CHAIR OF
STATISTICS
DER FREIEN UNIVERSITÄT BERLIN

Bachelor Thesis

Exploratory Spatial Analysis with JMP: An example based on urban data from Berlin

Marcelo Rainho Avila

Supervisor:	Prof. Dr. Ulrich Rendtel
Semester:	Winter Term 2017/2018
Matriculation-Nr.:	4679876
Address:	Seestraße 100, 13353, Berlin
Email:	m.avila@fu-berlin.de
Subject:	Bachelor Economics
Submission:	24.11.2017

Abstract

Exploratory data analysis is a crucial step when searching for patterns in a dataset. This thesis aims to explore key capabilities of the statistical software JMP relative to spatial data analysis, where two common scenarios are examined. Firstly, a Kernel Density Estimation is conduct using exact geocoordinates as data points and implemented over an interactive map via Web Mapping Service functionality. Secondly, the implementation of a choropleth map, including reading a ESRI shapefile and creating a custom map shape within JMP, is explored to illustrate a scenario where exact geocoded data is not available and a comparison between these two methods is also discussed. Datasets relating to the city of Berlin are used in both scenarios. The boundaries used are based on the *Lebensweltlich orientierten Räume* and the data points are retrieved from *Stolpersteine* art project, which aims to remember those persecuted by the National Socialist regime in Germany. The results of this analysis shows that JMP's interactive capabilities of exploring and visualising spatial data and its intuitive graphical user interface reveal to be useful tools in both scenarios. Nonetheless, some aspects could be improved if the official documentation would include more detailed information on the calculation procedures and algorithms used. Further, if a higher degree of flexibility were available within the Web Mapping Service context, the user would gain from the access to this functionality for a broader set of use cases, even considering the trade-off between user friendliness and complexity.

Contents

List of Figures

1. Introduction

Exploratory data analysis is a important starting point when searching for patterns in a dataset. At early stages, iterative and responsive visualisation procedures facilitate quicker examination on previously unnoticed features within the data, which can become the basis for further inquiry. When dealing with spatial data, the capability of combining the estimation procedures with mapping functionality allows for quick verification of results plausibility and fine adjustments when needed. Interactiveness and intuitive visualisation are key focus areas in the offering from statistical software JMP, developed by SAS Institute. In this respect, the main aims of this thesis are twofold: 1) Examine JMP's capabilities for exploratory spatial data analysis while documenting in a didactic manner the steps necessary in each implementation for reproducibility purposes. And 2) compare two methods of density estimation for two common use cases when dealing with spatial data. That is, a kernel density estimation for when exact geo-coded data is available compared to a choropleth mapping implementation for aggregated data, which illustrates a scenario where exact geocoordinates are inaccessible or not available.

To achieve these objectives, in the Second Chapter I firstly present the data used on the subsequent procedures and discuss JMP's mapping capabilities in a more general sense. The data is retrieved from the *Stolpersteine* art project from Gunter Demnig, which aims to remember all those persecuted by the National Socialist regime between 1933 and 1945 by placing a brass block in front of their last chosen place of residence (Demnig, n.d.). Regarding the mapping features, I focus on JMP's Web Mapping Service (WMS) integration. This functionality is used to plot interactive background maps in the subsequent estimations procedures. The Third Chapter is devoted to the non-parametric density estimation in the context of spatial analysis when exact geocoordinates are available, with focus on the Kernel Density Estimation method. An alternative procedure, which generates different visual results and uses a Delaunay triangulation algorithm is also briefly discussed. In Chapter 4, again using the Stolpersteine data, but now aggregated over the *Lebensweltlich orientierte Räume*, I present the shapefile creation and implementation within JMP, that serves as a basis for a choropleth mapping procedure. This process aims to illustrate one of the available tools for density estimation when only aggregated data is accessible. In Chapter 5, I briefly compare the results achieved with

both methods and present the results of a kernel density estimation also for the case when exact geocoordinates are not available, using weighted centroids of cell boundaries as data points.

In conclusion, JMP reveals to be a useful tool for exploratory spatial data analysis with its intuitive and interactive interface. For more profound research and statistical testing, JMP's documentation, with its strong focus on user-friendliness, lacks an adequate level of transparency and it is often unclear what exact computation and algorithms are in place. Moreover, when comparing the estimation procedures with exact geo-coded information against aggregated data, the preliminary results indicate that, given relatively high granular boundaries, a kernel density estimation can achieve acceptable results also when precise geocoordinates are not publicly accessible.

