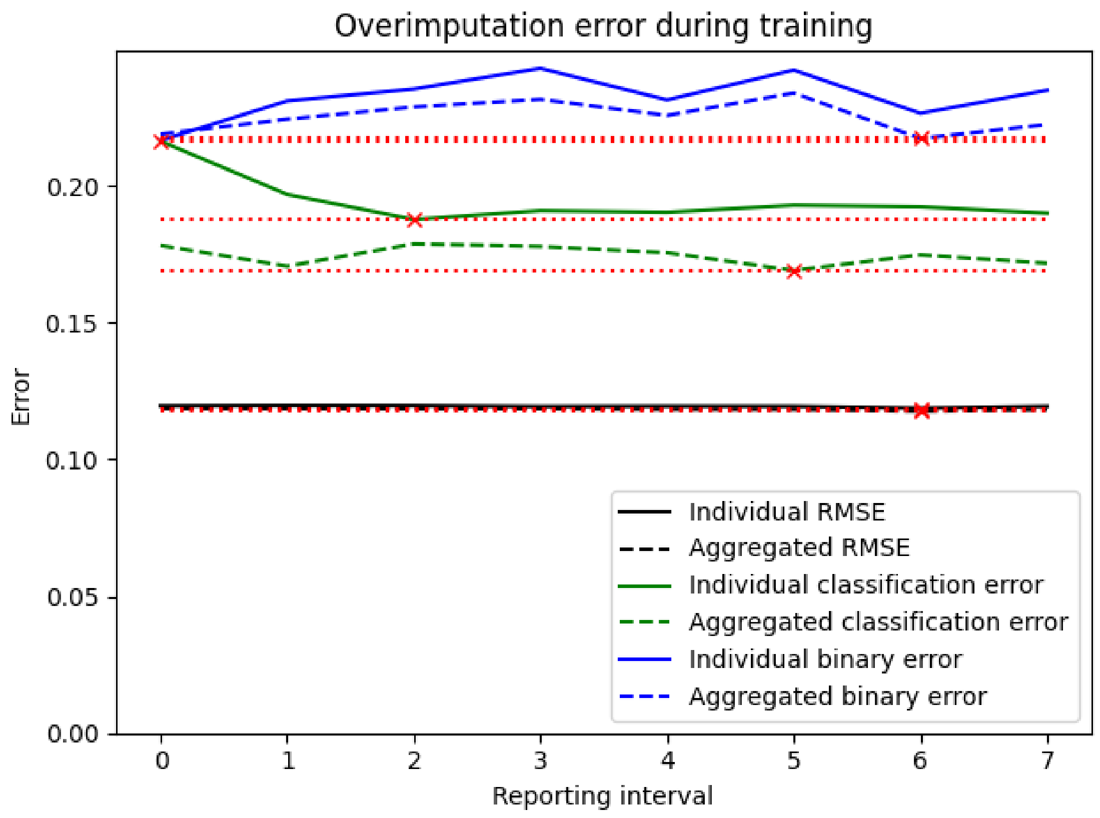


Figure 2.1: Imputation validation

3 Classifying data

Obviously, not all cars are equal. The federal vehicle office publishes every year statistics on car segments (KBA, 2023a). Closer examination of the federal statistics reveal the most popular car models in each category, though this list is non-exhaustive. There obviously exists an exhaustive list of which car model belongs to which segment, but it is not available. A phone call confirmed that segment statistics are done by an external service provider. The model names for each segment from table (KBA, 2023a) are extracted, and regular expressions are used to match the names to the corresponding manufacturer and model ID in the (KBA, 2023b) table already discussed in chapter 2. The manufacturer and model ID are then matched against the municipal's vehicle database. Most of those segments are recognized by the European Commission, but there are some subtle differences as outlined in table 3.1. The following analysis deals only with cars (i.e. maximum weight of 3,500kg).

Table 3.1: Car segments

German classification	European classification	EU Segment code
Mini	Mini	A
Small	Small	B
Medium	Medium	C
Large	Large	D
Executive	Executive	E
Luxury	Luxury	F
Sport utility vehicle All-terrain vehicle	Sport utility	J
Sports	Sports	S
Mini-van Big-van Utilities	Multi-purpose	M
Motorhome	n/a	n/a

Special considerations must therefore be taken when talking about the utilities and motorhome segments as these usually include vehicles regarded as lorries. Table 3.1 differentiates between sport utility vehicles (SUV) and all-terrain vehicles (ATV). The main difference is that ATVs can be used off-road (e.g. have four-wheel drive) whereas SUVs just look as if they could.

Strangely, there is no publicly available technical definition of vehicle segments (e.g. cars over 300hp belonging to segment S). Instead, car manufacturers do the classification by themselves based on some non-published rules. The following section tries to find a rule for classifying cars into segments based on the car's technical data. The following variables are used:

- top speed
- length in mm
- height in mm
- width in mm
- first axle load in kg
- engine output in kW
- kerb weight in kg

- number of axles (2 or 3)
- number of drive axles (1 or 2)
- number of seats.

The current consensus on which classification method seems to recommend neural network based approaches for unstructured data (e.g. text, audio, pictures, videos) and gradient boosted trees (especially xgboost) for structured data (Chollet et al., 2022). Furthermore, xgboost has a speed advantage compared to neural networks. The basic idea behind xgboost is outlined below.

To get some intuition into classification trees consider the following simple example: If a car has more than 300hp, classify as sports car. If not and if number of seats is greater than eight, classify as big-van, else continue down the remaining tree. In formal terms, a classification tree splits the feature space into T disjoint multidimensional boxes $\mathcal{R}_i$. Let m be the number of features (i.e. length, height, width, mass, etc.), s the number of labels (i.e. 13) and $x_i = (x_{i,1}, \ldots, x_{i,m})^T$ be the features of car i. A tree q is then defined as (Hastie et al., 2009, p. 305):

$$q(x_i) = \sum_{i=1}^{T} w_i I\{x_i \in \mathcal{R}_i\} \text{ with } \dot{\bigcup}_{i=1}^{T} \mathcal{R}_i = \mathbb{R}^m, \text{ and } w_i \in \mathbb{R}$$

w_i is a weight for the leave / basket x_i has been classified into. To construct a tree, three decisions need to be taken: The first one is on which feature (e.g. number of seats) to split the tree. The second one is on which value the split should be done (e.g. > 8 or ≤ 8). And the last one is when to stop splitting the tree. In order to find an optimal tree, the set of cars ending up in one basket should ideally all have the same label y_i (i.e. segment). After all training data have been distributed into the baskets, the empirical distribution function of the labels is calculated for each basket, i.e. counting the cars corresponding to each label divided by all cars in basket w. The resulting probability mass function is called q_w and $q_{w,i}$ refers to the probability of label i within basket w. This is used to construct a loss function between the true label y_i and the estimated label $\hat{y}_i$:

$$l(\hat{y}_i, y_i) = -\sum_{i=1}^{s} \ln(q_{w,i}) I\{y_i = i\} \text{ with } w = \hat{y}_i$$

This loss function is also referred to as cross-entropy (Martinez and Stiefelhagen, 2019). The further $q_{w,i}$ (the probability of guessing the correct label) is away from 1, the exponentially higher the cost. Minimizing the loss function allows for finding the locally best split points by finding the best split points for all features and then choosing the feature and split point with minimal loss. Without regularization these results in constructing a tree having zero loss by assigning each observation its own exit node. This is clearly undesirable. Therefore, a punishing factor γ for having a tree with many exit nodes T is added to the loss function $\mathcal{L}$:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_i, y_i) + \gamma T, \ \gamma > 0.$$

For simple classification problems, this approach might already be sufficient. More complex problems require the combination of multiple trees (an ensemble). Unfortunately this also means sacrificing the interpretability of the model. The method of combination "xgboost" uses is boosting. The following part describes how "xgboost" works and is based on the works of (Chen and Guestrin, 2016).

"xgboost" works with K classification trees. The prediction of a car's segment $\hat{y}_i$ now becomes

$$\hat{y}_i = \sum_{i=1}^{K} q_k(x_i)$$

the sum over K trees and the new loss function extends to

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_i, y_i) + \sum_{j=1}^{K} \left(\gamma T_k + \frac{1}{2}\lambda ||w_k||^2 \right),$$

where w_k is the vector of all weights of tree k. The remaining questions are how to add a tree and how to optimize $\mathcal{L}$. Boosting is a method to sequentially add trees as needed. The loss function with t trees $\mathcal{L}^{(t)}$ and corresponding prediction $\hat{y}_i^{(t)}$ is

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} l[y_i, \hat{y}_i^{(t-1)} + q_t(x_i)] + \gamma T_t + \frac{1}{2}\lambda ||w_t||^2.$$

The added tree t should be the one, which decreases above's loss function the most (if at all). The idea is to differentiate $\mathcal{L}^{(t)}$ and search for the optimum. This can not be done directly. Instead, a second order taylor approximation is used:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{N} [l(y_i, \hat{y}_i^{(t-1)}) + g_i q_t(x_i) + \frac{1}{2} h_i q_t^2(x_i)] + \gamma T_t + \frac{1}{2}\lambda ||w_t||^2,$$

where

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \text{ and } \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial^2 \hat{y}_i^{(t-1)}}.$$

This why the method is called gradient boosting. When optimizing for q_t the first term $l(y_i, \hat{y}_i^{(t-1)})$ can be ignored. Let $I_j = \{1 \leq i \leq n | q_t(x_i) = j\}$ be the indices of observations q_t puts into basket j. The optimal weights w_t for a given tree q_t are

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}.$$

The only remaining question is how to split the tree. For a proposed split of the observations in I into the two baskets I_R and I_R, the loss reduction is

$$\mathcal{L}_{\text{Reduction}} = \frac{1}{2}\left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda}\right] - \gamma.$$

The algorithm for finding the best split according to above's formula given the observations in I is

```
1. Set $G = \sum_{i \in I} g_i$, $H = \sum_{i \in I} h_i$ and score = 0
2. repeat steps 3 to 7 for k=1 to m
    3. $G_L = 0$ and $H_L = 0$
    4. repeat steps 5 to 7 for j in {sort I by $x_{jk}$}
        5. $G_L = G_L + g_j$ and $H_L = H_L + h_j$
        6. $G_R = G - G_L$ and $H_R = H - H_L$
        7. score = max $\left(\text{score}, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
8. split with maximum score
```

The two hyperparameters λ and γ need to be set manually in a trial-and-error approach by reserving 20% of the training set as evaluation set and testing the model on the evaluation set. Xgboost is implemented in the R-package `xgboost` (Chen, He, et al., 2023). In order to ensure consistency one global model is estimated on the imputed datasets and subsequent applied to the data for each year.

Figure 3.1 shows the classification results. Comparing the results to the federal statistics shows a higher proportion of minis and small cars. This is to be expected as Dortmund is a metropolitan area with usually smaller cars than in rural areas. Overall the results are within the same range, which makes the classification plausible. About one quarter of the cars is classified by xgboost and the other three quarter are direct matches. Tables 3.2 and 3.3 show the absolute numbers and proportions of all cars (private and commercial/ public) in each segment. There is a general trend of an increasing number of cars. The most remarkable trend is the increase in SUVs from about 3% in 2014 to 10% in 2023, whereas the proportion of small, medium and large cars all drop.

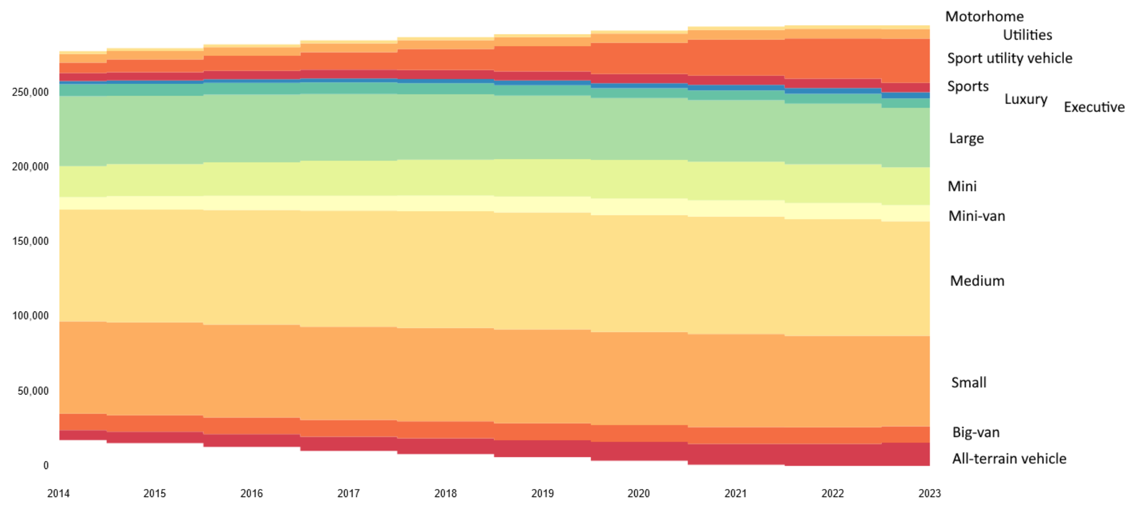


Figure 3.1: Streamgraph of car segments

Table 3.2: Segment Analysis (absolute frequencies)

Segment	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
ATV	6715	7625	8623	9442	10598	11311	12649	14002	14766	15547
Big-van	10884	11001	11161	11277	11411	11387	11297	11252	11078	10944
Small	61758	62106	62177	62281	62373	62815	62263	62228	61288	60588
Medium	74833	75454	76555	77703	78159	78245	78281	78542	77957	76692
Mini-van	8333	9028	9608	9963	10387	10627	10909	10934	10864	10652
Mini	20785	21380	22383	23451	24093	24988	25860	25821	25840	25336
Large	46708	45727	45247	44645	43727	42557	41464	41076	40563	39783
Executive	8019	8013	7888	7791	7405	6987	6604	6572	6583	6383
Luxury	2160	2326	2474	2691	2865	3130	3337	3746	3958	4091
Sports	5421	5449	5600	5834	5966	6088	6104	6246	6214	6265
SUV	6968	8568	10026	11701	14023	16809	20890	24108	26974	29614
Utilities	5723	5645	5758	5889	5925	6062	6149	6436	6397	6455
Motorhome	1836	1886	1902	1953	2021	2085	2129	2250	2313	2419

4 Visualizing data

Chapters 2 and 3 provide geocoded datasets for different groups of vehicles. The first step in analysing the data is to visualize them on a map. The naive approach is to divide the map into grid cells, count the vehicles in each grid, encode the counts into grouped colours and plot the resulting grid. This yields a rough / uneven map, which needs to be smoothed. Keep in mind, that vehicles move, and the smoothing is done in reality by people looking for parking lots and thereby spreading out the vehicle mass. A very common way to smooth an empirical distribution in a spatial context is to use a method called kernel density estimation.

A one dimensional kernel density estimator based on n samples $x_1, \dots x_n$ is de-

Table 3.3: Segment Analysis (relative frequencies in %)

Segment	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
ATV	2.60	2.90	3.20	3.40	3.80	4.00	4.40	4.80	5.00	5.30
Big-van	4.20	4.20	4.10	4.10	4.10	4.00	3.90	3.80	3.80	3.70
Small	23.70	23.50	23.10	22.70	22.40	22.20	21.60	21.20	20.80	20.60
Medium	28.80	28.60	28.40	28.30	28.00	27.60	27.20	26.80	26.40	26.00
Mini-van	3.20	3.40	3.60	3.60	3.70	3.80	3.80	3.70	3.70	3.60
Mini	8.00	8.10	8.30	8.50	8.60	8.80	9.00	8.80	8.80	8.60
Large	18.00	17.30	16.80	16.30	15.70	15.00	14.40	14.00	13.80	13.50
Executive	3.10	3.00	2.90	2.80	2.70	2.50	2.30	2.20	2.20	2.20
Luxury	0.80	0.90	0.90	1.00	1.00	1.10	1.20	1.30	1.30	1.40
Sports	2.10	2.10	2.10	2.10	2.10	2.20	2.10	2.10	2.10	2.10
SUV	2.70	3.20	3.70	4.30	5.00	5.90	7.30	8.20	9.20	10.00
Utilities	2.20	2.10	2.10	2.10	2.10	2.10	2.10	2.20	2.20	2.20
Motorhome	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.80	0.80	0.80

fined as (Wasserman, 2006, p. 132):

$$\hat{f}_n(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where $h > 0$ is the bandwidth and K is a kernel function that needs to satisfy (Wasserman, 2006, p. 55):

- $\int K(x)dx = 1$
- $\int xK(x)dx = 0$
- $\sigma_K^2 \equiv \int x^2 K(x)dx > 0.$

The choice of kernel function is of minor concern compared to the choice of bandwidth. This paper uses the commonly used Epanechnikov kernel defined as (Wasserman, 2006, p. 55):

$$K(x) = \frac{3}{4}(1 - x^2)I(|x| \leq 1).$$

Kernel density can be understood as drawing the kernel function above each sample, summing up the function values and dividing by the number of samples. The last step (dividing by the number of samples) will be skipped, because time series comparisons are made and the total number of vehicles differs between different years. Those differences are important and should not be discarded, which means following references to kernel density estimation will be made to the unnormalized density, which technically is not a density.

Figure 4.1 shows a simple example with five data points. Selecting the bandwidth h is the crucial part of any density estimation. If the distribution is multi-modal

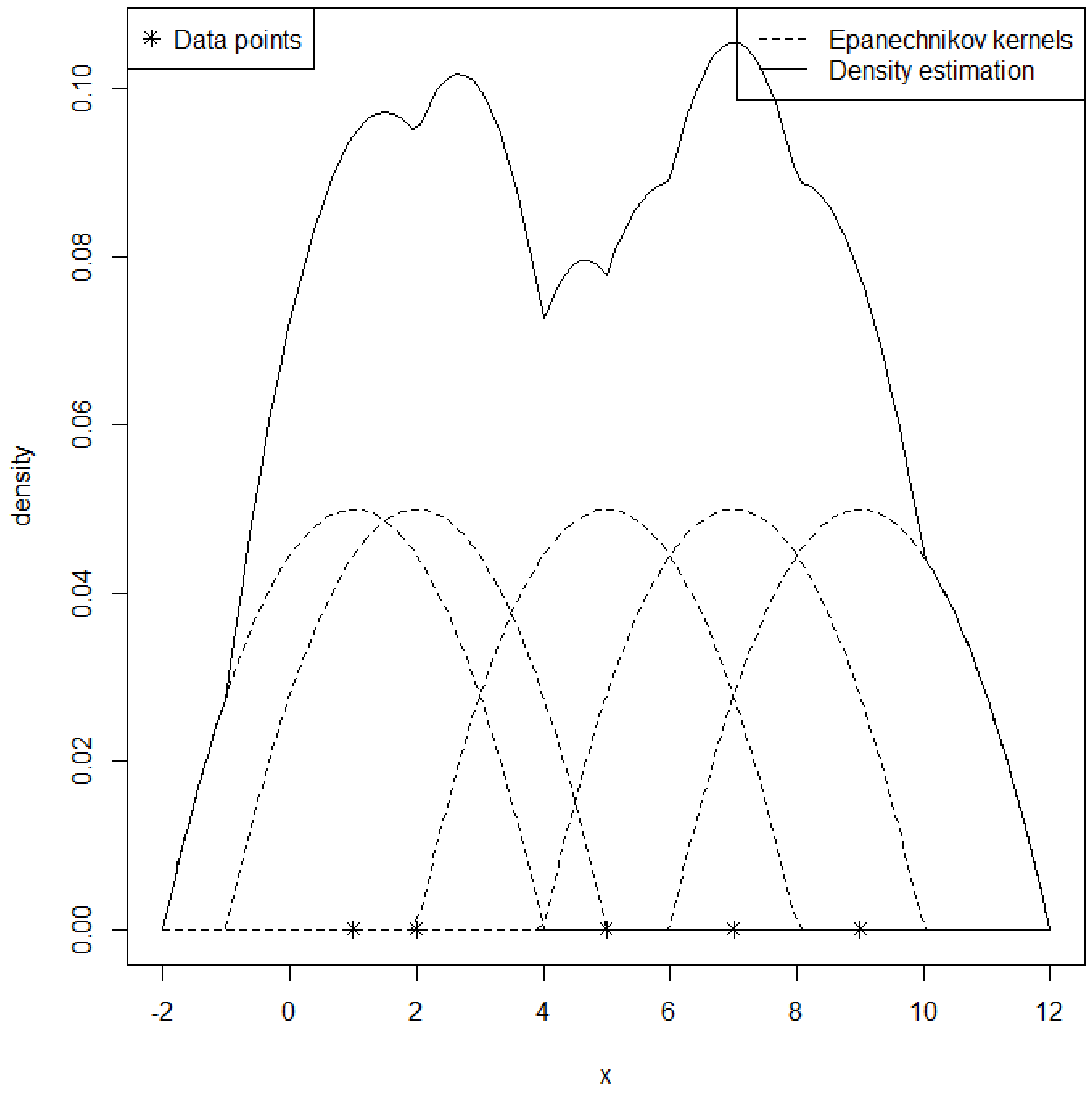


Figure 4.1: Example kernel density estimation

(which is often the case in spatial data, see below) there may not be one global bandwidth, which produces sufficient results everywhere. To overcome this limitation, local adaptive bandwidths can be used, i.e.

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_i}\right),$$

the bandwidths h_i are individually selected for each point. The general idea is to require less smoothing where many samples are located and to require more smoothing where fewer samples are located, i.e. choosing the bandwidth inversely to the unknown density. The following sections generalize the kernel density estimation to two dimensions, by plugging the ellipse equation into above's kernel density formula and finding methods to adaptively scale the ellipsis.

4.1 Delaunay tessellation

The key to local adaptive bandwidth density estimation is finding how much density mass is around a point and scaling the bandwidth inversely to that mass. This can be done e.g. by triangulation the convex hull of the coordinate points by a Delaunay tessellation. Delaunay's method can be thought of as calculating the area around a point till the next point. Every point is one address. It does not deal with the situation where multiple cars share an address. This will be covered later on. The following steps are illustrated by a simple toy example. Figure 4.2 shows a map with

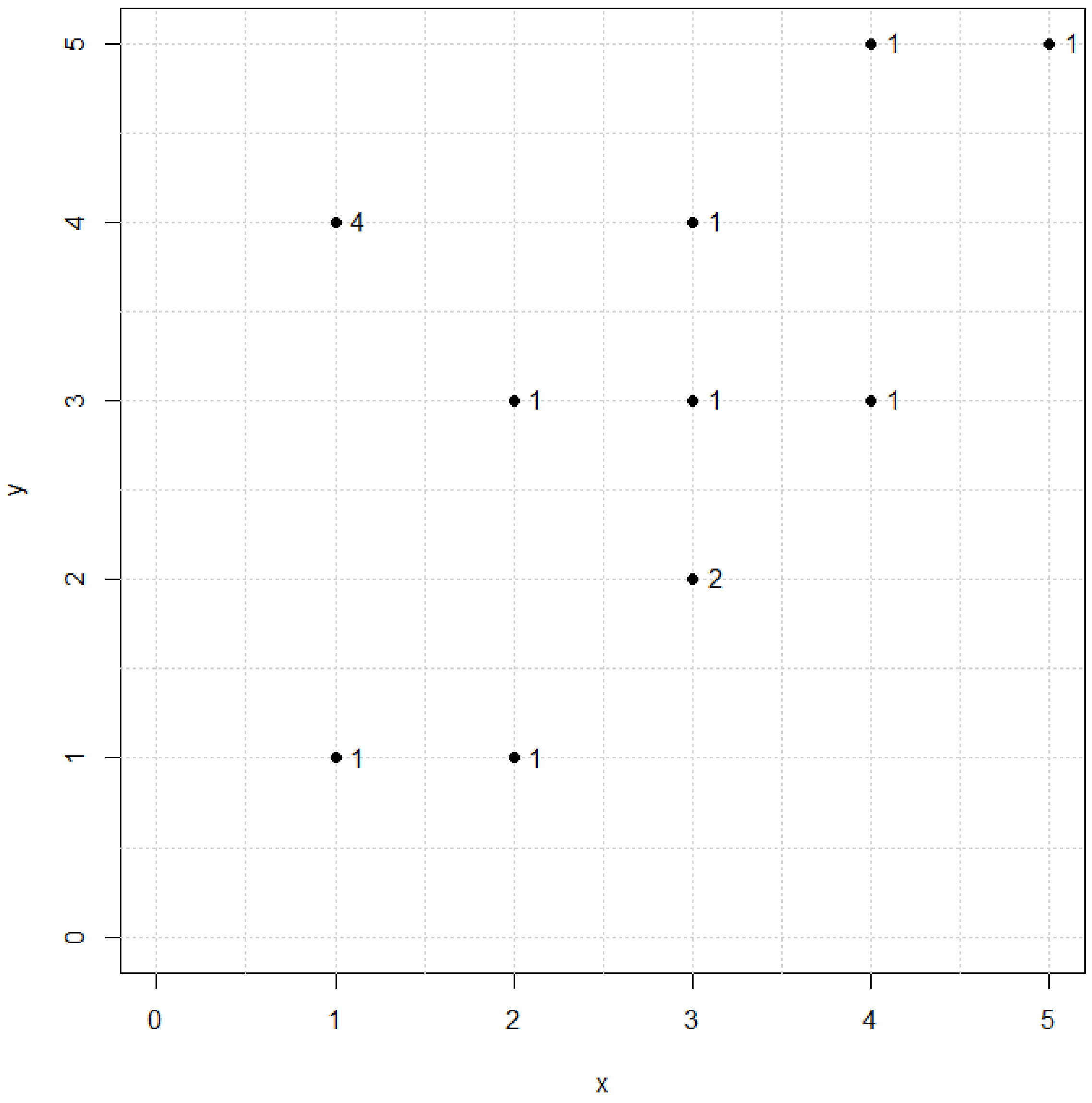


Figure 4.2: Coordinates of vehicles

10 coordinate points. The number next to each point is the number of vehicles at that coordinate, e.g. two vehicles are at the coordinate $x = 3$ and $y = 2$. The first step is to calculate the Delaunay triangulation of the coordinates.

To provide some intuition to the Delaunay triangulation, one should first consider the Voronoi / Dirichlet tessellation. Let $P = \{p_1, \ldots, p_n\}$ with $p_i \in \mathbb{R}^2$ $1 \leq i \leq n$ be the coordinate points which $n \geq 3$ and not all P collinear or four points co-circular. The Voronoi polygon of point p_i is defined by (D. Lee and Schachter, 1980)

$$V(i) = \{x \in \mathbb{R}^2 | d(x, p_i) \leq d(x, p_j), \ \forall i \neq j \text{ and } 1 \leq i, j \leq n\}.$$

This can be thought of a process where circles start simultaneously expanding at the same rate at all points until they make contact with the neighbouring cells. Voronoi polygons for the points on the convex hull of the coordinates continue to infinity in the direction away from the convex hull. The contact lines between Voronoi polygons are called Voronoi edges and the knots are called Voronoi vertices. In a two-dimensional space, a Voronoi edge is always created by two coordinate points. Each point is on one side of the edge and all points on the edge have the same distance to the forming points. The Voronoi vertices are created where three Voronoi edges meet. By construction, the Voronoi vertex is equidistant to the forming points of its Voronoi edges. This implies that no other coordinate point can be within the circle centred at the Voronoi vertex and having a radius equal to the distance between the vertex and its three forming points.

Connecting the three forming points leads to a triangle within the circle around the vertex. This triangle is called a Delaunay triangle. Applying the aforementioned steps to all Voronoi vertices creates a tessellation of the convex hull of all coordinate points made out of triangles (D. Lee and Schachter, 1980). Figure 4.3 shows the Voronoi polygons and the Delaunay tessellation for the toy example. The solid lines show the Delaunay tessellation and the dashed lines show the Voronoi cells. This plot has been made with the R-package `deldir` (Turner, 2021).

The vertices of those triangles are always coordinate points. In order to use the triangulation for bandwidth adoption, all triangles from the Delaunay tessellation that share a certain coordinate point and all other coordinate points that establish those triangles are selected. Then the minimum volume encircling ellipse around those coordinate points is calculated and fed into the Epanechnikov kernel.

But first, the Delaunay tessellation needs to be calculated. The algorithm to construct the Delaunay tessellation by Bowyer, 1981 and Watson, 1981 is an iterative process. The following algorithm is taken from Sloan and Houlsby, 1984. It starts by defining a "super triangle" that encloses the convex hull of all coordinate points. Each point is than added one by one.

1. Sort $(p_1, \ldots, _n)$ in ascending order of their x-coordinate.

2. Define the vertices of the super triangle in anticlockwise order and flag the super triangle as incomplete.

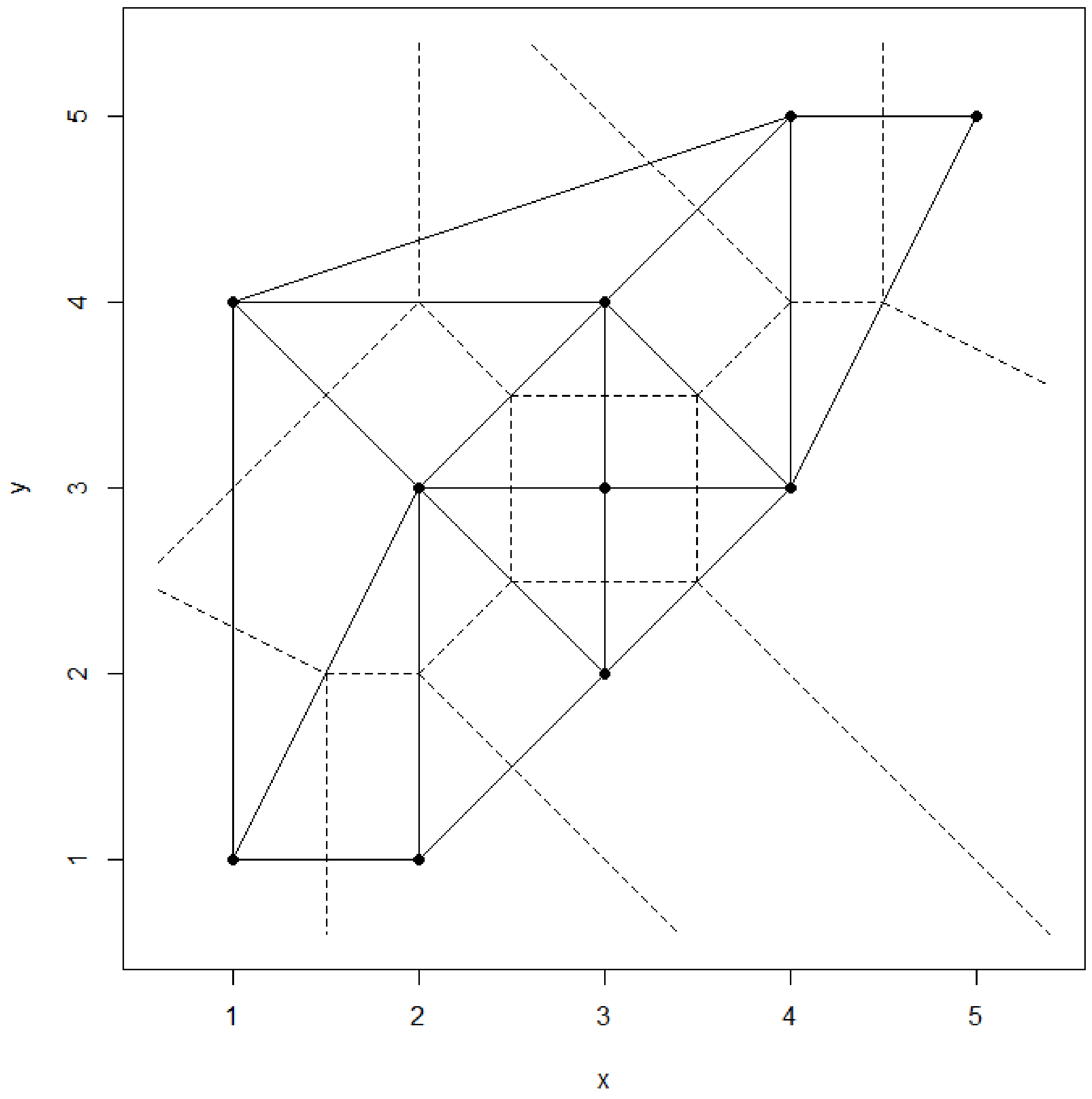


Figure 4.3: Voronoi and Delaunay tessellations

3. Repeat steps 4 to 12 until the list of points to be triangulated is exhausted.

    4. Add new point with co-ordinates $(X_{\text{new}}, Y_{\text{new}})$.

    5. For each triangle which is flagged as incomplete, do steps 6 to 10.

        6. Compute the coordinates of the triangle circumcentre, $(X_c, Y_c))$, and the square of its circumcircle radius R^2.

        7. Compute the square of the x-distance from the new point to the triangle circumcentre $D_x^2 = (X_c - X_{\text{new}})^2$.

        8. If $D_x^2 \geq R^2$, then the circumcircle for this triangle, flag this triangle as complete and do not execute steps 8 and 9.

        9. Compute the square of the distance from the new point to the triangle circumcentre $D^2 = D_2^x + (Y_c - Y_{\text{new}})^2$.

10. If $D^2 < R^2$ delete this triangle from the list of triangles formed and store the three pairs of vertices which define its edges on a list of edges. If $D^2 \geq R^2$, then the triangle remains unmodified.

11. Loop over the list of edges and delete all edges which occur twice.

12. Form the new triangles by matching the new point with each pair of vertices in the last of edges. The new point forms new triangles with each pair of vertices on the boundary of the polygon formed by the intersected triangles. Define each new triangle such that its vertices are always listed in an anticlockwise sequence and flag at as incomplete.

13. Form the final triangulation by removing all triangles which contain one or more of the super triangle's vertices.

The following analysis uses the R-package `delaunay` (Laurent, 2022). Note that the Sloan's algorithm presented above runs extremely efficient in terms of CPU usage and memory requirements even for very large datasets. Common kernel density implementations for example require massive amounts of memory as will be shown later.

4.2 Minimum area enclosing ellipse

The next step is to locate all the Delaunay triangles, that a certain coordinate point is a member of. E.g. point (2,3) is part of the six triangles $T_1 = \{(2,1),(1,1),(2,3)\}$, $T_2 = \{(1,1),(1,4),(2,3)\}$, $T_3 = \{(1,4),(3,4),(2,3)\}$, $T_4 = \{(3,3),(3,4),(2,3)\}$, $T_5 = \{(3,3),(3,2),(2,3)\}$ and $T_6 = \{(3,2),(2,1),(2,3)\}$. Those six triangles are made from seven distinct points $\{(2,3),(2,1),(1,1),(1,4),(3,4),(3,3),(3,2)\}$. The next step is calculating a minimum area enclosing ellipse (MAEE) around the convex hull of those points. An ellipse in centre form is defined as

$$E = \{x \in \mathbb{R}^2 | (x-c)'A(x-c) \leq 1, \text{ with } c \in \mathbb{R}^2 \text{ and } A \in \mathbb{R}^{2\times 2} \text{ symmetric and } \det(A) > 0\}.$$

Its area is $\frac{\pi}{\sqrt{\det(A)}}$. This leads to an optimization problem minimizing the area with the restriction of all points lying inside the ellipse. Moshtagh, 2005 show the required steps to solve this problem. In mathematical terms

$$\min \det(A^{-1}) \text{ subject to } (x_i - c)'A(x_i - c) \leq 1 \ \forall 1 \leq i \leq n \text{ and } A > 0,$$

which can also be expressed alternatively by parametrizing the ellipse as

$$E = \{x \in \mathbb{R}^2 : ||Ex - b|| \leq 1 \ \forall 1 \leq i \leq n \text{ and } A > 0\}$$

with $E = A^{1/2}$ and $b = A^{1/2}c$. The optimization problem becomes

$$\min \text{logdet}(E^{-1}) \text{ subject to } ||Ex_i - b|| \leq 1 \ \forall 1 \leq i \leq n.$$

Finally, it is easier to work with an ellipse centred at the origin. This can be achieved by a coordinate transformation $q_i = (x_i', 1)$ into $\mathbb{R}^3$.

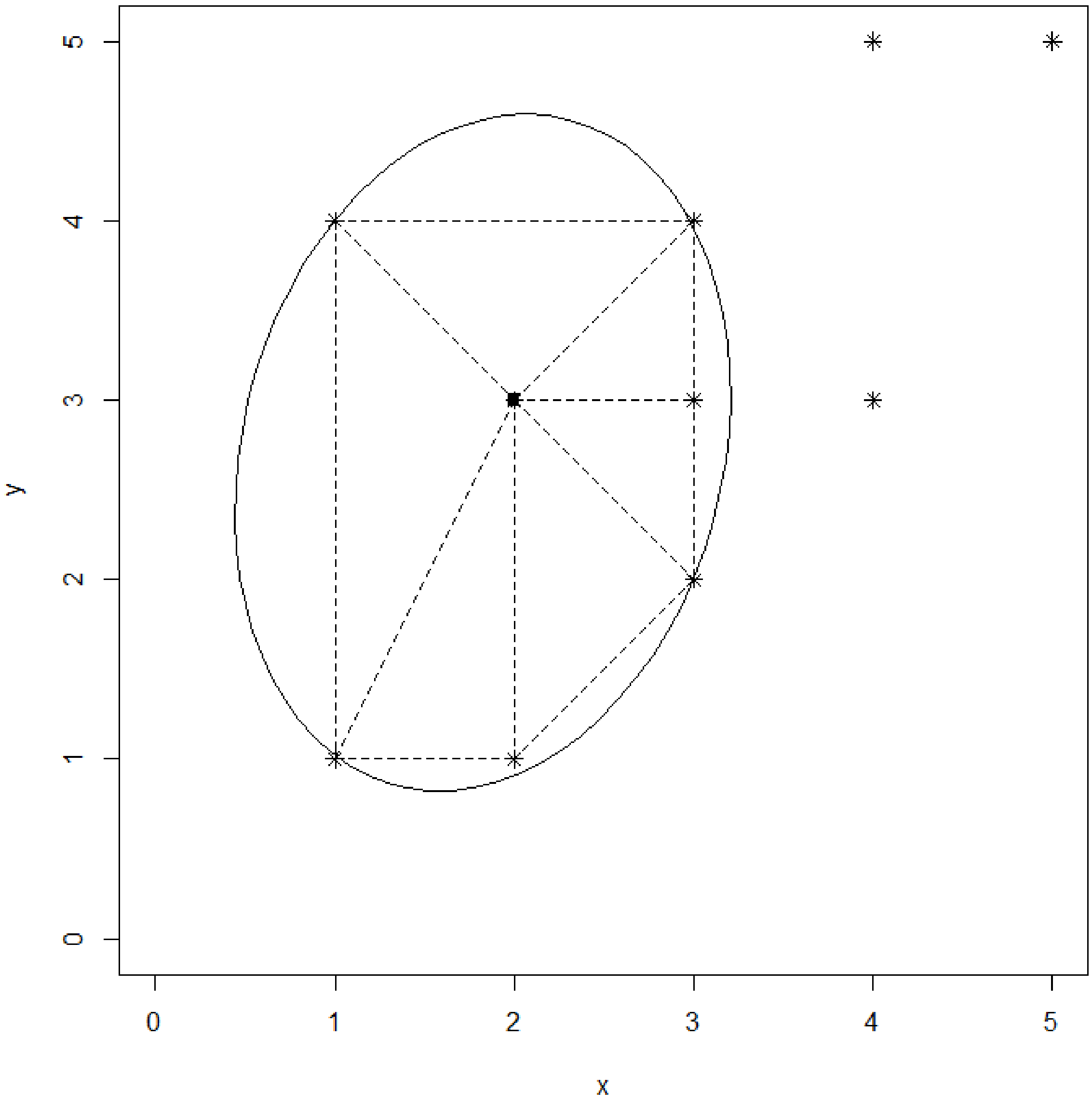


Figure 4.4: Minimum area enclosing ellipse

Khachiyan, 1996 provides the following optimized algorithm to solve the above optimization problem:

1. Set

$$Q = \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \\ 1 & 1 & 1 \end{pmatrix} \text{ and } u = \begin{pmatrix} 1/n & \dots & 1/n \end{pmatrix}$$

2. Repeat steps 3 to 9 until err is below threshold

3. $X = Q * \mathrm{diag}(u) * Q'$
4. $M = \mathrm{diag}(Q' * X^{-1} * Q)$
5. max = maximum(M) and j is the corresponding index to max
6. step = $\frac{\max - 3}{3\max - 3}$
7. $u_{\text{new}} = (1 - \text{step})u$
8. $u_{\text{new}}[j] = u_{\text{new}}[j] + \text{step}$
9. $u = u_{\text{new}}$ and err $= ||u_{\text{new}} - u||$
10. $A = \frac{1}{2} * [P * \mathrm{diag}(u) * P' - (P * u) * (P * u)']^{-1}$ and $c = P * u$

Figure 4.4 shows the resulting ellipse for point (2,3). The R implementation by (Lyons, 2011) is used. This process is iterated over all points and the resulting ellipses are centred at its corresponding coordinate points. This is especially important for coordinates on the convex hull of all points. Their ellipses' centres can be relatively far away, so the ellipses are recentred to avoid biases. Before the ellipse equations can be fed into the Epanechnikov kernel function and summed up, some adaptations are still needed.

4.3 Local adoptive kernel density estimation

Applying the above-mentioned method without further modifications leads to serious "under smoothing". Therefore, a global smoothing parameter is calculated, and the ellipses are scaled in such a way that the median ellipse area is equal to the area determined by the global smoothing estimate. Furthermore, the vehicle count at each coordinate needs to be included in the scaling process as well as making sure each ellipse covers at least one grid cell on the one hand, and is not unreasonably large on the other hand.

The global bandwidth is calculated for the x- and y-coordinates separately. In general, there are two commonly used methods to determine the smoothing bandwidth based on the data. Recall the univariate kernel density estimator (Wasserman, 2006, p. 132)

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right).$$

The objective is to find some h, that minimizes some measure of the difference between the estimated density $\hat{f}$ and the true f called MISE:

$$\mathrm{MISE}(h) = \mathbb{E}\left[\int [\hat{f}_h(x) - f(x)]^2 dx\right].$$

MISE depends on the unknown f. The first way to solve this issue is to use a monte-carlo simulation. This approach is referred to as “cross-validation”. Expanding MISE yields

$$\text{MISE}(h) = \mathbb{E}\left[\int \hat{f}_h^2(x)dx - 2\int \hat{f}_h(x)f(x)dx + \int f^2(x)dx\right].$$

The first term can be directly calculated, and the third term does not depend on h. The interesting part is the second term, which is estimated by cross-validation (Wasserman, 2006, p. 127):

$$\int \hat{f}_h(x)f(x)dx = \int \hat{f}_h(x)dF(x) \approx \int \hat{f}_h(x)dF_n(x) \approx \frac{1}{n}\sum_{i=1}^{n} \hat{f}_{(-i)}(X_i),$$

where $F_n(x)$ is the empirical cumulative density function and $\hat{f}_{(-i)}$ is the density estimate on the data without the i-th observation

$$\hat{f}_{(-i)}(x) = \frac{1}{n-1}\sum_{j=1, i\neq j}^{n} \frac{1}{h}K\left(\frac{x-X_j}{h}\right).$$

There is no closed form solution to calculate the optimal h. It has to be done by numeric optimization. Unbiased cross-validation is implemented in the R-function `bw.ucv`, which assumes a Gaussian (normal) kernel and calculates the standard deviation instead of h. For the Epanechnikov-kernel the bandwidth is $\sqrt{5}$ times the standard deviation. The results need to scaled accordingly.