Note on formatting syntax

In subsequent chapters, the adopted syntax where a short series of actions are placed within square brackets indicates computer operations within JMP, with the following formatting: [*Starting Point*: Action 1 > Action 2 > ... > Action N].

2. Stolpersteine Data and General Mapping Functionality

In this chapter, I demonstrate the implementation of a background map within JMP with the objective to first present this functionality in a more general scenario before applying estimations procedures in subsequent chapters. In Section 2.1, I briefly discuss about Gunter Demnig's art project called *Stolpersteine*, from which the data will be used for this implementation and in the following chapters. Further, in Section 2.2, I discuss the geocoding method for extracting geocoordinates from street addresses. Finally in Section 2.3, I present the interactive mapping capability that will serve as a basis for the estimation visualisations discussed in the subsequent chapters.

2.1. The Stolpersteine Project

The ongoing art project initiated by Gunter Demnig in the beginning of 1990s in Cologne aims to remember those who were persecuted by the National Socialist regime between 1933 and 1945 by placing a brass plated block in front of the victims last freely chosen residence (Stolpersteine Art Project, n.d.). The list of commemorated people include not only members of the Jewish community, but also Sinti and Roma, homosexuals, Jehovah's Witnesses and other people who were otherwise persecuted for political or religious reasons (ibid.). As of April 2017, the project counts with over 60.000 *Stolpersteine* placed in about 1.200 cities around Europe. The project is still active with blocks still being placed in Berlin and in other European cities and any voluntary can join by researching about the life and fate of those persecuted and applying to lay a Stolpersteine (see *Technical* at Demnig, n.d.).

Due to process with which the data is gathered and the fact that it encompasses heterogeneous groups of people, I will avoid to derive conclusions on the broader population based on estimations conducted in the subsequent chapters. This data is rather used for illustration purposes of the underlying estimation and mapping procedures, which is the main focus of this paper.

In Berlin, information about over 7.200 *Stolpersteine* is maintained by the *Koordinier-*

ungsstelle Stolpersteine Berlin. The data can be easily exported and downloaded as a CSV file, (see "Finding Stolpersteine" page at Stolpersteine Art Project, n.d.). The dataset contains information on the name, address, district and birth year of each individual, with very few missing values.

Although the maintainer also presents the data on an interactive map on their website, the exported file does not include exact geolocation of the blocks, only their respective street addresses. In order to extract their exact coordinates, the `geocode()` function of the R package *ggmap* is used and further explored in the following section.

2.2. Geocoding the Stolpersteine Addresses

The `geocode()` function from the R package *ggmap*, through usage of Google Maps Geocoding API[1], takes as input addresses in a human-readable format and returns as output the best-matching geocoordinates, that means, the first result one would get when using the normal Google Maps website. With that in mind, it is important to minimise ambiguity in the addresses entered. The raw data, which includes the district alongside the street name and number, already restricts potential ambiguity to an acceptable level. But the fact that it lacks the zip code indicates that some mismatching can still occur. In order to further reduce the ambiguity level, it is advised to append attributes, such as "berlin - germany" to every query, in order to reduce the scope of potential results. Besides geocoordinates, the `geocode()` function also returns the matched address. With this, one can still compare and evaluate if the function is indeed returning the desired information. For more the complete R script used to retrieve the coordinates, see Appendix A.2.

With the geocoded addresses at hand, the CSV file created with the aforementioned script can be directly imported from JMP and saved as data table (*.jmp) for further analysis. At the importing stage, JMP usually assigns the expected Data Type for each column, that means, the columns containing strings are assigned as *Character* and those consisting of numbers *Numeric*. However, it is possible to let JMP know that the *Latitude* and the *Longitude* columns are actually geocoordinates. For that, the user can open the *Column Info* window [*On Data Table*:Double click on Latitude/Longitude columns] and change the assigned format from *Best* to [Geographic > Latitude DDD] and [Geographic > Longitude DDD], respectively. While this is not strictly necessary, it is helpful to let JMP know that it is dealing with geographic information when plotting the data later

[1]The Google Maps Geocoding API terms of services can be found on `https://developers.google.com/maps/terms`.

on.