The second approach to find h is called plug-in estimation and is based on a taylor series expansion of the variance and bias of MISE (Sheather, 2004)

$$\text{MISE}(h) = \int \text{Bias}(\hat{f}_h(x))^2 dx + \int \text{Var}(\hat{f}_h(x))dx,$$

which results in an asymptotic MISE (AMISE) (Wand and Jones, 1994)

$$\text{AMISE}(h) = \frac{R(k)}{nh}R(k) + \frac{h^4}{4}\sigma_K^4 R(f''), \text{ with } R(K) = \int K^2(x)dx,\ \sigma_K^2 = \int x^2 K(x)dx.$$

Taking the derivative of AMISE and solving for the optimal h yields (Heidenreich et al., 2013)

$$h_{\text{AMISE}} = \left[\frac{R(K)}{\sigma_K^4 R(f'')n}\right]^{\frac{1}{5}}.$$

Instead of estimating f, it is needed to estimate the unknown second derivative f''. This requires itself some density estimation, i.e. a pilot bandwidth. Let $\hat{R}(f'')$ be the estimator of $R(f'')$, that will be plugged into the above equation (Sheather and

Jones, 1991)

$$\hat{R}(f'') = \frac{1}{n^2\alpha^5} \sum_{1\leq i,j\leq n}^{n} K\left(\frac{X_i - X_j}{\alpha}\right).$$

$\hat{R}(f'')$ depends again on a bandwidth α. Sheather and Jones propose to estimate α by

$$\alpha = \left(\frac{2K(0)}{\sigma_K^2}\right)^{\frac{1}{7}} R(f''')^{-\frac{1}{7}} \frac{1}{n^7}.$$

Their procedure is implemented in the R-function `bw.SJ`. At this point, the process is aborted, because the new problem is to estimate $R(f''')$, which runs into the same problem as estimating $R(f'')$. Instead, a normal rule of thumb is used to estimate α (Sheather, 2004). This leads to the third and usually not recommended way.

Silverman's rule of thumb (Silverman, 1986) estimates the bandwidth as

$$\hat{h} = 0.9 \times \min\left\{\frac{\text{IQR}}{1.34}, \text{sd}\right\} n^{-\frac{1}{5}},$$

where sd is the standard deviation and IQR is the distance between the 75% and the 25% quantile. Silverman's rule is implemented in the R-function `bw.nrd0`

All of those approaches can be used to estimate the pilot bandwidth. It is therefore needed to have some visual inspection. Keep in mind, that the global bandwidth should be rather smooth. The more interesting local peaks will be covered by the local bandwidth adaptations. To find some good recommendations, the toy example will be disregard for a moment and the real data is used. Figure 4.5 shows the x-coordinates of privately owned vehicles in 2023. The three lines show kernel density estimations with bandwidths selected by cross-validation, plug-in and rule of thumb methods. The lines are shifted in y-direction to separate them. The true y-values are of no concern at this stage, as only the smoothing level needs to be addressed. The plug-in as well as the cross-validation produce similar results, which both lead to under smoothed results. Silverman's rule of thumb is therefore selected as global bandwidth scaling method. Based on the global bandwidths, scaling of the ellipses' area takes part in four stages:

1. divide the area by the number of cars at that coordinate
2. scale the areas so that the median area of all ellipses matches the area of the global bandwidth ellipse
3. apply minimum scaling (every ellipse must cover at least one grid cell)
4. apply maximum scaling (no ellipse should be bigger than 20 times the minimum ellipse)

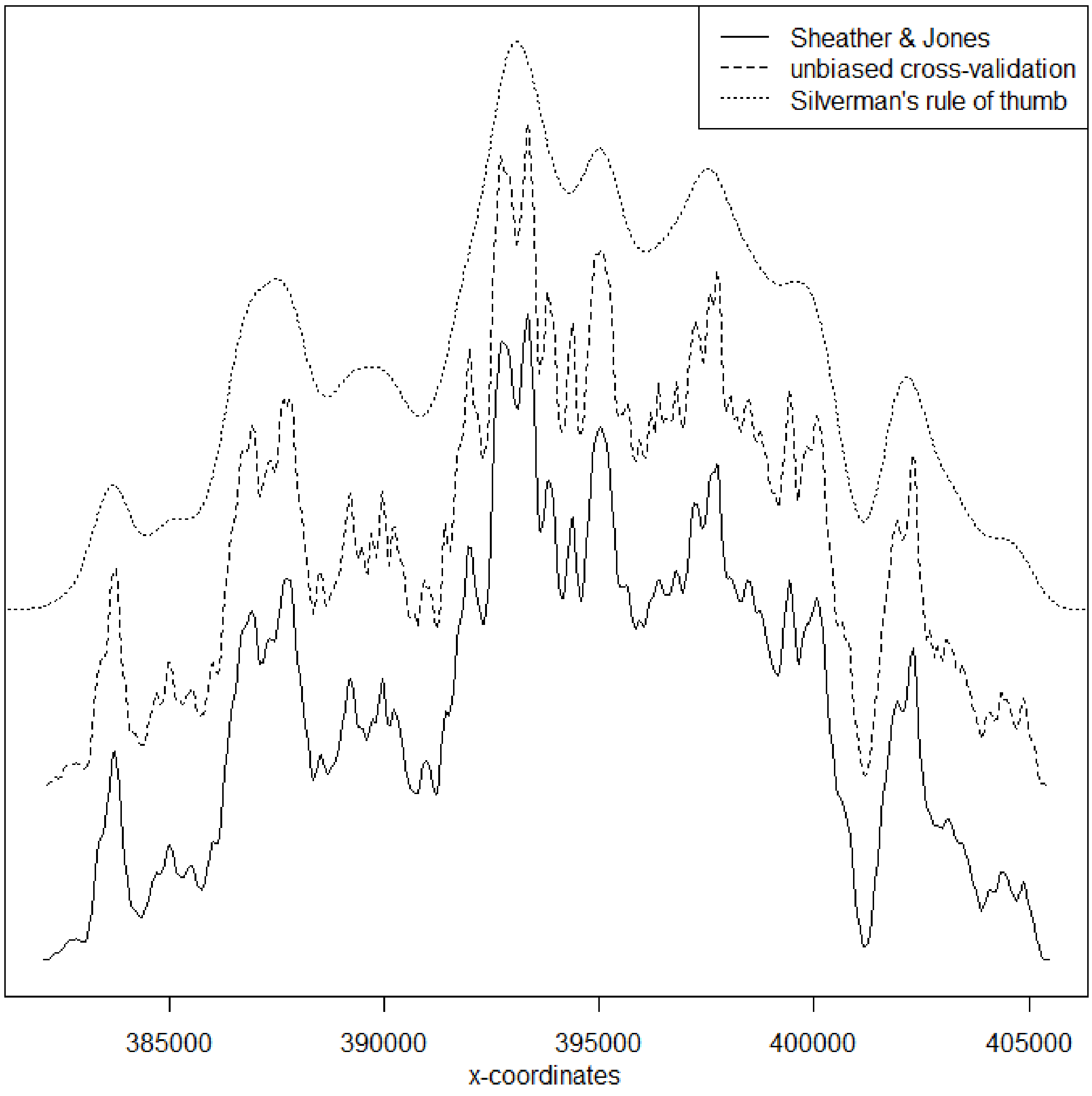


Figure 4.5: Global bandwidth selectors

Scaling an ellipse is done in terms of the eigenvalues of its matrix A. Figure 4.6 shows an ellipse and its two semi-axes. The eigenvectors of A define the direction of the semi-axes and $\frac{1}{\sqrt{\lambda}}$, with λ being the corresponding eigenvalue, is the length of the semi-axis. Scaling the ellipse is done by scaling both eigenvalues proportionally according to the four-step procedure described above and reconstruction of the A matrix based on the scaled eigenvalues λ_1', λ_2':

$$A_{\text{scaled}} = U\text{diag}(\lambda_1', \lambda_2')U^{-1}, \text{ with U as a matrix with A's eigenvectors.}$$

This section ends with comparing the local adoptive method with the global method. The main aim is to allow for more detailed information where the density is high while still retaining the smoothing properties in areas with low density.

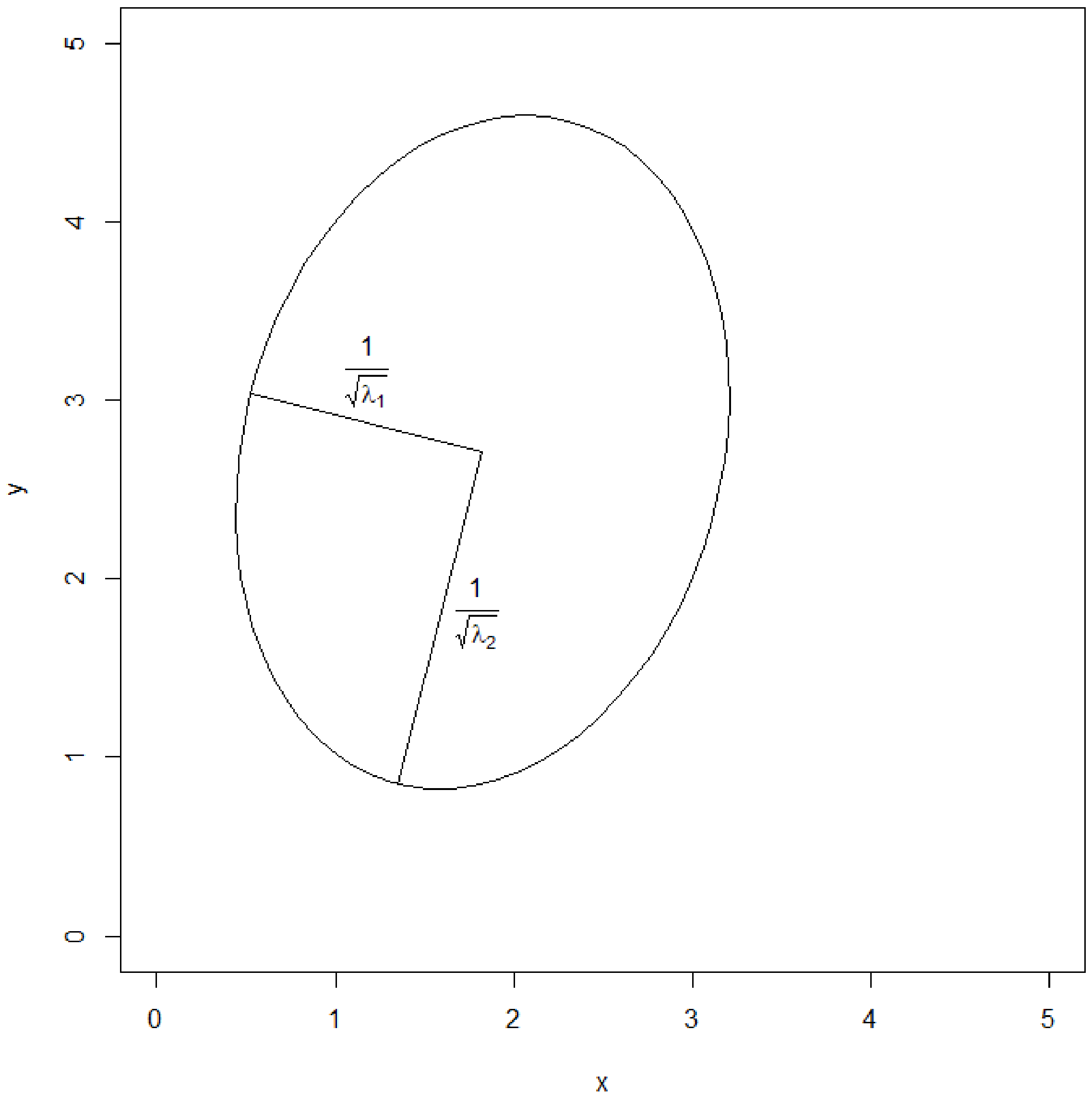
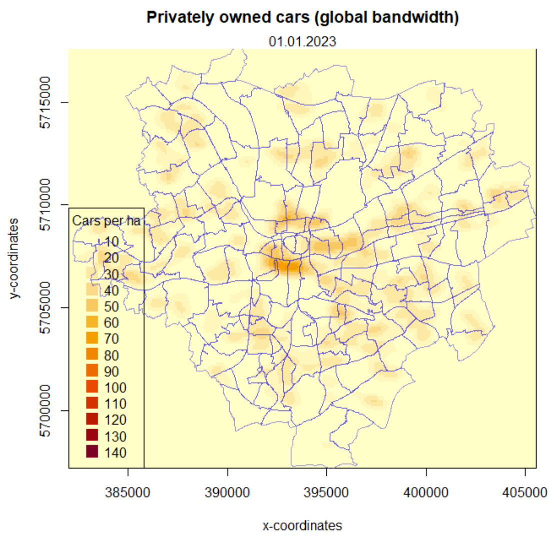
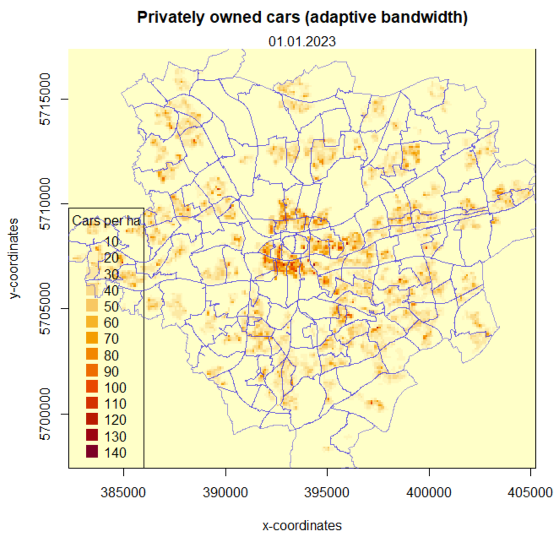


Figure 4.6: Eigenvalues and eigenvectors of ellipse

Figure 4.7 show the results. The blue lines indicate Dortmund's boroughs. This is usually the highest available spatial resolution in municipal statistics. Both figures in 4.7 use a grid size of 100m*100m and the same global bandwidth. The left figure 4.7a is created with the R-package `MASSExtra` by (Venables, 2023) and the right figure 4.7b by the method described above. The right graph provides much more details compared to the left graph, which means the main goal has been achieved. Furthermore, it can be seen, that Dortmund actually consists of an inner city core and a bunch of smaller former "villages", that have been included into the city's boundaries over a timespan of centuries. Some caveats need to be mentioned. The adoptive method has a much longer computation time. This is partly due to missing optimizations. The global method has the disadvantage of requiring huge amount of memory. Figure 4.7a took about 50GB of ram, whereas the right figure only requested less than 16GB of ram. No further comparisons are made, because figure



(a) Global bandwidth



(b) Adoptive bandwidth

Figure 4.7: Comparison of methods

4.7a required access to different hardware.

5 Providing data

Kernel density estimation certainly produces information dense graphs for reports and can be used as a starting point for deeper analysis. Those can be done either by the statistical office itself or by interested third parties. A key part of official statistics is providing accurate data to those third parties without sacrificing the individual's privacy. Strong and state-of-the-art confidentiality protection is a legal requirement in almost all countries. Currently, many different data altering methods are used to provide microdata. A good disclosure avoidance system should

- provide quantifiable confidentiality protection
- keep data alteration to a minimum
- be unbiased
- be transparent (e.g. no secret parameters or methods).

Unfortunately many systems currently in use by statistical offices are black boxes and lack quantifiable protection guarantees. The cell-key method used by the German statistical office for the Census 2022 relies on secret parameters, which thwart any in-depth inspections. Those "security by obscurity" methods are wildly regarded as insufficient and insecure. A new method called "Differential privacy" by (Dwork et al., 2006) which is already used by the 2020 U.S. census, addresses those shortcomings.

5.1 Differential privacy

The differential privacy idea is based on an attack model. Suppose the attacker already knows the complete data except whether one single individual is in the data or not. Let x and x' be two databases, that differ on at most one individual. The attacker is interested in $T(x) = \sum(x_i)$, the total number of individuals. $M(T(x))$ is called a random ε-differentially private data alteration function iff:

$$\left| \ln \left(\frac{\Pr[M(T(x)) \in S]}{\Pr[M(T(x')) \in S]} \right) \right| \leq \varepsilon$$

for some $\varepsilon > 0$ and $\forall\ S \subseteq$ Image(M) (Dwork et al., 2006, p.6)

ε is called the privacy budget and allows quantifying confidentiality requirements. The smaller ε the more privacy is guaranteed and the fewer data utility is provided. From the definition two main questions arise.

- What is the random function M?
- How should ε be chosen?

Let ΔF be the sensitivity of T:

$$\Delta F = \max |T(x) - T(x')|.$$

Then

$$M(T(x)) := T(x) + Y, \qquad \text{where } Y \sim \text{Lap}\left(\frac{\Delta F}{\varepsilon}\right)$$

is a ε-differentially private mechanism (Dwork et al., 2006, p.6), where Lap$(\frac{\Delta F}{\varepsilon})$ refers to the double exponential or Laplace distribution with mean 0 and scale parameter $\frac{\Delta F}{\varepsilon}$. M works by adding laplacian noise to the true result in order to obscure the difference between $T(x)$ and $T(x')$. If T is the total number of individuals, T's sensitivity is obviously 1. The same result holds for density estimation. In this case a car's or individual's density mass would be concentrated in a single grid cell. In general a laplace distribution with mean μ and scale parameter b has

- pdf: $f(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$
- cdf: $F(x) = \frac{1}{2} + \frac{1}{2}\text{sgn}(x-\mu)\left(1 - \exp\left(-\frac{|x-\mu|}{b}\right)\right)$
- qf: $F^{-1}(p) = \mu - b\ \text{sgn}(p - \frac{1}{2})\ln(1 - 2|p - \frac{1}{2}|)$ (Abramowitz et al., 1988).

An example density plot of a laplace distributed random variable is provided in figure 5.1. Adding laplacian noise creates a problem with count data or densities. If $T(x)$ is close to zero, adding laplacian noise can lead to negative values, which are outside the target range.

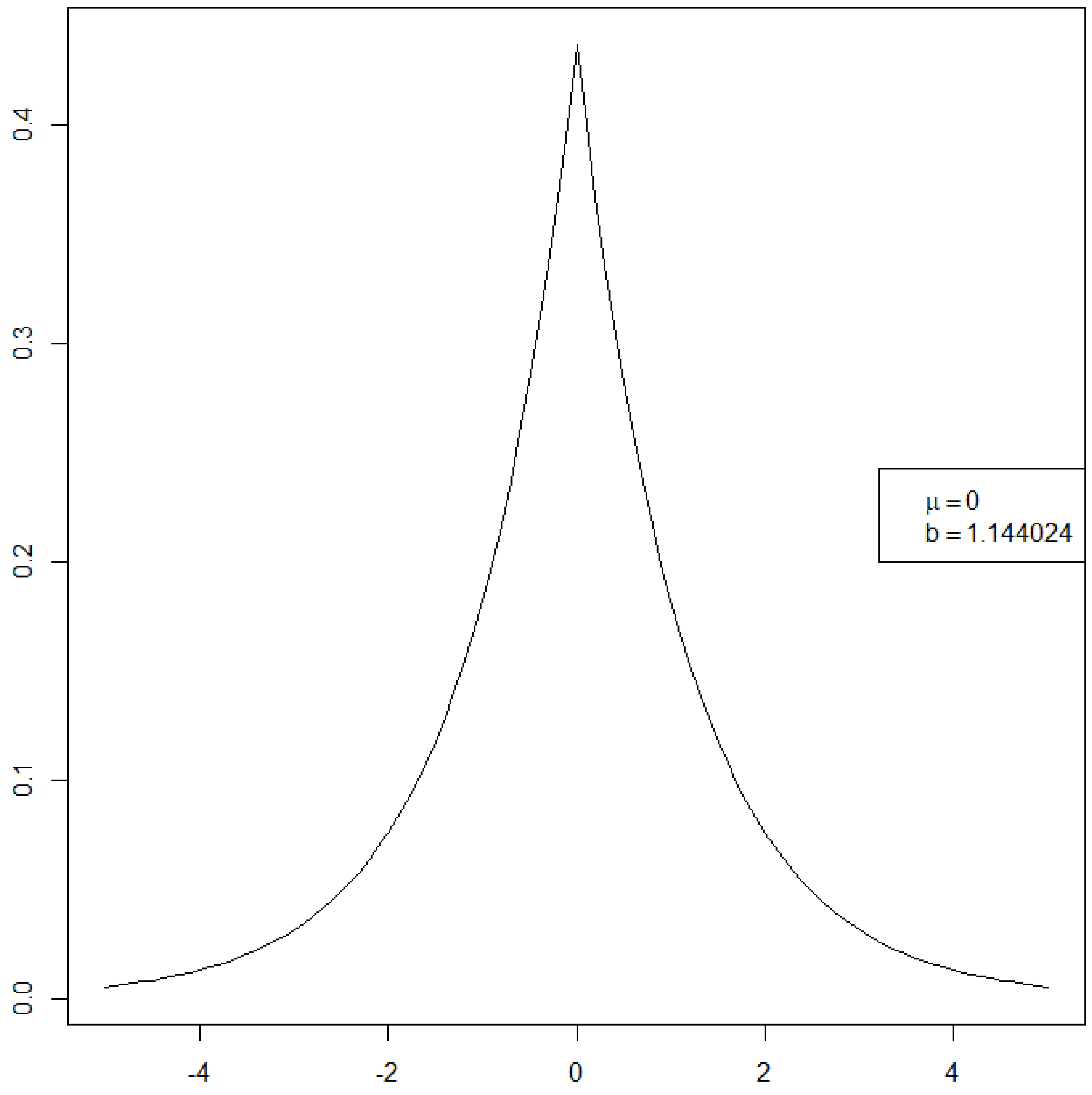


Figure 5.1: Laplace distribution density

5.2 Choosing ε

Unfortunately, the definition of differential privacy is not helpful in directly determining the right value for ε. Instead, ε should be related to some other easy to interpret measure. J. Lee and Clifton, 2011 relate ε to the probability of an attacker guessing the right value. Their approach can only be used for discrete/ count data, as the probability of guessing a realization of a continuous random variable is always zero. Therefore, this does not work for continuous density estimates. Pankova and Laud, 2022 expand on J. Lee and Clifton, 2011 ideas and apply them to continuous data. Their idea is to calculate the attacker's guessing advantage. The prior probability of guessing right within a distance of r is called $\Pr_{\text{pre}}(x)$. Let x be the full dataset, x' the dataset with one observation missing, d some distance measure, $r > 0$, $B(x,r) := \{y \in \mathbb{R} | d(x,y) \leq r\}$ and T the statistic of interest. The prior probability is

$$\mathrm{Pr}_{\mathrm{pre}}(x) := \mathrm{Pr}(T(x') \in \mathcal{B}(T(x), r)).$$

The posterior probability $\mathrm{Pr}_{\mathrm{post}}(x)$ is the probability of guessing right after having the additional information on x through the differentially private mechanism M

$$\mathrm{Pr}_{\mathrm{post}}(x) := \mathrm{Pr}(T(x', M(T(x))) \in \mathcal{B}(T(x), r)).$$

The aim is to limit

$$|\mathrm{Pr}_{\mathrm{post}}(x) - \mathrm{Pr}_{\mathrm{pre}}(x)| \leq \gamma$$

J. Lee and Clifton, 2011 show this is achieved if

$$\varepsilon \leq \frac{-\ln\left(\frac{p}{1-p}\left(\frac{1}{\gamma+p} - 1\right)\right)}{\Delta F}$$

with $p = \mathrm{Pr}_{\mathrm{pre}}(x)$. Because $\mathrm{Pr}_{\mathrm{pre}}(x)$ is unknown, taking the derivative of the right term with regard to p yields a minimum at $p = \frac{1-\gamma}{2}$.

Setting $\delta = \frac{1}{3}$ provides appropriate privacy protection for the density data. Figure 5.1 shows the resulting laplace noise for grid cells having a density of zero. The probability of the noise to be between -4 and 4 is 96.9%. This gives an idea on the magnitude for the added noise, though sometimes much higher / lower values are possible. When plotting on a map with bin size 10 cars per ha, the plots of the actual data and the confidentiality protected data are indistinguishable.

Other values are certainly possible and need to be addressed by the legal departments. Having chosen ε the remaining problem regards the potential for negative values.

5.3 Truncating laplace noise

Densities are always non-negative between $[0, \infty)$, whereas the laplace distribution's domain is $(-\infty, \infty)$. Adding laplacian noise to a density may lead to negative and therefore impossible values. The obvious solution to this problem is to draw the noise instead from a to $(0, \infty)$ truncated laplace distribution. This causes two new problems:

- reduced randomness around small values due to truncation that has to be compensated to uphold differential privacy safeguards, and
- positively biased random noise leads to biased results.

To compensate the first issue, the laplacian noise must be drawn from a distribution with higher variance the closer the density is to zero. Croft et al., 2022 provide a formula for the needed variance adaptations. This formula requires the lambert W function, which is the solution for z in the equation $z = x\ \exp(x)$. Figure 5.2 shows the graph of the lambert W function. It has two branches, the zero branch W_0 and the minus-one branch W_{-1}. They meet at the point $x = -\frac{1}{\exp(1)}$ and $y = 1$. If $x \geq 0$, there is one solution for $z = W_0(x)$. If $-\frac{1}{\exp(1)} \leq x < 0$, there are two solution $z_1 = W_0(x)$ and $z_1 = W_{-1}(x)$. If $x < -\frac{1}{\exp(1)}$ there is no solution (Corless et al., 1996). Numerical evaluation of the lambert W function is implemented in the R-package “lamW” (Adler, 2015). According to Croft et al., 2022, the scale

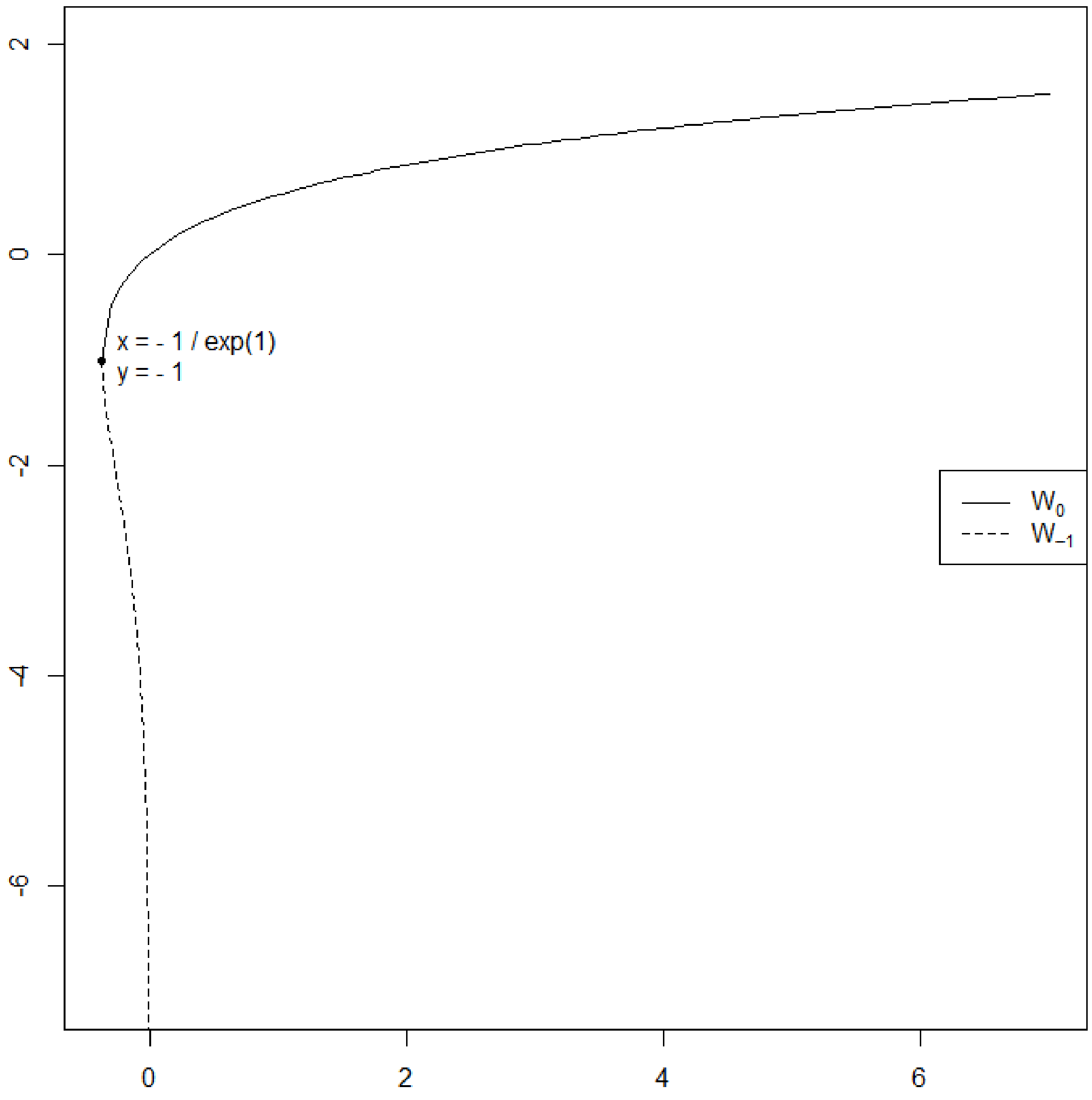


Figure 5.2: Lambert W function

parameter of the laplace distribution for a restraint to the interval $(0, \infty)$ should be

$$b_1 = -\frac{\Delta F}{W\left(-\frac{1}{2\exp(1)}\right)\exp(1)\varepsilon}$$

at zero and at a point $i\Delta F$ away from zero by

$$b_2 = -\frac{i\Delta F \exp(i\varepsilon)\sigma_1}{W_Z\left(-\frac{2i\Delta F \exp(-i\varepsilon)\exp\left(-\frac{i\Delta F \exp(-i\varepsilon)}{\sigma_1}\right)}{\sigma_1}\right)\exp(i\varepsilon)\sigma_1 + i\Delta F},$$

where W_Z with $Z \in \{0, -1\}$ denotes either the minus-one branch or the zero branch. To determine the correct branch, one has to find the i for which the term inside W_Z yields $-1/exp(1)$. This is done by a simple numerical optimization (R's built-in optimize function). Let cp be the result of above's optimization. If $i \leq$ cp Z should be zero and if $i >$ cp Z should be -1.

Let's have a look at an example for $\delta = 1/3$ and $\Delta F = 1$. Figure 5.3 shows on the

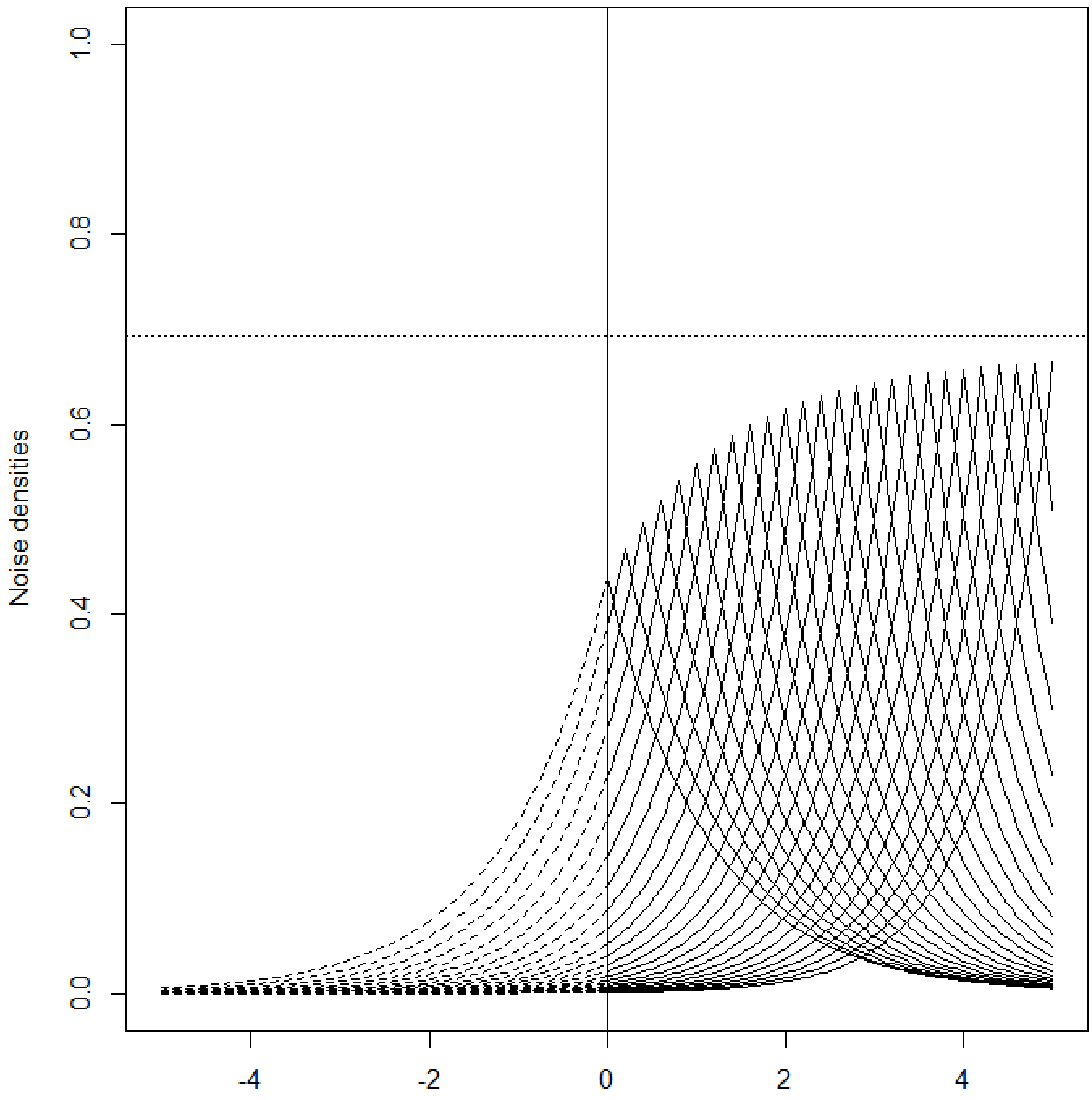


Figure 5.3: Example noise distributions

x-axis the true density value of a grid cell. On the y-axis the density of the laplace noise to be added is shown. The closer the x values are to zero, the more variance the laplace distribution gets as desired. The horizontal line at $y = 0.69$ is the density at μ for a $\text{Lap}(\mu, \frac{1}{\varepsilon})$ distributed random variable. This is the distribution for $i \to \infty$. The further the x values are away from zero, the closer the density peaks reach the limiting distribution. The picture further more shows the truncated negative parts of the noise distributions in dashed lines, which is the source for the bias in the current method.

5.4 Making the noise unbiased

One might argue that this bias is insignificant as it only affects small values. This is not the case. Keep in mind, that the kernel density plots show multiple clusters with high densities combined with vast "wastelands" of relatively low densities around them. This violates basic accuracy demands for official statistics.

The idea is to not discard the negative noise as show in figure 5.3, but to add it where appropriate. The law of total probability (Papoulis and Unnikrishna Pillai, 2002)

$$P(B) = P(B|A_1)P(A_1) + P(B|A_2)P(A_2), \text{ with } A_1 \dot{\cup} A_2 = \Omega$$

can be used to split a random variable into the sum of a negative and a positive truncated random variable. When applying differential privacy to a spatial grid, the first step is to add the positive noise part to all grid cells, while retaining the negative noise part. In the second step, the negative noise for each randomly shuffled grid cell is selected. All grid cells, to which the negative noise can be added without getting negative are searched for. From those grid cells, the one with the lowest euclidean distance to the current grid cell is selected. If there are more than one with identical distance, a random one is chosen. The negative noise is added to that grid cell. This procedure makes the noise unbiased.

Consider the example from the end of the previous chapter (total number of privately owned cars in 2023). The total number is 267,121 adding only the positive noise changes the total number to 295,386. This is a 10.5% increase and therefore not acceptable. When subtracting the negative noise, the new total number changes to 269568.3. An increase of 0.9% is much more acceptable. Table 5.1 show the

Table 5.1: Distribution of distances between cells and their negative noise

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	No distance
100.2	100.2	100.2	278.7	223.9	5480.4	33935

distances in meters between the grid cell the negative noise belongs to and the cell the noise got added to. The last column is the number of cells where the distance is zero. A maximum distance of 5.5km is much, but this is the most extreme value and in 75% the noise is within a radius of 225m, which is acceptable. Figure 5.4

shows the relative noise (i.e. the true value of a grid cell divided by the value after differential privacy obfuscation) depending on the true value. The further the true value is away from 0, the smaller the relative deviations get.

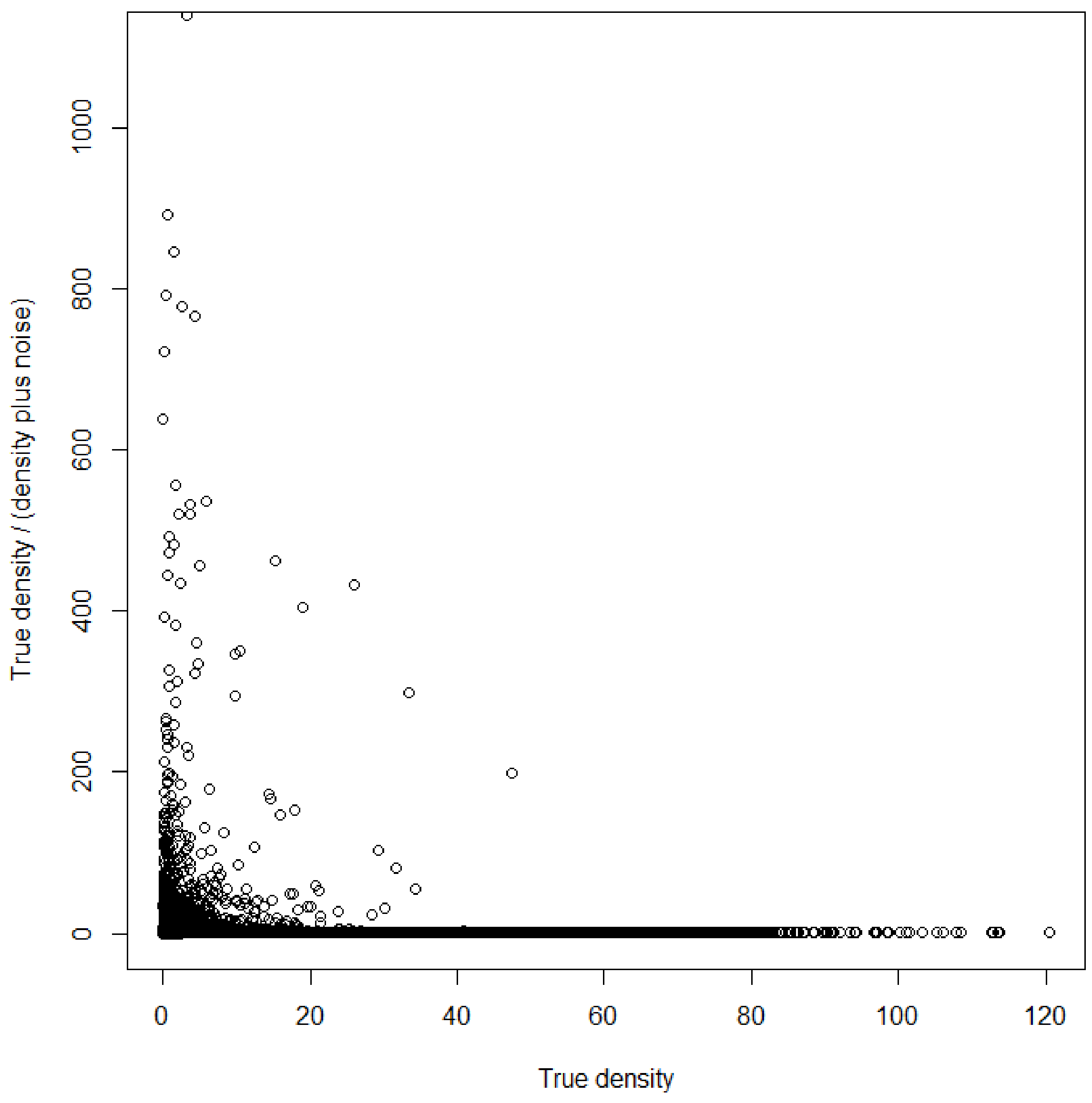


Figure 5.4: Relative noise distribution by true value

6 Analysing data

This chapter analyses the in chapter 4 created density plots for all cars (see appendix A), electric cars (appendix B) and SUV (appendix C) owned by a private person. Electric cars and SUVs have been selected because those subgroups show the greatest changes over the last ten years. This caused some troubles, when selecting proper plot bins. They have been handcrafted to avoid empty plots. For electric cars the grid size had to be expanded from the 100m*100m to a 500m*500m grid to make

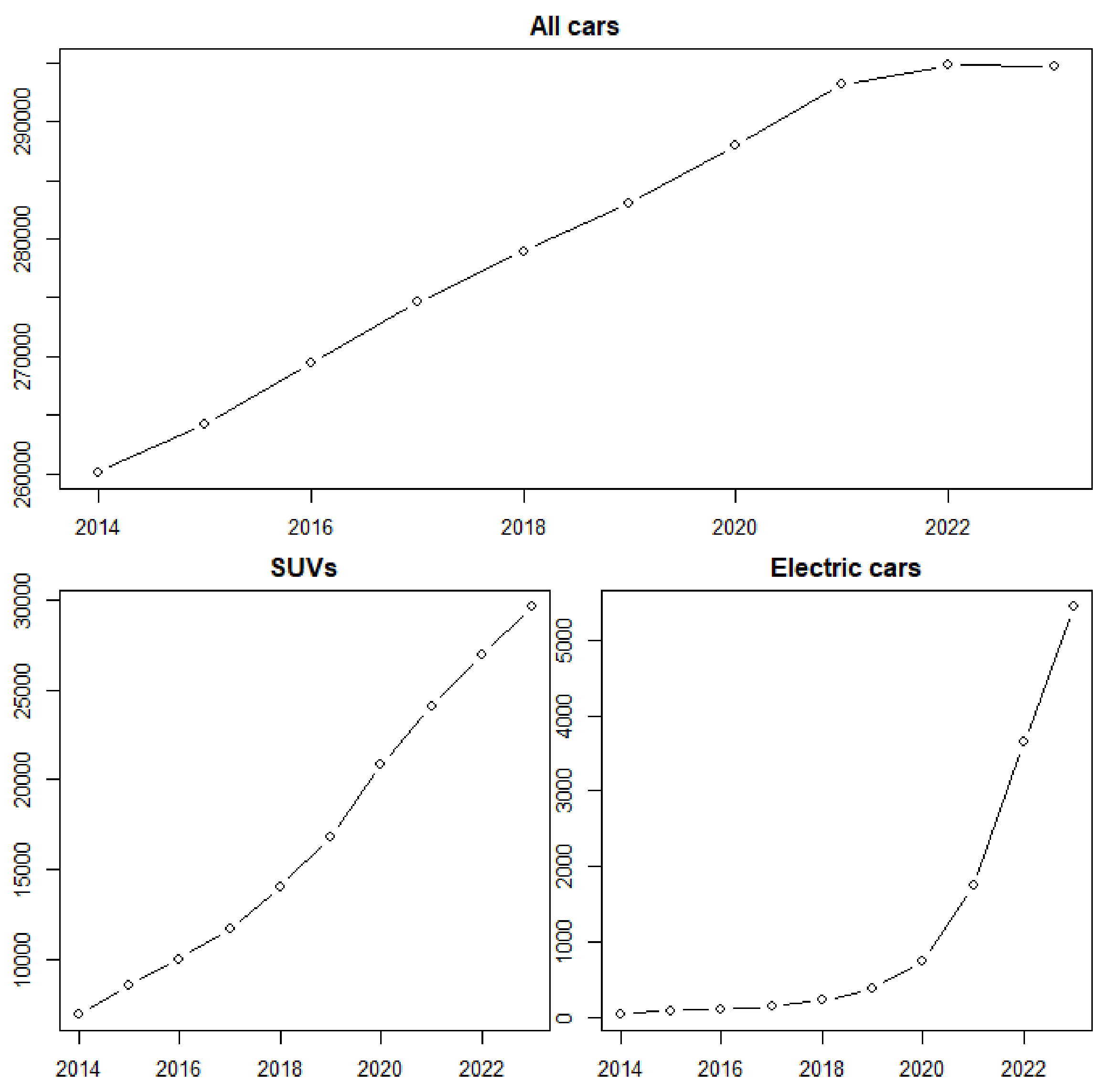


Figure 6.1: Total Number of cars

plots somewhat comparable. The SUV and electric car plots both show an emerging trend. Figure 6.1 shows the total number of cars by year. The spatial distribution illustrate that those trends start in the southern part of Dortmund and slowly travel northbound. This trend can be explained by Dortmund's social structure. The proportion of rich people in the south is much higher than in the northern parts. Likewise, the proportion of poor people is lower in the southern parts compared to the northern ones. Given that both SUVs and electric cars are very expensive, the social structure is reflected on the maps. Apart from this general insights, some more advanced methods to compare the maps should be applied. To give some intuition to the problem take e.g. the data from 2014 and from 2023. Firstly, the total number of vehicles has changed. Secondly, the distribution of vehicles has changed (e.g. vehicles shift from the inner city parts to the outer city areas or from north to south). To analyse changes in distribution the Kantorovich metric is used.