2.3. General Background Map Implementation via WMS Servers

Before applying the estimation and visualisation methods, this section is dedicated to explore JMP's background map functionality in a more general setting. This mapping feature is available in any graphing method within JMP (SAS Institute, 2017a, p. 304), including the *Fit Y by X* and the *Graph Builder* contexts discussed in Chapter 3. The only requirement is that the data contains latitude and longitude information in the World Geodetic System 1984 (WGS 84) format, which is the most common system used by GPS and other mapping services. The *Set Background Map* context can be accessed within the *Graph Builder* [*On Graph Frame*: Right-click > Graph > Background map...], and within the *Fit Y by X* contexts [*On Graph Frame*: Right-Click > Background map...].

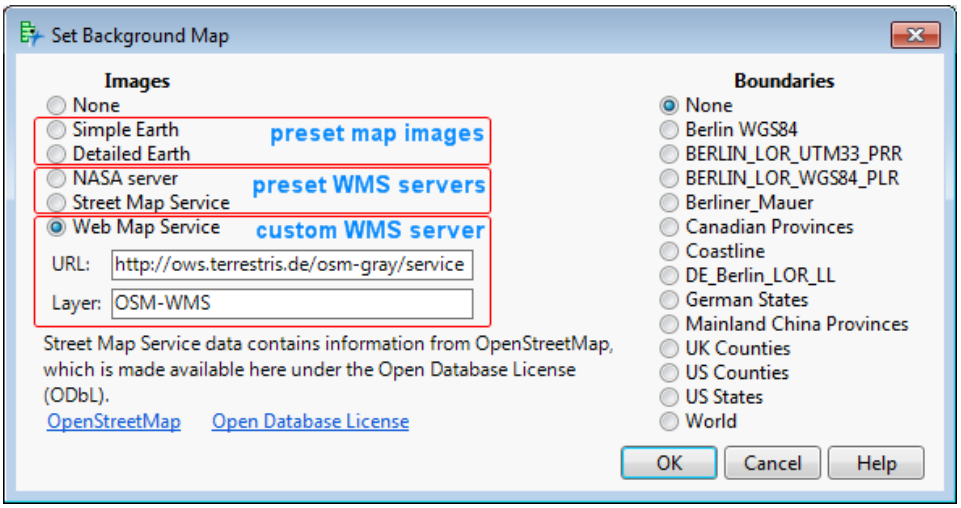


Figure 2.1. Set Background Map window.

The *Set Background Map* context, as shown above, offers both options for map *Images* and *Boundaries*. The Boundaries option offers the user the ability to draw regions, such as districts in a city or states in a country. This functionality will be further explored under Chapter 4. Focusing on the *images* options, the user can select from **Simple Earth** and **Detailed Earth**, which provide pre-installed map images and internet connection is not required. The resolution offered, however, limits the usage only to global or continental visualisation, even if selecting the *Detailed Earth* option. Further, the **NASA Server** requests map images from a publicly available Web Mapping Service (WMS) maintained by NASA. This option requires internet connection, but the resolution is still low for city