The metric can best be understood by a small simile.

Imagine being in a garden equipped with a shovel and the task at hand is to transform a patch of land shaped like the 2014 distribution into the 2013 distribution. The problem is how to do it with the least amount of effort. Effort can be quantified as the mass (amount) of dirt moved times the distance the dirt has to travel. Additionally, a extra pile of dirt is needed, because there are more cars in 2013 than in 2014. Minimizing the effort in mathematical terms leads to the Kantorovich/Wasserstein distance (for obvious reasons also called earth mover's distance).

Initially assume the total mass for both distributions is equal (true density distribution). Let μ and ν be two discrete two-dimensional density distributions with domains $\mathcal{T}_\mu$ and $\mathcal{T}_\nu$. The Wasserstein distance is defined as

$$\mathcal{W}(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \sum_{(a,b)\in\mathcal{T}_\mu\times\mathcal{T}_\nu} d(a,b)\pi(a,b),$$

where d is a distance measure between points a and b. Π is the set of all two-dimensional joint probability distributions with marginal distributions μ and ν. In other words, the Wasserstein distance minimizes the effort (π is the mass and d the distance) of moving the dirt among all possible transportation plans Π. All $\pi \in \Pi$ can be identified with a matrix $T \in \mathbb{R}^{k_1 k_2}$, where k_1 and k_2 are the number of grid cells (in our case $k_1 = k_2 := k$), giving the probability of moving mass from one cell to another one. All the 1-norm distances d can be calculated in advance and stored in a matrix $C \in \mathbb{R}^{k_1\times k_2}$, where the diagonal elements are zero as there is zero distance between the same grid cells. Let μ_i and ν_i be the marginal probabilities evaluated at grid cell $1 \le i \le k$. Giving this notation, the problem can be reformulated as

$$\begin{array}{cc} \min_{T\in\mathbb{R}^{k_1k_2}} & \sum_{1\le i\le j\le k} T_{ij}C_{ij} \\ s.t. & \forall 1\le i\le j\le k:\ T_{ij}>0 \\ & \sum_{1\le i\le k} T_{ij} = \nu_j\ \forall j\in\{1,\dots,k\} \\ & \sum_{1\le j\le k} T_{ij} = \mu_i\ \forall i\in\{1,\dots,k\}. \end{array}$$

Above's problem is a linear program, which is solvable by e.g. the simplex algorithm (Solomon, 2018). Going back to the original problem, the issue of differing total numbers has not been addressed yet. The idea is to put the additional dirt (extra mass) outside the grid into a virtual grid cell and define a fixed distance between the virtual cell and the real cells. Form there on the computation goes on as usual. An optimized algorithm for two-dimensional data is provided by Bassetti et al., 2020 and implemented in the R-package `SpatialKWD` (Gualandi, 2022).

Figure 6.2, 6.3 and 6.4 show the pairwise Kantorovich metric between the years

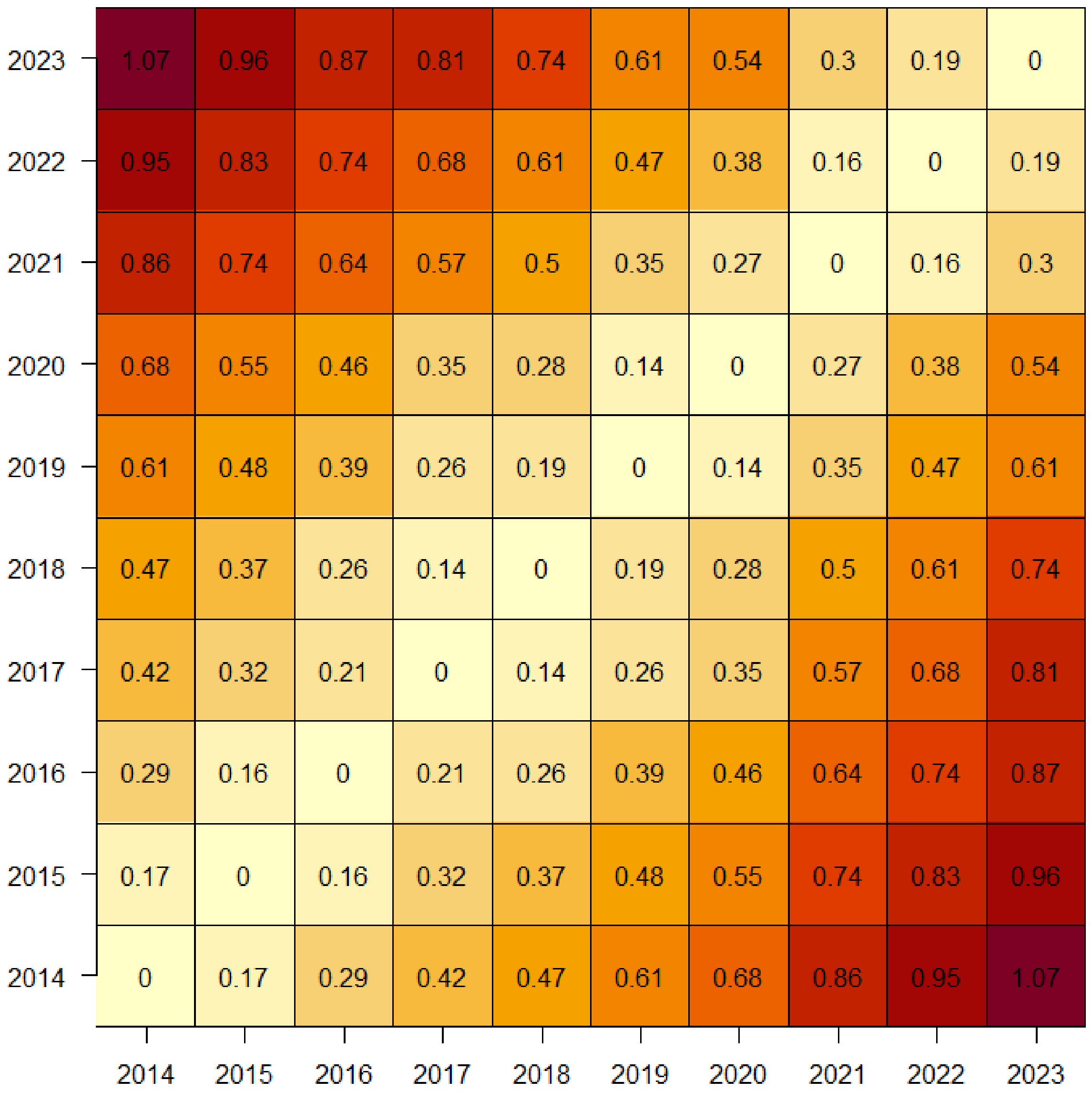


Figure 6.2: Wasserstein distance for privately owned cars

without virtual cells. The most interesting data can be found on the line above (or equally below) the diagonal, which are the year by year distances. Keep in mind, that the general change in the number of vehicles (see figure 6.1) is not reflected in those images, because the values have been normalized. Those images show the changes in the spatial distribution within the city. For all cars (see figure 6.2), the Kantorovich year-by-year metric shows some stable changes which can be regarded as normal fluctuation (between 0.14 and 0.21). This implies the growth in the number of cars that occurred is more or less present in all areas of Dortmund. The SUV plot in figure 6.4 shows a decreasing trend from 1.03 to 0.43 which means, that in the early years around 2014 the structural distribution was still evolving. This evolution is still going on, but will come to a standstill in the near future. A similar, though more extreme development can be seen regarding electric cars (see 6.3). The Kantorovich distances decreased from 3.68 to 0.3 within the last ten

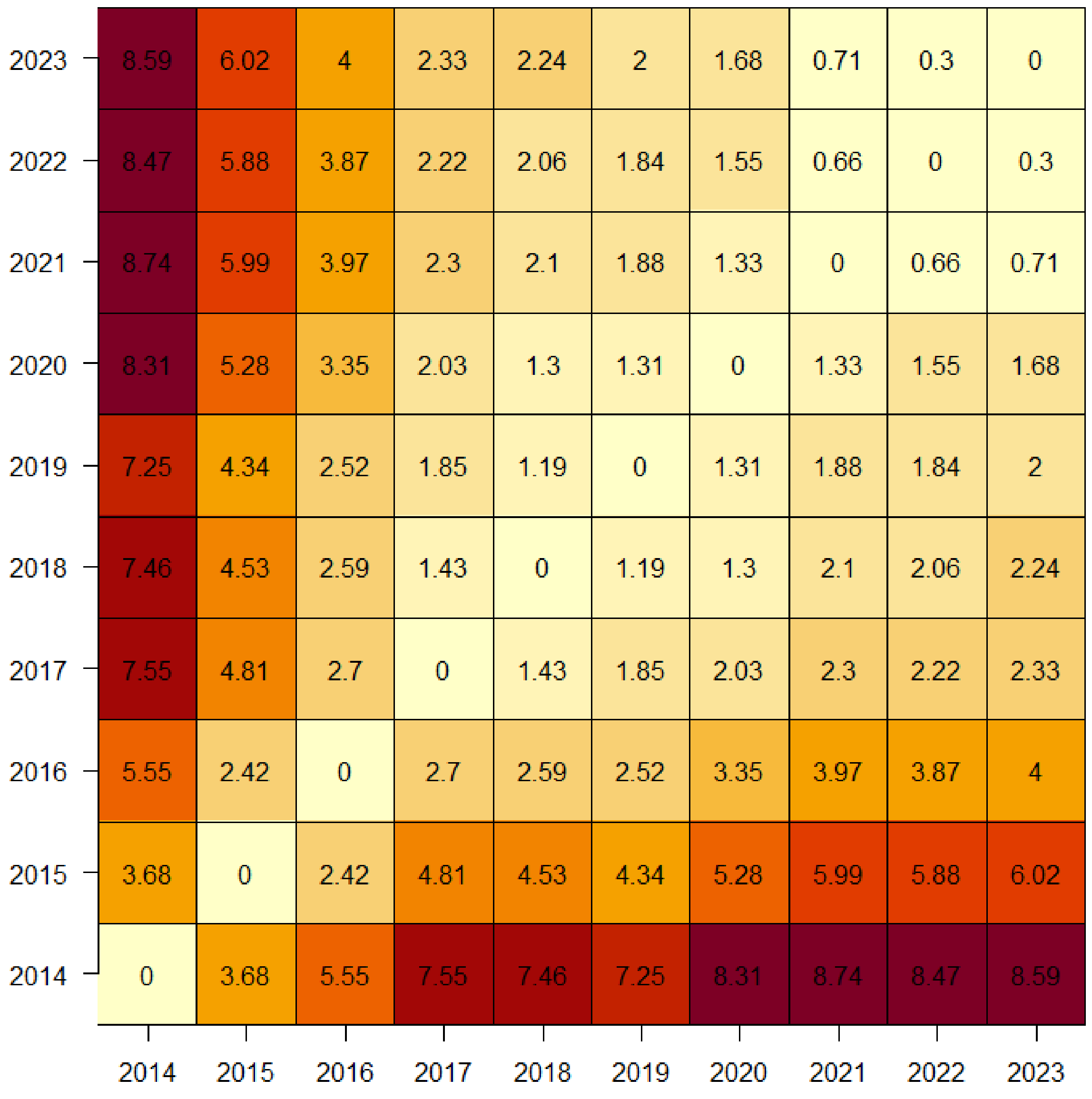


Figure 6.3: Wasserstein distance for privately owned electric cars

years. Though the changes in distribution will likely stall, the total number is still expanding rapidly.

7 Conclusion

To conclude this paper, this section summarizes the main findings. From the point of official statistics it is recommended to use fuzzy matching based on string distances to geocode addresses. Manual encoding yields better results, but is infeasible for huge data sets. Computing missing data can be done with de-noising auto-encoders like the one provided by "MIDAS". This data altering process may are difficult to justify for an authority but simply removing records with missing data is serious data alteration as well and from a statistician's point of view much worse. Further development is needed to allow the auto-encoders predict continuous variables

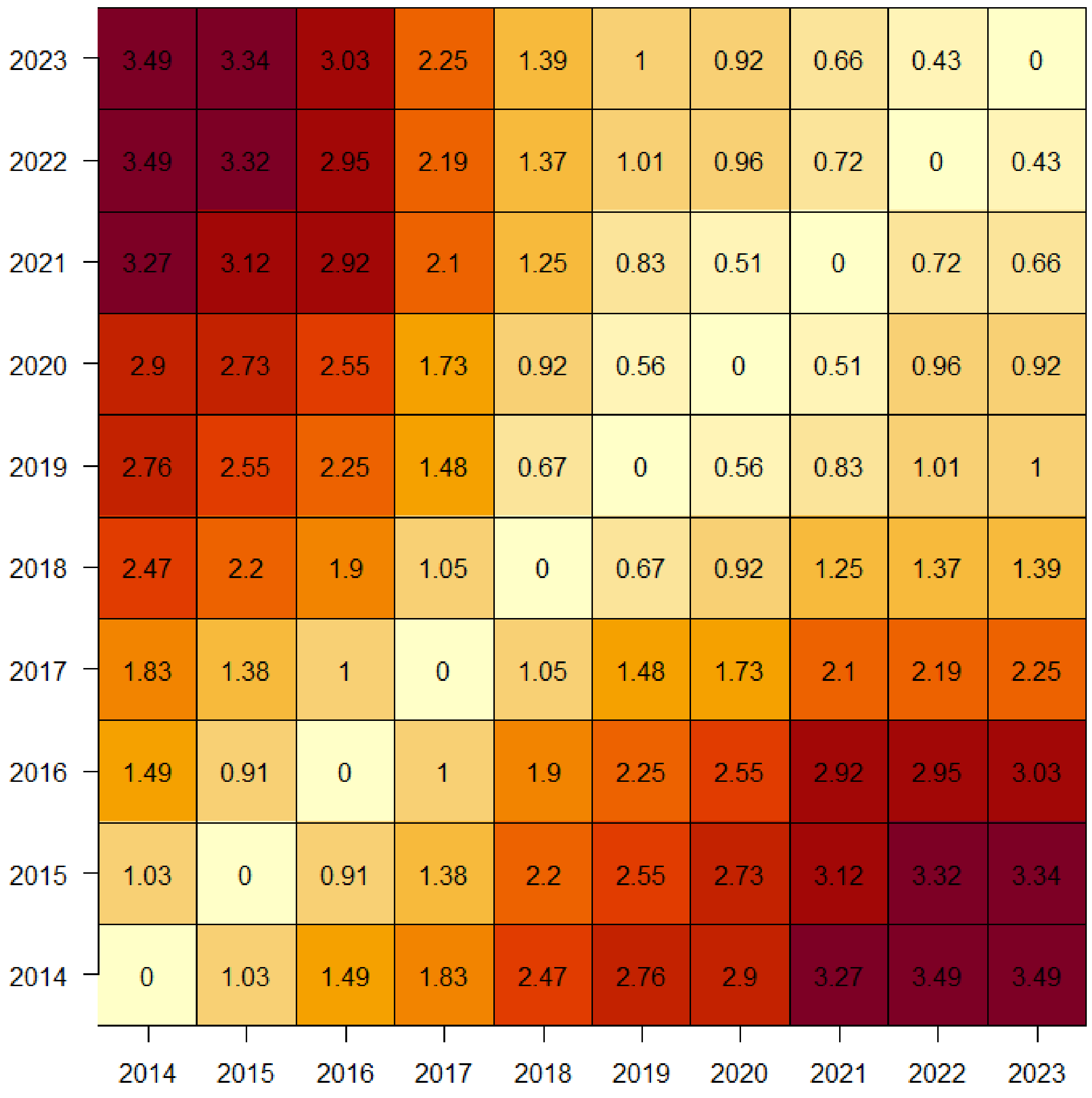


Figure 6.4: Wasserstein distance for privately owned sport utility vehicles

within a restricted interval instead on the whole real line. Supervised classification tasks on structural data should be conducted with xgboost. It often produces reasonable results. Kernel density estimation can be used to visualize spatial data. The methods provided for local bandwidth scaling work and create the desired results. Unfortunately, they do not run fast enough to be used in practice yet. This can be solved by optimizing the used code. To make confidentiality requirements as transparent as possible, black-box methods should be avoided. Differential privacy is the state-of-the-art method to do so. This approach can be expanded to cover multivariate data. Finally, the Kantorovich distance is extremely useful in exploring changes in spatial distributions over time. Most of the described methods can as well be used with population data (which is the main share of all official statistics data available at the municipal level).

References

Abramowitz, M., Stegun, I., and Romer, R. (1988). *Handbook of mathematical functions with formulas, graphs, and mathematical tables.*

Adler, A. (2015). *lamW: Lambert-W Function.* R package version 2.1.2. URL: `https://CRAN.R-project.org/package=lamW`.

Bassetti, F., Gualandi, S., and Veneroni, M. (2020). "On the Computation of Kantorovich-Wasserstein Distances Between Two-Dimensional Histograms by Uncapacitated Minimum Cost Flows". In: *SIAM Journal on Optimization* 30.3, pp. 2441–2469.

Bowyer, A. (1981). "Computing dirichlet tessellations". In: *The computer journal* 24.2, pp. 162–166.

Chen, T. and Guestrin, C. (2016). "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Chen, T., He, T., Benesty, M., Khotilovich, V., and Tang, Y. (2023). *xgboost: Extreme Gradient Boosting.* R package version 1.7.5.1. URL: `https://CRAN.R-project.org/package=xgboost`.

Chollet, F., Kalinowski, T., and Allaire, J. (2022). *Deep learning with R.* Vol. 2. Manning.

Corless, R., Gonnet, G., Hare, D., Jeffrey, D., and Knuth, D. (1996). "On the Lambert W function". In: *Advances in Computational mathematics* 5, pp. 329–359.

Croft, W., Sack, J., and Shi, W. (2022). "Differential privacy via a truncated and normalized laplace mechanism". In: *Journal of Computer Science and Technology* 37.2, pp. 369–388.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). "Calibrating noise to sensitivity in private data analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3.* Springer, pp. 265–284.

Frag den Staat (2012). *Maschinenlesbare Auflistung der Typen-Einordnung von Fahrzeugen.* URL: `https://web.archive.org/web/20230428102412/https://media.frag-den-staat.de/files/foi/6858/312-RDRK9513-M-Maas-Fz2-SV4-als-XLS-F-R_geschwaerzt.pdf`.

Gondara, L. and Wang, K. (2018). "Mida: Multiple imputation using denoising autoencoders". In: *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22.* Springer, pp. 260–272.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning.* MIT press.

Gualandi, S. (2022). *SpatialKWD: Spatial KWD for Large Spatial Maps.* R package version 0.4.1. URL: `https://CRAN.R-project.org/package=SpatialKWD`.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer.

Heidenreich, N., Schindler, A., and Sperlich, S. (2013). "Bandwidth selection for kernel density estimation: a review of fully automatic selectors". In: *AStA Advances in Statistical Analysis* 97, pp. 403–433.

KBA (2023a). *Bestand an Personenkraftwagen nach Segmenten und Modellreihen.* URL: `https://web.archive.org/web/20230428095204/https://www.kba.de/SharedDocs/Downloads/DE/Statistik/Fahrzeuge/FZ12/fz12_2023.xlsx?__blob=publicationFile&v=7`.

— (2023b). *Verzeichnis der Hersteller und Typen.* URL: `https://web.archive.org/web/20230428094749/https://www.kba.de/SharedDocs/Downloads/DE/SV/sv42_pdf.pdf?__blob=publicationFile&v=6`.

Khachiyan, L. (1996). "Rounding of polytopes in the real number model of computation". In: *Mathematics of Operations Research* 21.2, pp. 307–320.

Lall, R. and Robinson, T. (2022). "The MIDAS touch: Accurate and scalable missing-data imputation with deep learning". In: *Political Analysis* 30.2, pp. 179–196.

— (2023). "Efficient Multiple Imputation for Diverse Data in Python and R: MIDASpy and rMIDAS". In: *Journal of Statistical Software.*

Laurent, S. (2022). *delaunay: 2d, 2.5d, and 3d Delaunay Tessellations.* R package version 1.1.1. URL: `https://CRAN.R-project.org/package=delaunay`.

Lee, D. and Schachter, B. (1980). "Two algorithms for constructing a Delaunay triangulation". In: *International Journal of Computer & Information Sciences* 9.3, pp. 219–242.

Lee, J. and Clifton, C. (2011). "How much is enough? Choosing ε for differential privacy". In: *Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14.* Springer, pp. 325–340.

Lyons, A. (2011). *Minimum volume enclosing ellipsoid (MVEE) of a set of points.* URL: `https://web.archive.org/web/20220806081221/https://stat.ethz.ch/pipermail/r-help/2011-March/272997.html`.

Martinez, M. and Stiefelhagen, R. (2019). "Taming the cross entropy loss". In: *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40.* Springer, pp. 628–637.

Moshtagh, N. (2005). "Minimum volume enclosing ellipsoid". In: *Convex optimization* 111.January, pp. 1–9.

Pankova, A. and Laud, P. (2022). "Interpreting Epsilon of Differential Privacy in Terms of Advantage in Guessing or Approximating Sensitive Attributes". In: *2022 IEEE 35th Computer Security Foundations Symposium (CSF).* IEEE, pp. 96–111.

Papoulis, A. and Unnikrishna Pillai, S. (2002). *Probability, random variables and stochastic processes.*

Pebesma, E. (2018). "Simple Features for R: Standardized Support for Spatial Vector Data". In: *The R Journal* 10.1, pp. 439–446. DOI: `10.32614/RJ-2018-009`. URL: `https://doi.org/10.32614/RJ-2018-009`.

Pebesma, E. and Bivand, R. (2023). *Spatial Data Science: With applications in R.* Chapman and Hall/CRC, p. 352. URL: `https://r-spatial.org/book/`.

R Core Team (2023). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

Robinson, T., Lall, R., and Stenlake, A. (2022). *rMIDAS: Multiple Imputation using Denoising Autoencoders.* R package version 0.4.1.

Sheather, S. (2004). "Density estimation". In: *Statistical science*, pp. 588–597.

Sheather, S. and Jones, M. (1991). "A reliable data-based bandwidth selection method for kernel density estimation". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 53.3, pp. 683–690.

Silverman, B. (1986). *Density estimation for statistics and data analysis.* Vol. 26. CRC press.

Sloan, S. and Houlsby, G. (1984). "An implementation of Watson's algorithm for computing 2-dimensional Delaunay triangulations". In: *Advances in Engineering Software (1978)* 6.4, pp. 192–197.

Solomon, J. (2018). "Optimal transport on discrete domains". In: *AMS Short Course on Discrete Differential Geometry.*

Turner, R. (2021). *deldir: Delaunay Triangulation and Dirichlet (Voronoi) Tessellation.* R package version 1.0-6. URL: `https://CRAN.R-project.org/package=deldir`.

van der Loo, M. (2014). "The stringdist package for approximate string matching". In: *The R Journal* 6 (1), pp. 111–122. URL: `https://CRAN.R-project.org/package=stringdist`.

Venables, B. (2023). *MASSExtra: Some 'MASS' Enhancements.* R package version 1.2.2. URL: `https://CRAN.R-project.org/package=MASSExtra`.

Wand, M. and Jones, M. (1994). *Kernel smoothing.* CRC press.

Wasserman, L. (2006). *All of nonparametric statistics.* Springer Science & Business Media.

Watson, D. (1981). "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes". In: *The computer journal* 24.2, pp. 167–172.