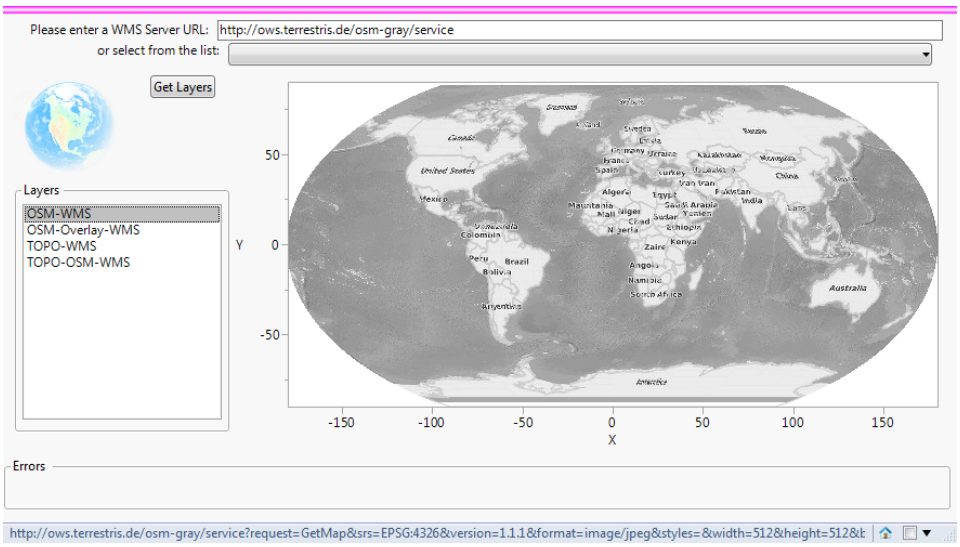


Figure 2.2. WMS Explorer Add-In

or street level applications. The last two options, **Street Map Service** and **Web Map Service** also use the WMS protocol. For the *Street Map Service*, map tiles (i.e. images) are requested through an internal WMS provided by JMP, for which the underlying data are gathered by the OpenStreetMap project, which provides worldwide geographic information collected by its contributors. This option should cover several potential use cases, but if the user has a specific objective, the *Web Map Service*, requests map tiles from a third-party WMS server. In this case, the resolution quality, type of tiles and availability depend on the capabilities offered by the selected server.

In brief, the Web Mapping Service (protocol) is a interface standard for requesting geographic map tiles from an internet server specified by the Open Geospatial Consortium (Beaujardiere, 2006, p. v). This standard specifies the interaction between client and server, such as which area to retrieve at which format, zoom level, coordinate systems, among others specifications. Within JMP, this interaction is quietly executed in the background when the user sets a WMS server. The only information required is the server's URL and the desired *layer(s)*. Some servers provide only one layer per URL, while others provide multiple, which might include supplementary information, labels or other geographic features. Unfortunately, there is not a naming convention for the layers and depending on the server's documentation, it is not easy to know which layers are available. In order to help identify them JMP provides an external add-in called WMS Explorer[2].

[2]More information and the download link of the add-in can be found on: `https://community.jmp.com/t5/JMP-Add-Ins/WMS-Explorer-Add-In/ta-p/21334`

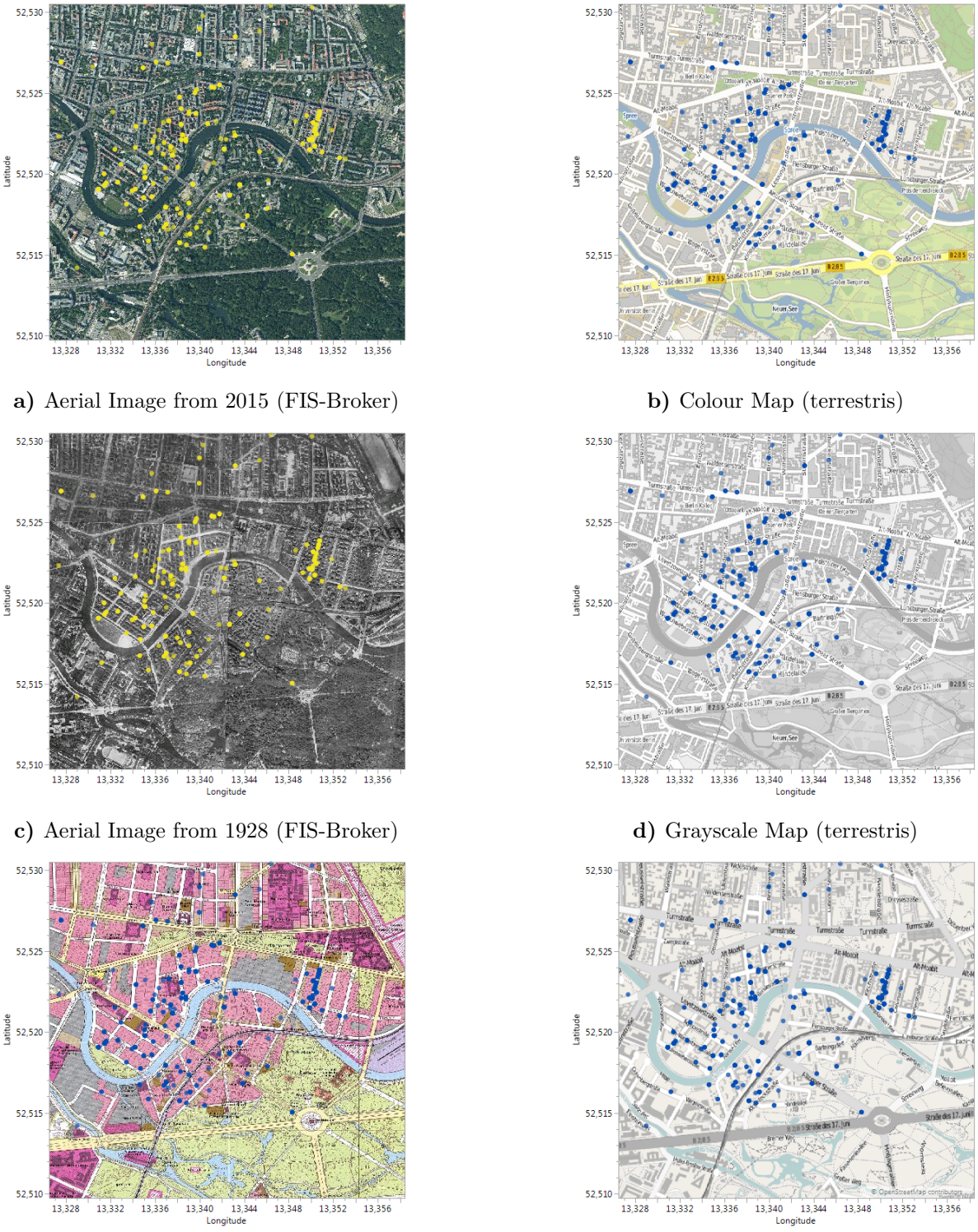
After installed, the add-in can be launched [*On Main Toolbar*: Add-Ins > Map Images > WMS Explorer] and the user can enter the WMS server URL and click on *Get Layers* to retrieve the map tiles provided by the server. After selecting one of the layers, the area covered by the WMS server and layer is displayed on the world map. In case an error occurs, it will be displayed on the *Errors box*. Further, the actual HTTP request that is sent in the background, - or part of it - is shown on the status bar on the lowermost area of the window. As shown in Fig. 2.2, the terrestris WMS server offers under the given URL four different layers covering the whole globe. After identifying a suitable URL and layer configuration, the user can insert this information in the required fields in the Background Map window.