A Density plots privately owned cars

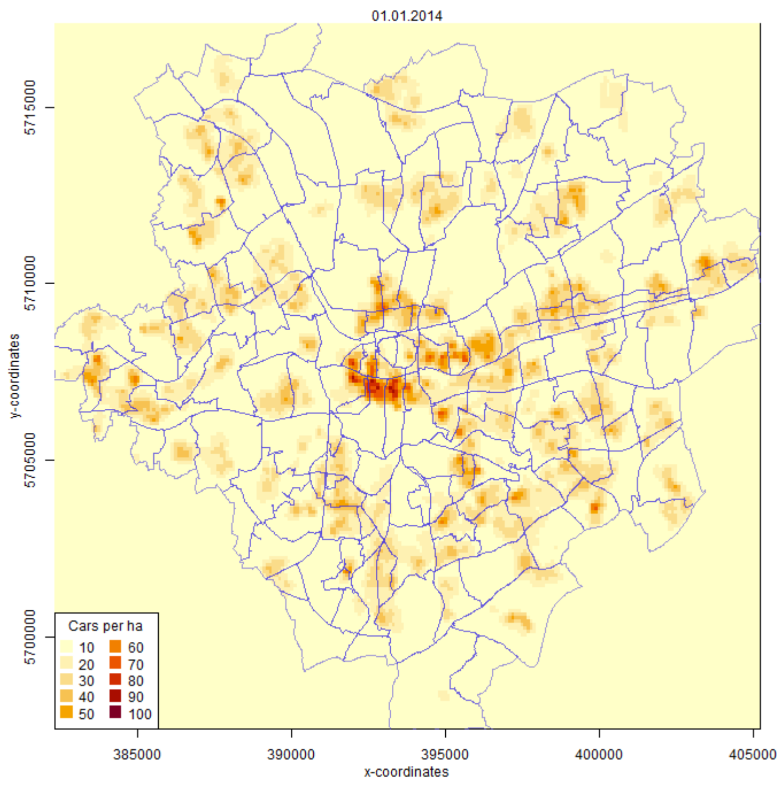


Figure A.1: Privately owned cars in 2014

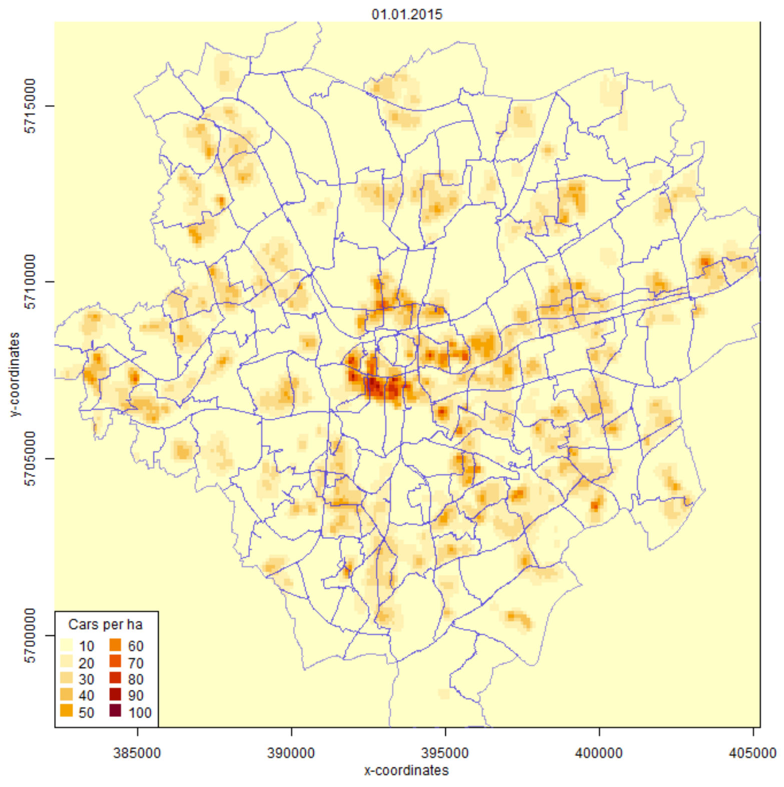


Figure A.2: Privately owned cars in 2015

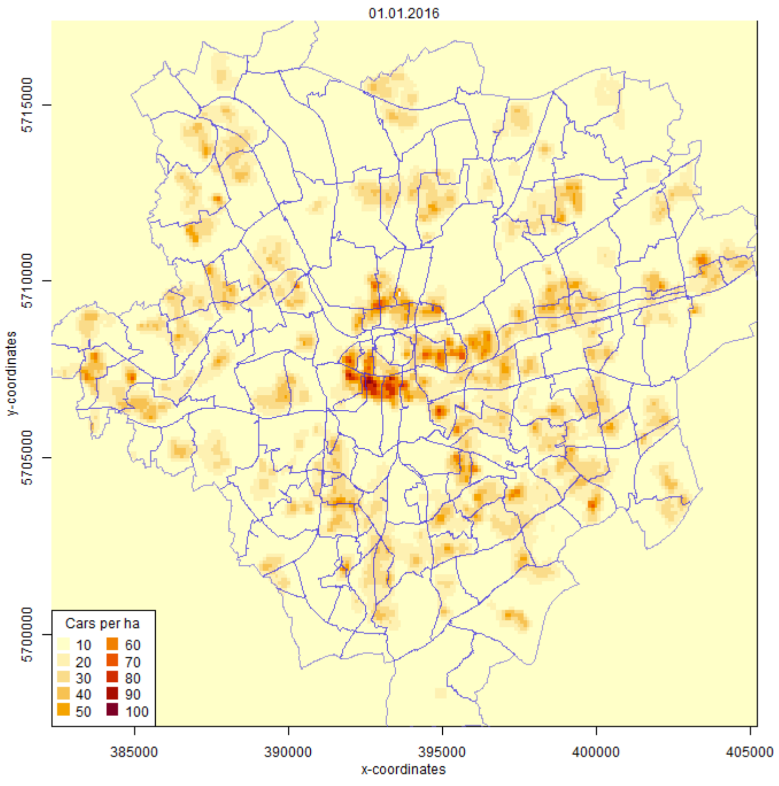


Figure A.3: Privately owned cars in 2016

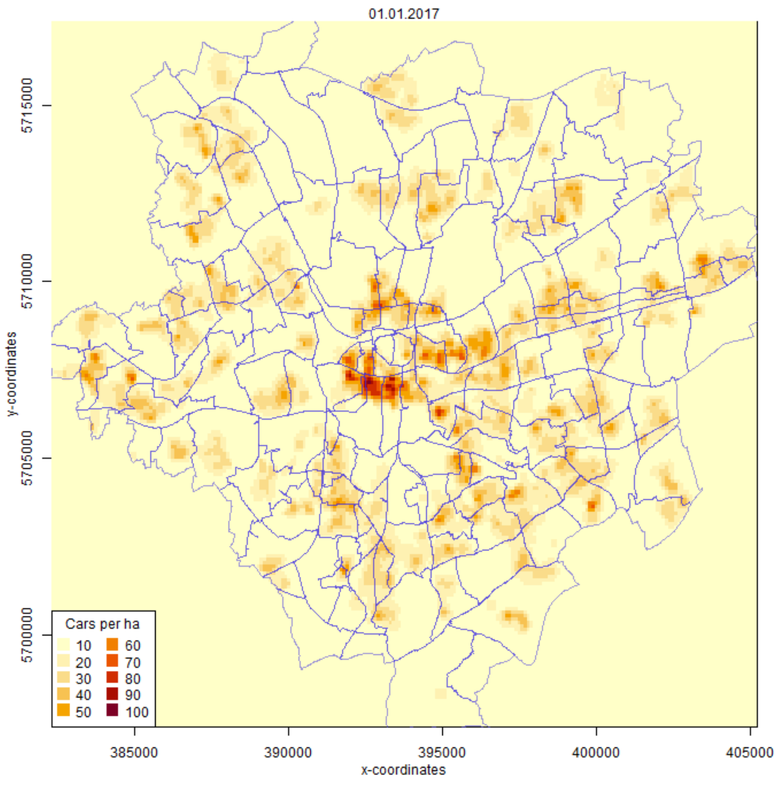


Figure A.4: Privately owned cars in 2017

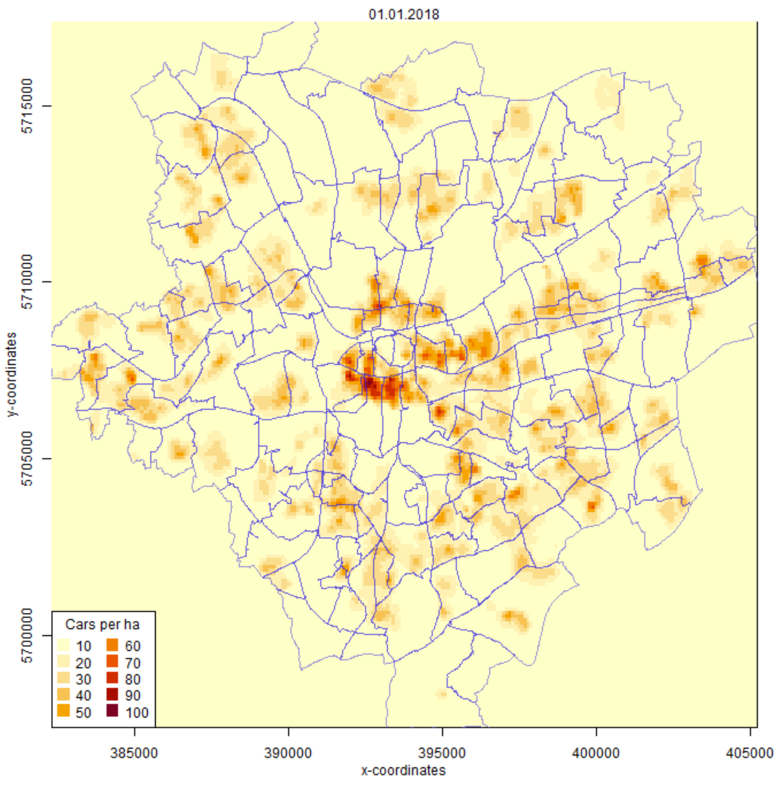


Figure A.5: Privately owned cars in 2018

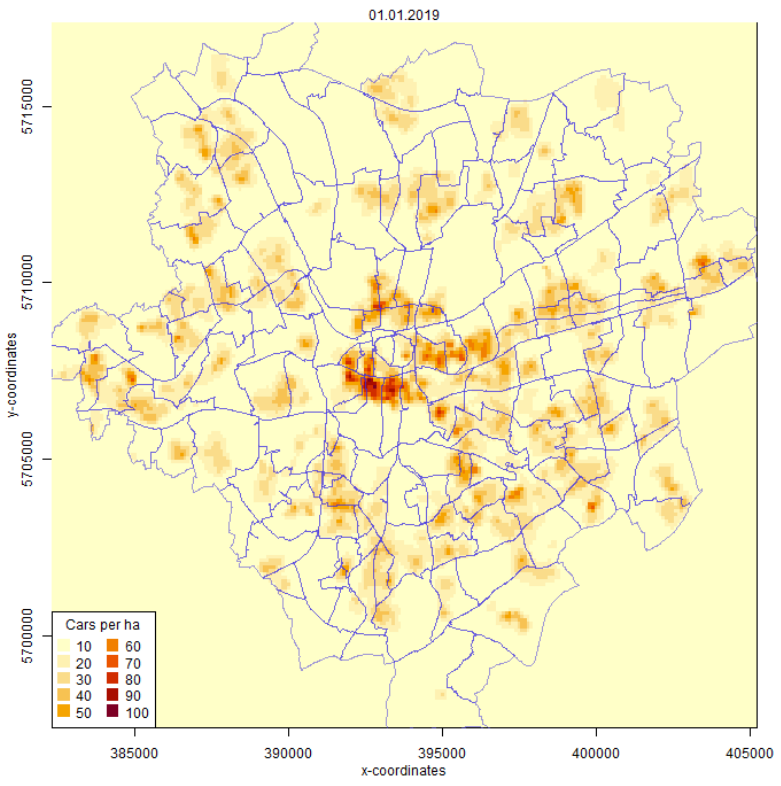


Figure A.6: Privately owned cars in 2019

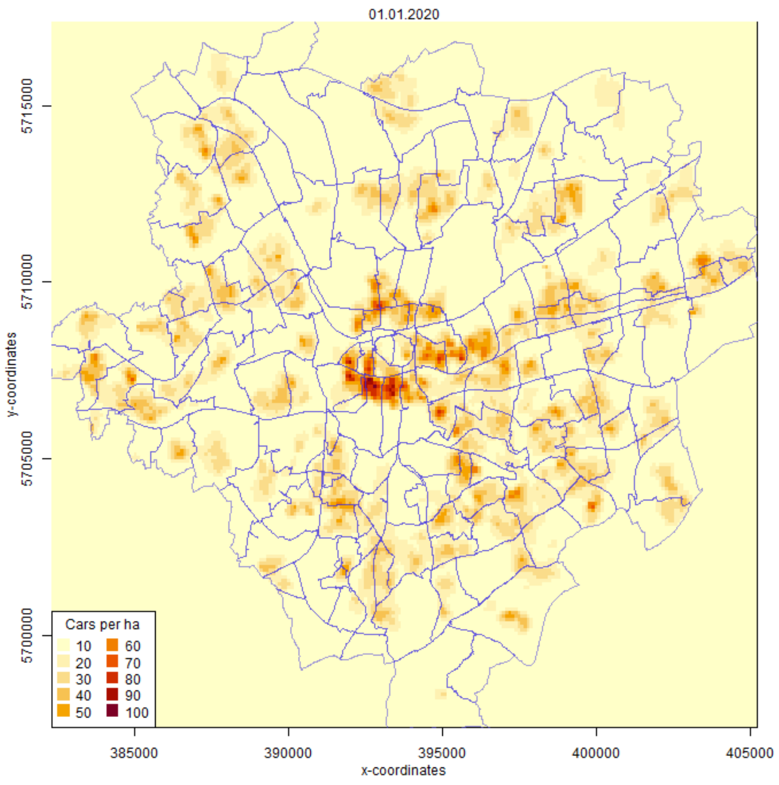


Figure A.7: Privately owned cars in 2020

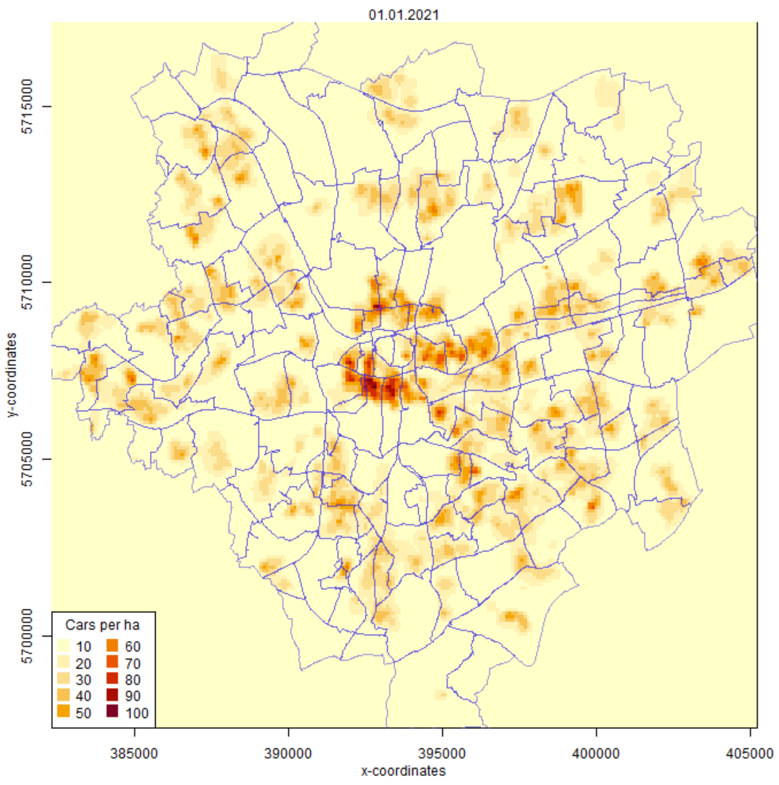


Figure A.8: Privately owned cars in 2021

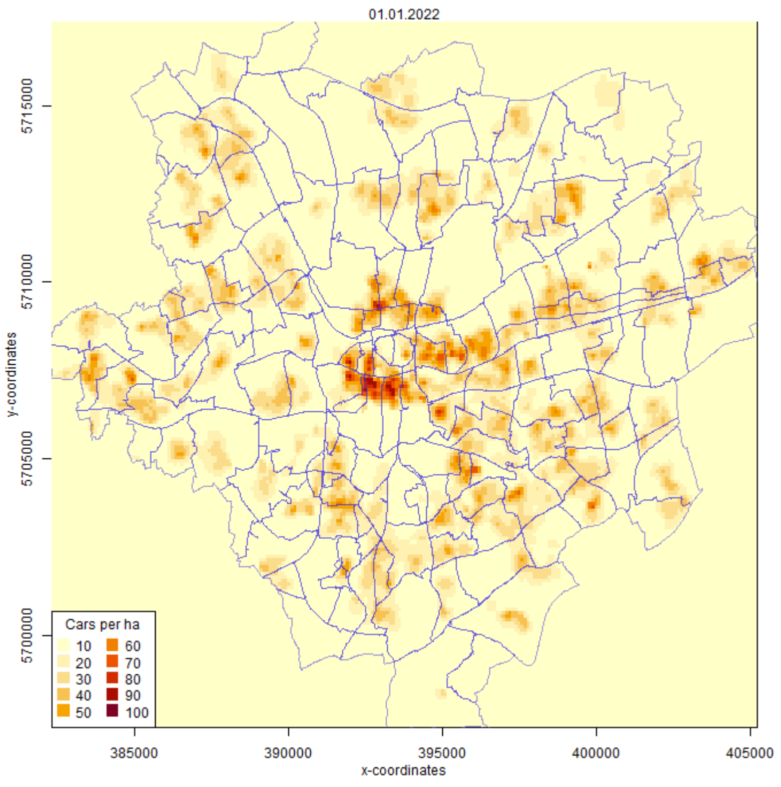


Figure A.9: Privately owned cars in 2022

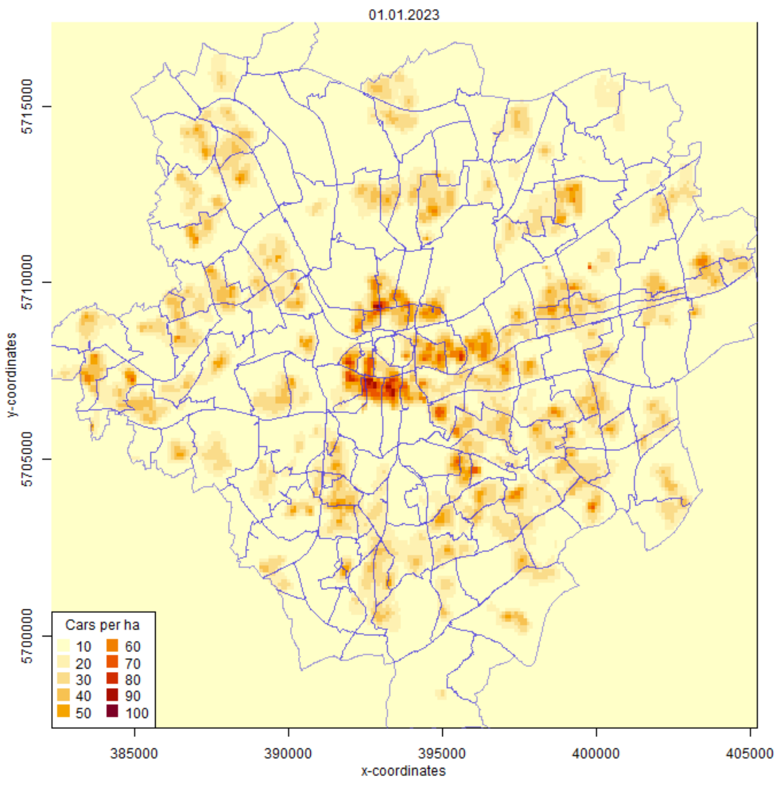


Figure A.10: Privately owned cars in 2023

B Density plots electric cars

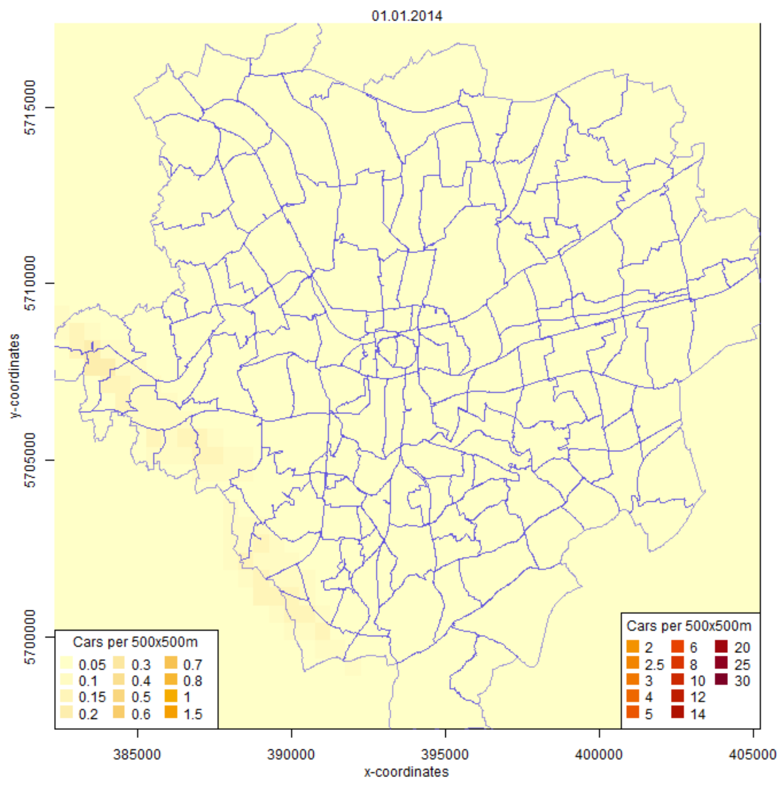


Figure B.1: Privately owned electric cars in 2014

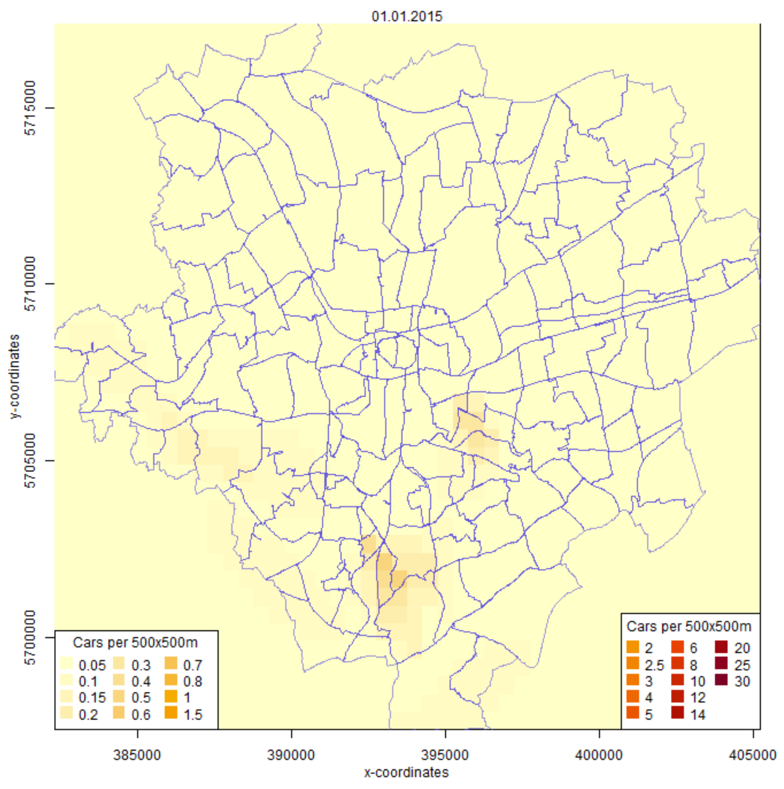


Figure B.2: Privately owned electric cars in 2015

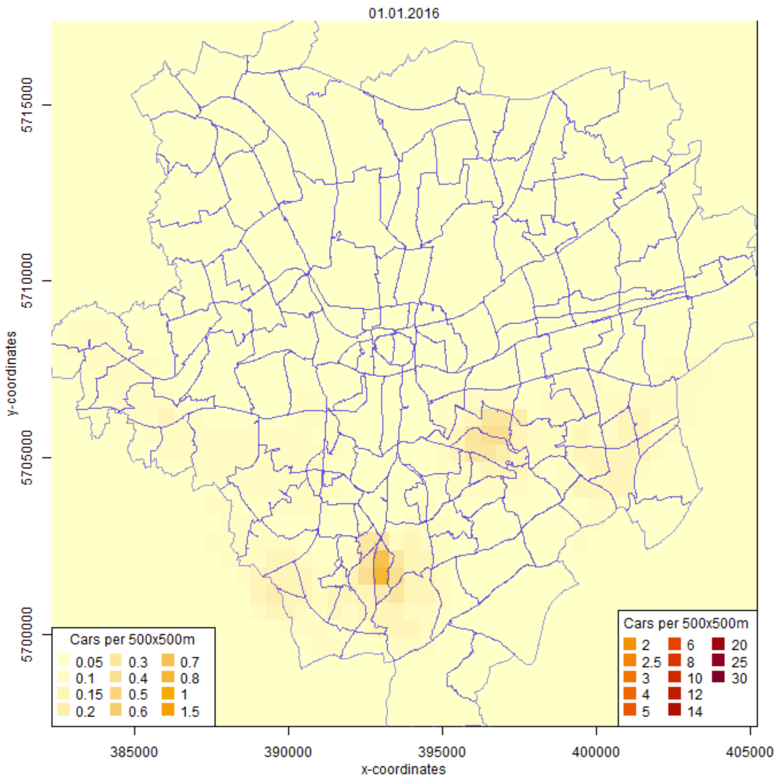


Figure B.3: Privately owned electric cars in 2016

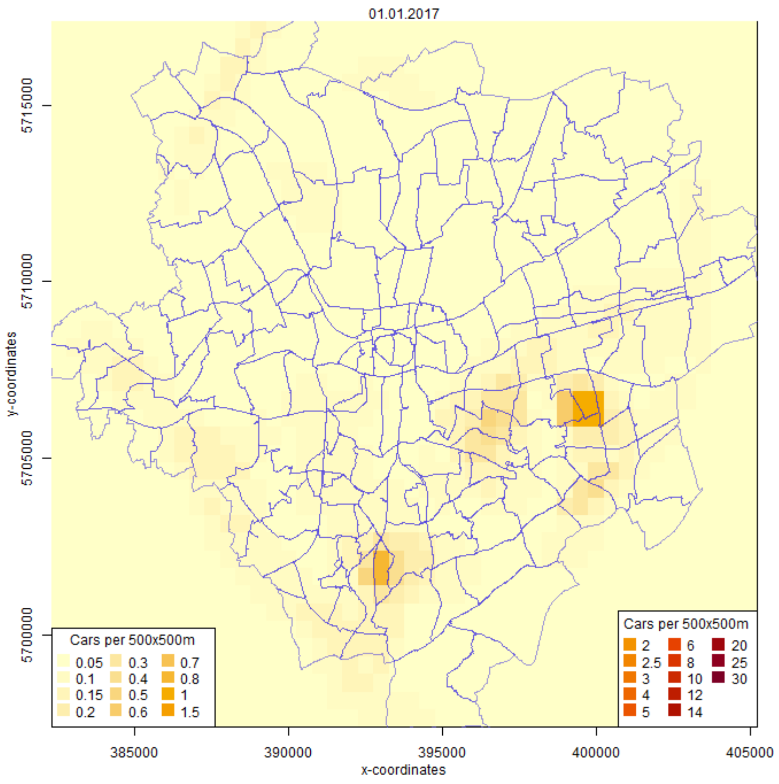


Figure B.4: Privately owned electric cars in 2017

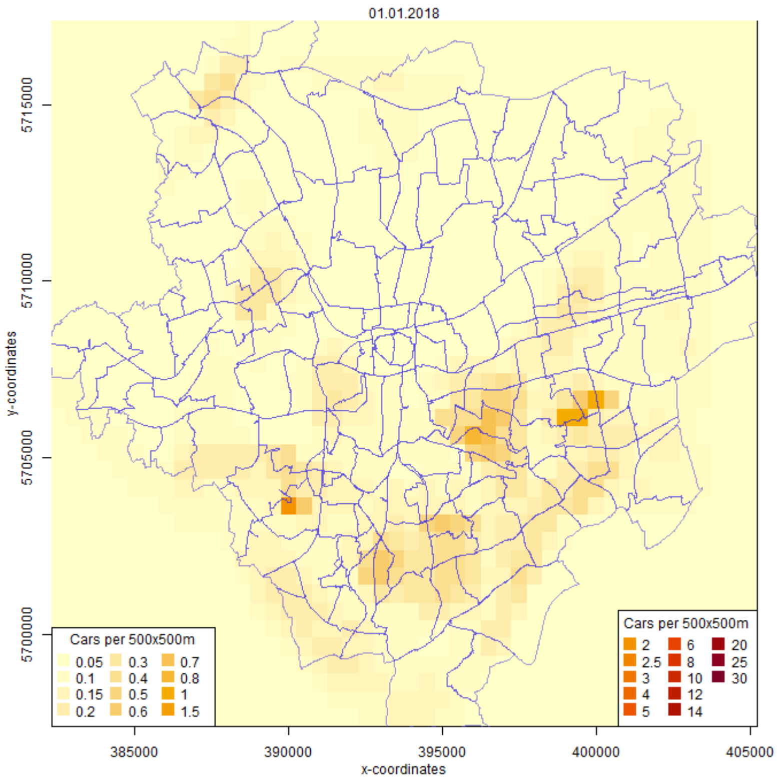


Figure B.5: Privately owned electric cars in 2018

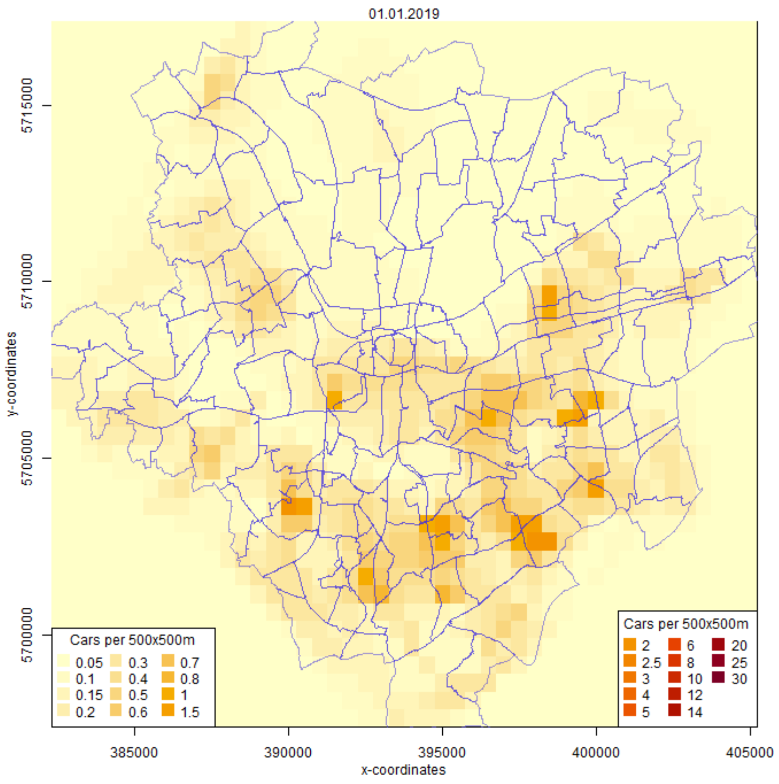


Figure B.6: Privately owned electric cars in 2019

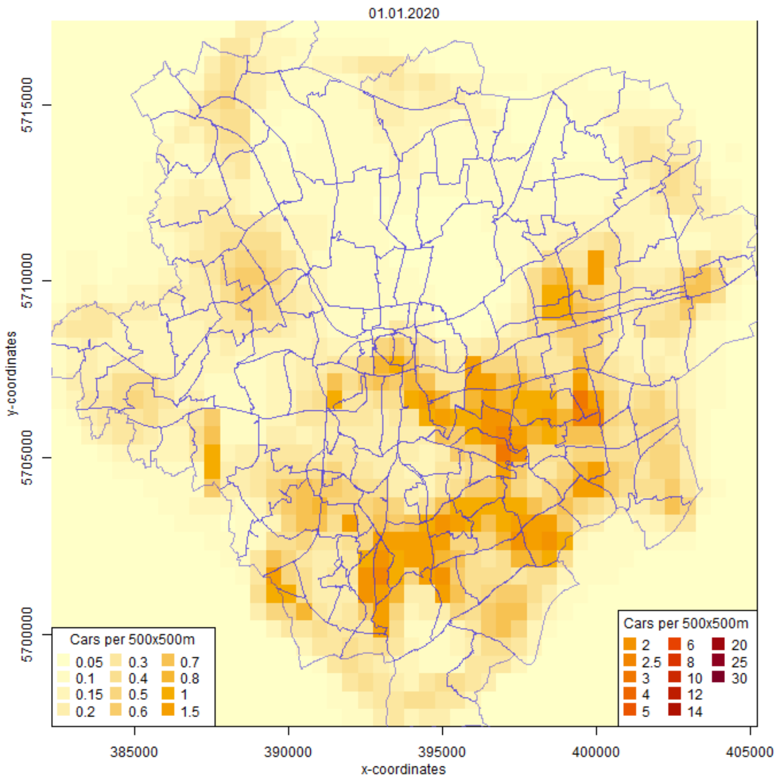


Figure B.7: Privately owned electric cars in 2020

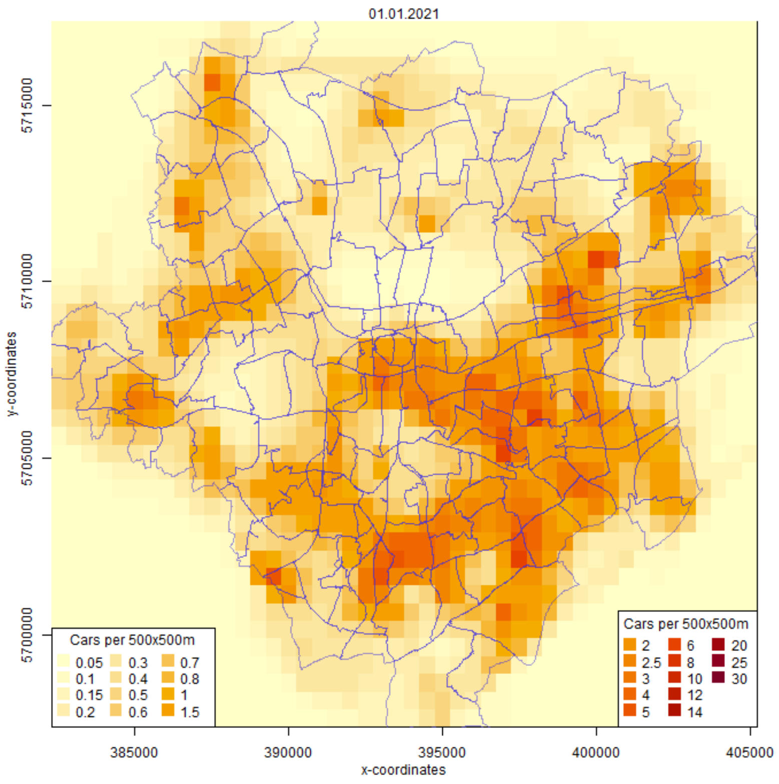


Figure B.8: Privately owned electric cars in 2021

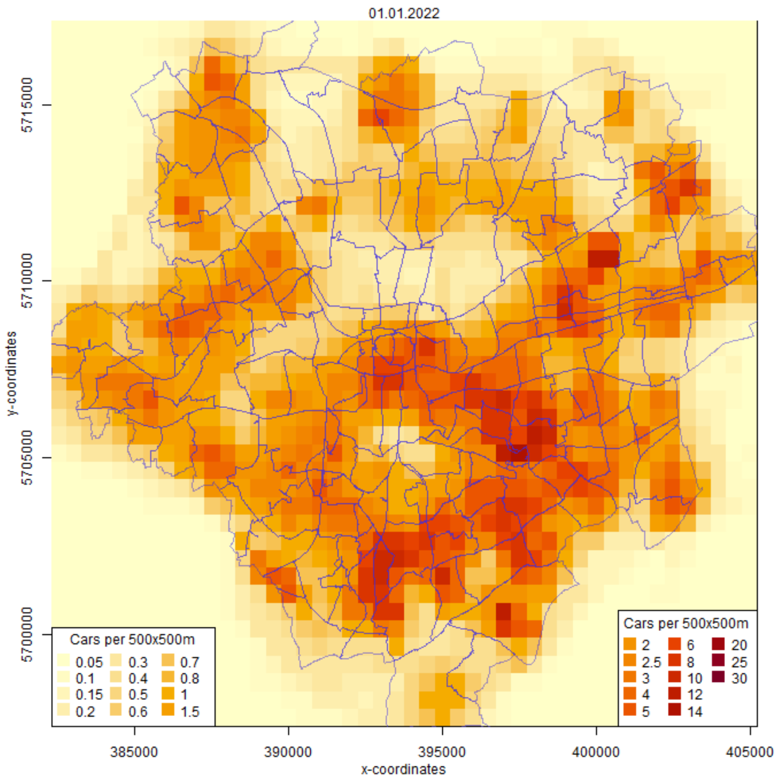


Figure B.9: Privately owned electric cars in 2022

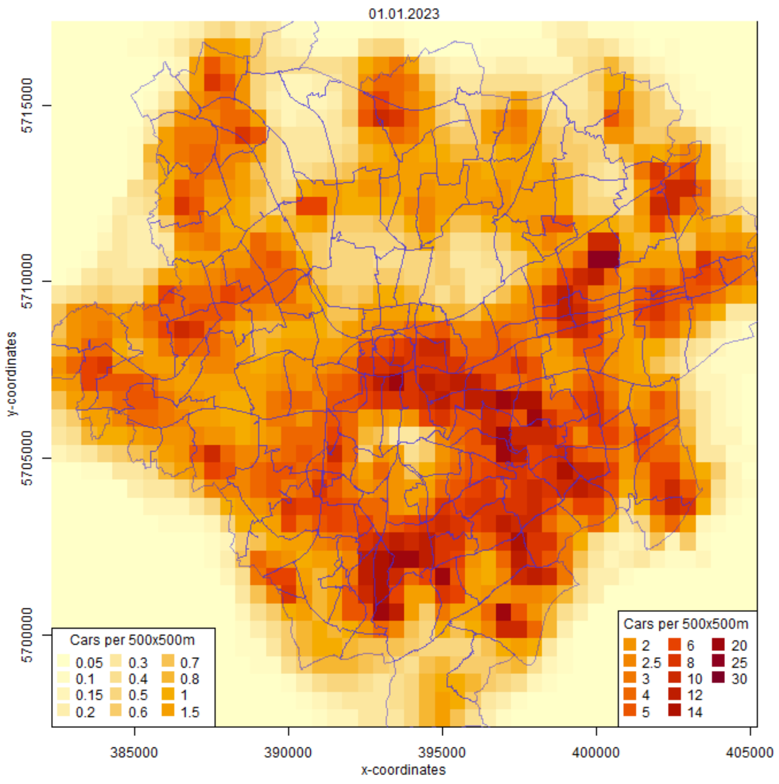


Figure B.10: Privately owned electric cars in 2023

C Density plots sport utility vehicles

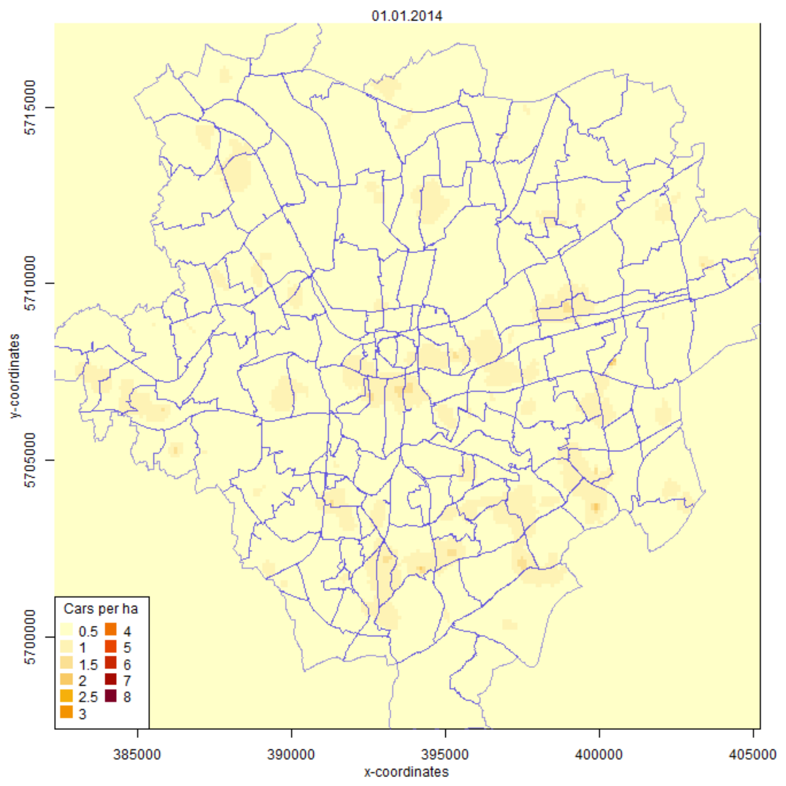


Figure C.1: Privately owned sport utillity vehicles in 2014

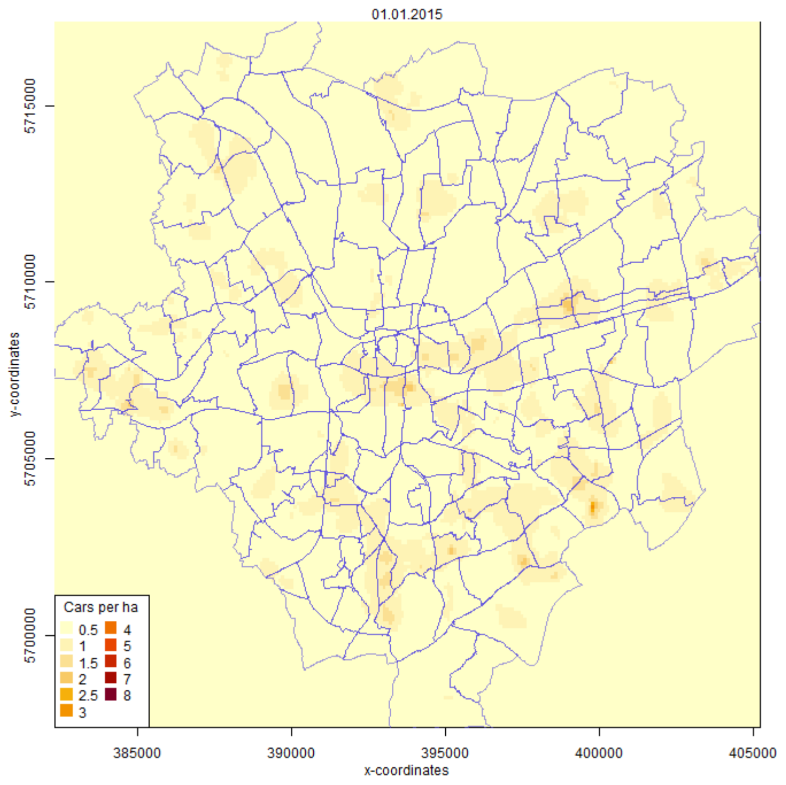


Figure C.2: Privately owned sport utillity vehicles in 2015

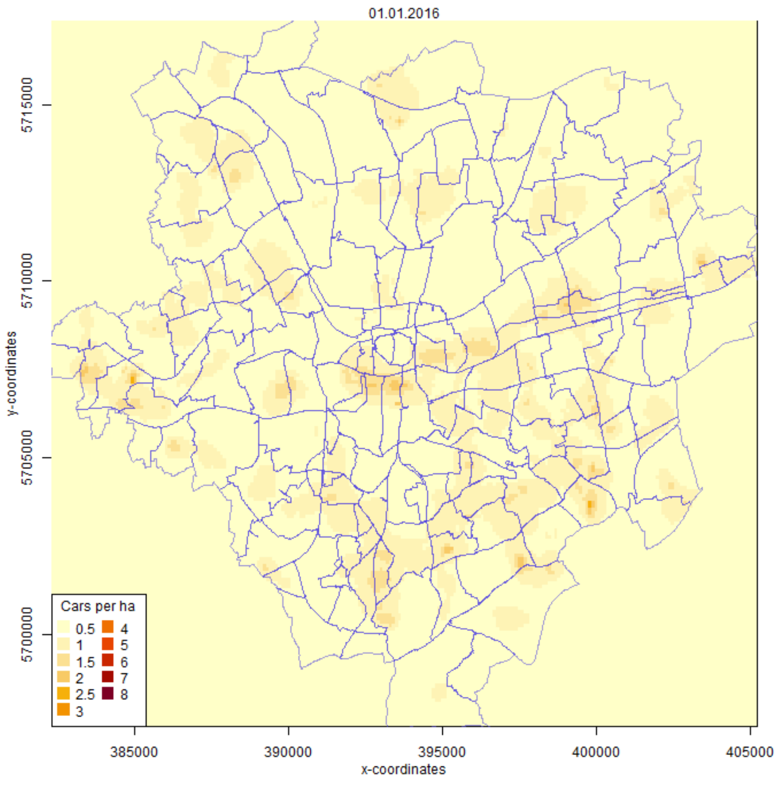


Figure C.3: Privately owned sport utillity vehicles in 2016

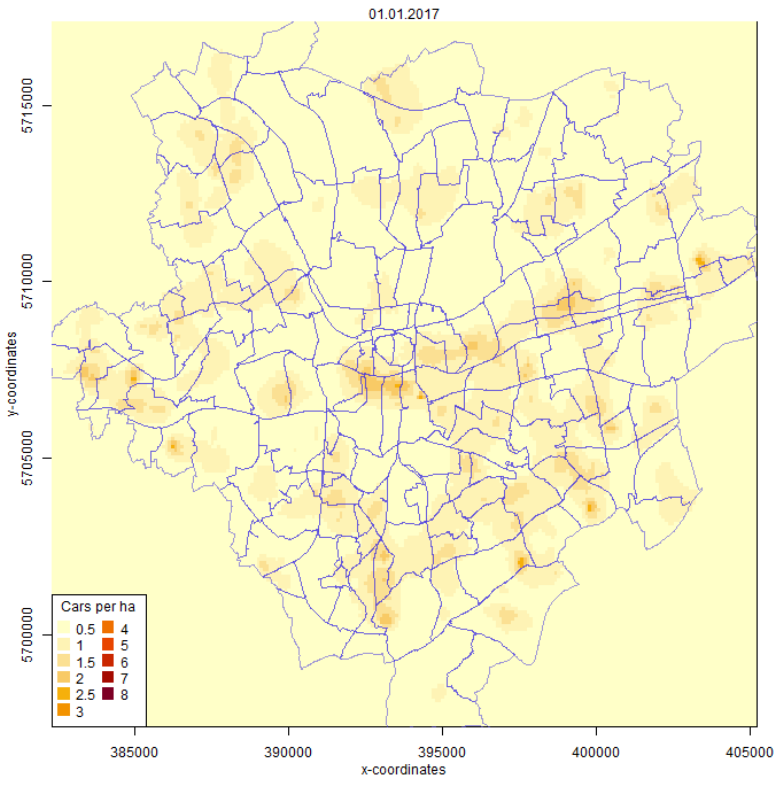


Figure C.4: Privately owned sport utillity vehicles in 2017

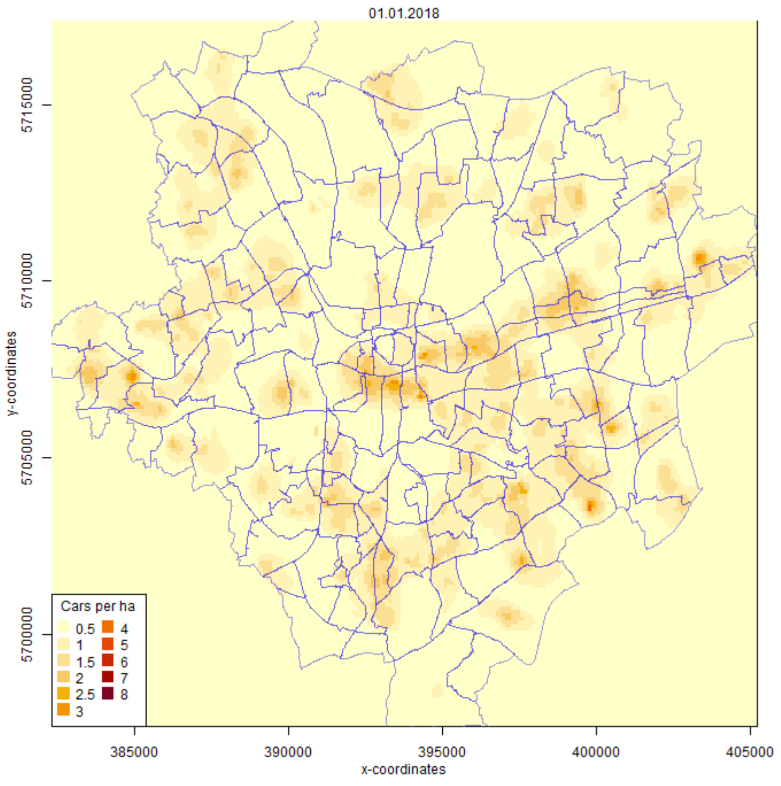


Figure C.5: Privately owned sport utillity vehicles in 2018

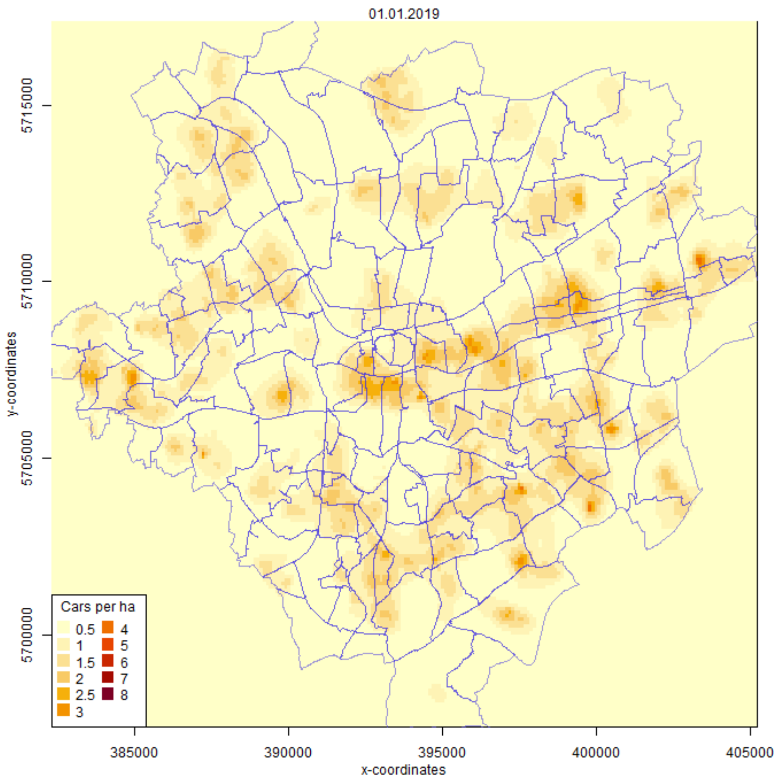


Figure C.6: Privately owned sport utillity vehicles in 2019

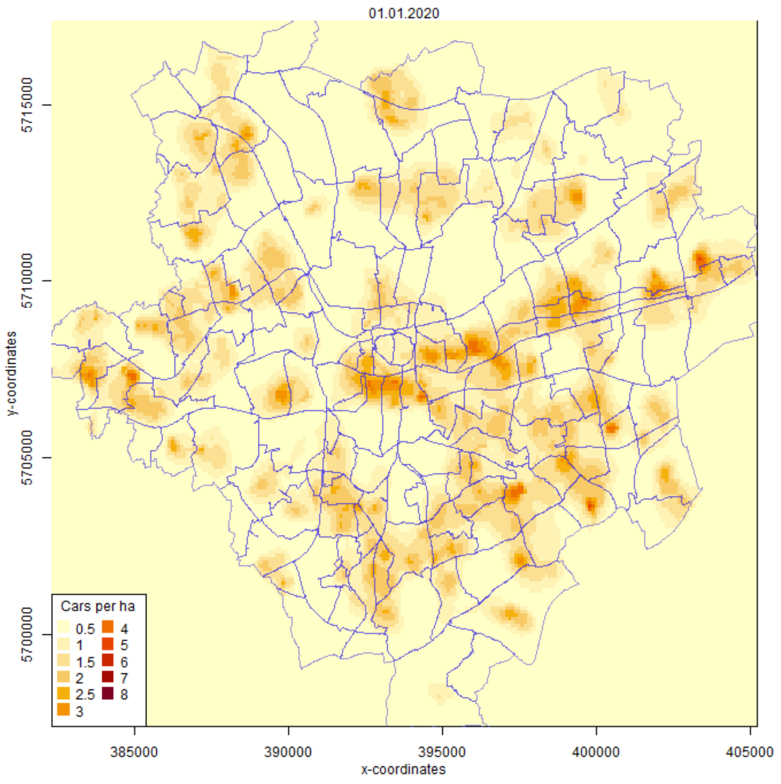


Figure C.7: Privately owned sport utillity vehicles in 2020

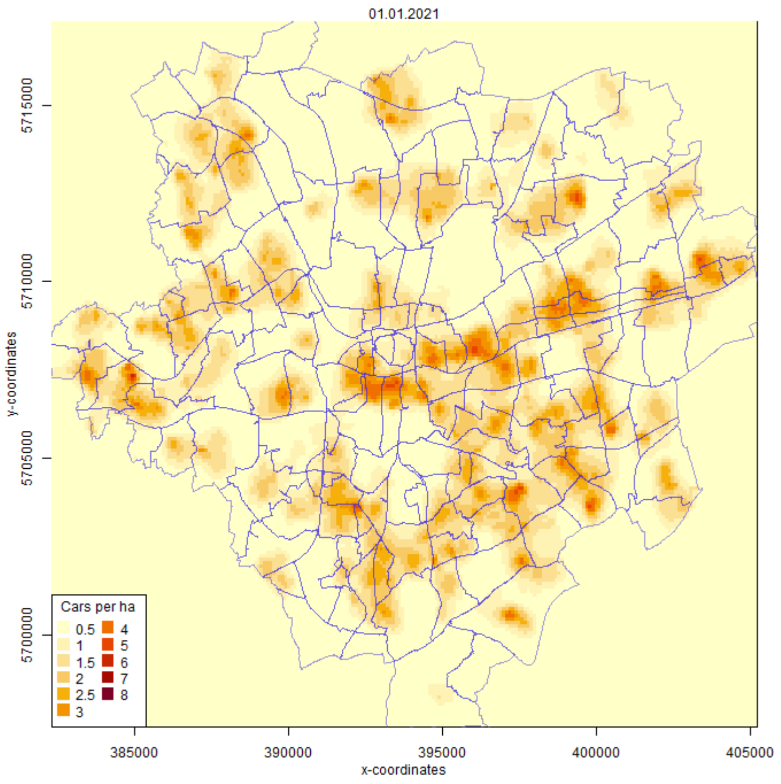


Figure C.8: Privately owned sport utillity vehicles in 2021

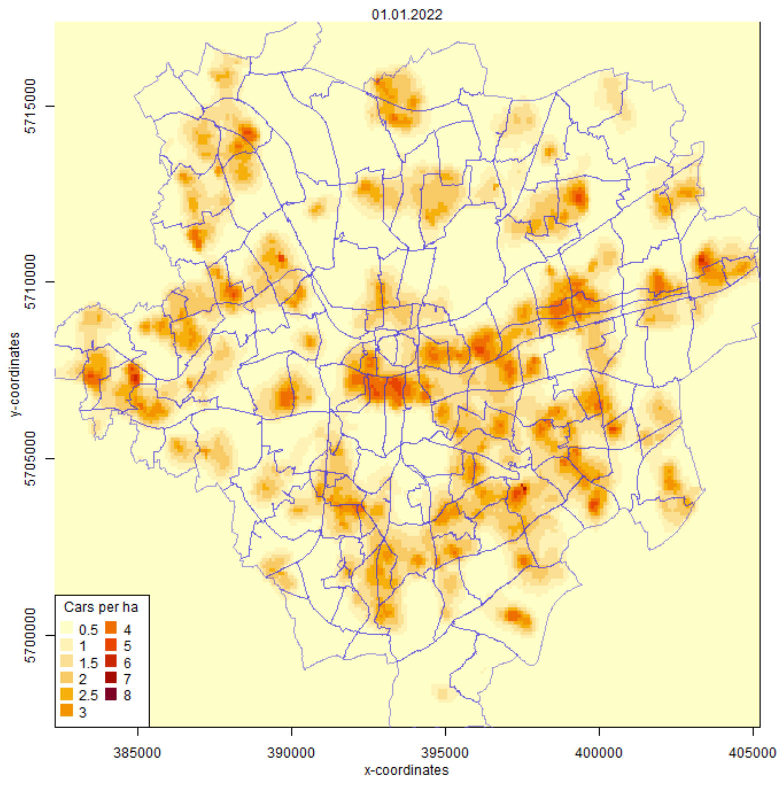


Figure C.9: Privately owned sport utillity vehicles in 2022

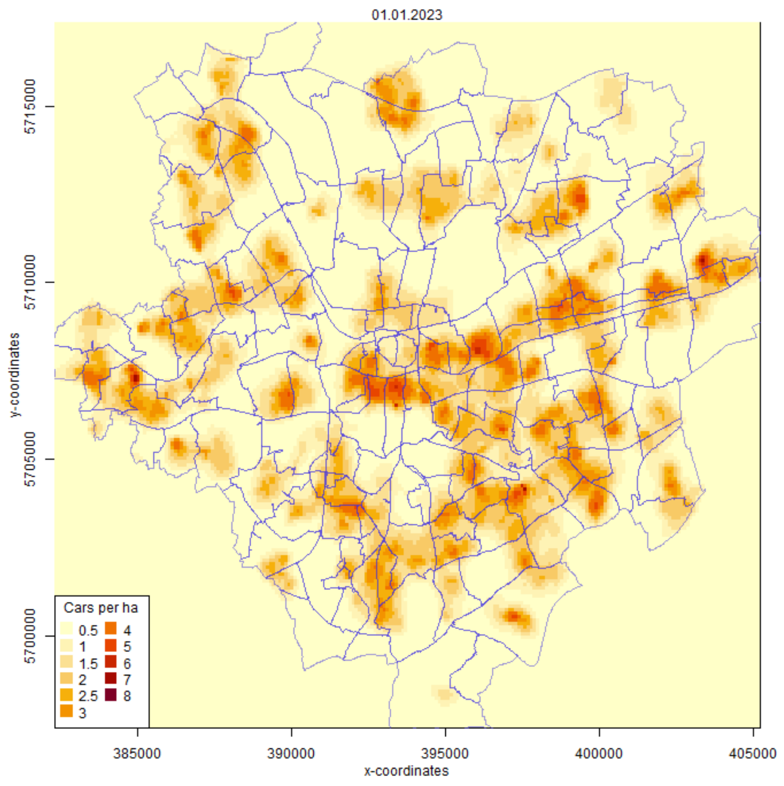


Figure C.10: Privately owned sport utillity vehicles in 2023