2.3.1. Third-Party WMS Server Comparison

Berlin's Senate Department for Urban Development and Housing for Berlin has a vast offer on public WMS servers through its FIS-Broker[3] service, covering several themes, including historical maps and historic and recent aerial images, which can be useful when working with historical data (Senate Department for Urban Development and Housing, n.d.[a]). Further subject areas involve climate and energy, air and water quality, social-economical development, among many others. The tiles mostly cover only the Berlin's area and occasionally surrounding areas as well. Unfortunately, not all servers support the WGS 84 coordinate system required by JMP's background map functionality. Also, some servers offer a very limited range of zoom level. This means that the map might disappear when the user focuses too close or zooms too far out. This might appear that the server is down or it doesn't support the required layer or coordinate system. An online tool created by Pantelis Rodis (2015) can be very useful for diagnosing such problems and also for further enquiry on the WMS capabilities. Moreover, if the user requires global coverage, a public WMS server is provided by *terrestris*, a company situated in Bonn.

For a short list of WMS servers and their visualisation results, see Fig. 2.3. These selected examples illustrate the variety of themes available for data exploration and presentation. Having access to historical maps or aerial images - as in Subfigs. c) and e) - can be useful when working with historical data, due to potential change in urban and geographical characteristics. Note, however, that the response of aerial images and historical maps are rather slow. It might be, therefore, better advised to conduct the

[3] A list of over 380 WMS servers and related offerings can be found on: `http://fbinter.stadt-berlin.de/fb/index.jsp` (accessed: 11.10.2017).



a) Aerial Image from 2015 (FIS-Broker)

b) Colour Map (terrestris)

c) Aerial Image from 1928 (FIS-Broker)

d) Grayscale Map (terrestris)

e) Berlin Map from 1940 (FIS-Broker)

f) Built-in OpenStreetMaps (JMP)

Figure 2.3. Comparison between built-in and selected third-party WMS servers. For exact layer and URL information, refer to Appendix A.1 on page 37.

earlier stages of data exploration via a traditional WMS server and in a later stage refer to these historical maps only for very specific examination. Exact information on the URL and layer used in each example can be found under Appendix A.1 on page 37.

One issue present within JMP's WMS integration is that the output maps suffer a slight distortion. More precisely, they get vertically squeezed and it also affects the street labelling, which becomes hard to read in some situations. Moreover, the requirement to have the data under the WGS 84 coordinate system reduces this functionality considerably relative to the usage of Berlin's public data, which is often offered under other coordinate systems, such as *UTM 33 N* and *Soldner Berlin*. The user is, therefore, forced to convert the data previous to the analysis. This restriction seems to be somewhat abstract, since there are WMS servers capable of delivering the mapping images under other coordinate systems, including the publicly available servers aforementioned.

3. Non-Parametric Density Estimation for Spatial Analysis

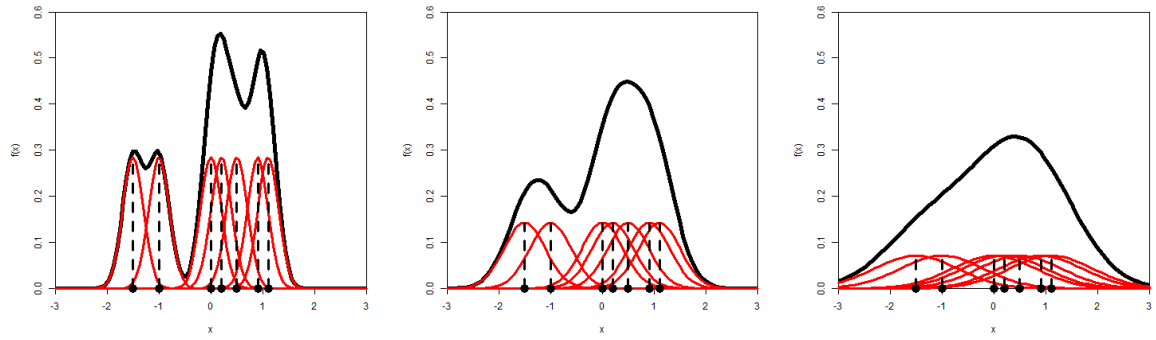
In this chapter I present two methods of implementing non-parametric density estimation within JMP. In Section 3.1, I briefly present the main aspects and the intuition behind the kernel density estimation method. In Section 3.2, I explore JMP's implementation for a bivariate variable within the *Fit Y by X* platform. Further, in Section 3.3, I present an alternative estimation method via the *Graph Builder* platform. These methods differ in their respective estimation and implementation procedures, each generating distinctive results and providing the user with a broad set of options that can be applied relative to the users main objectives.

3.1. Kernel Density Estimation

In general, a probability density function is one way of describing the distribution of a random variable. In practice, the shape of the density function is not known, but an estimation can be derived based on observed data. When it is plausible to make assumptions about its distribution, such as of normal distribution, one can compute the relevant parameters (i.e. mean and variance) from the observed data and, from that, derive a density estimation. This approach belongs, therefore, to the *parametric* estimation methods. When dealing with spatial data analysis, however, it is not trivial to define sound assumptions about the data distribution and a *non-parametric* approach might be better indicated. The Kernel Density Estimator (KDE) with kernel $K(\cdot)$ and bandwidth matrix $\mathbf{H}$ is computed by (Scott, 2015, p. 164; Silverman, 1986, p. 76):

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} \sum_{i=1}^{n} K\left(\mathbf{H}^{-1}\left(\mathbf{x} - \mathbf{x_i}\right)\right). \tag{3.1}$$

An intuitive interpretation of Eq. (3.1), as illustrated in Fig. 3.1 for the univariate case, reveals that the KDE method is the sum of *bumps* created around each data point. Given that $K(\cdot)$ satisfies the moment conditions for a probability density function (see Scott, 2015, p. 165), the kernel function delineates the *shape* of each bump and the bandwidth matrix (also called smoothing parameter or window width) defines the width or *precision*



a) h = 0.2 **b)** h = 0.4 **c)** h = 0.8

Figure 3.1. Bandwitdh parameter comparison illustrating an a) under-smooth, b) plausibly correct and an c) over-smooth estimation.

of each *bump*. In this illustration, based on the example presented by Silverman (1986, p. 14), the Gaussian kernel functions for each data point are depicted in red and the kernel density estimation, i.e. their sum, plotted in black. Naturally, one cannot infer a great deal about their density based only on seven data points, but it illustrates well the drastic effects of the bandwidth parameter on the overall results. More precisely, Subfig. a) illustrates the appearance of spurious spikes with the choice of an overly low bandwidth parameter. Further, Subfig. b) depicts a plausible choice for the parameter, while Subfig. c) represents an over-smooth estimation, where most of the more detailed information of the underlying data points, spurious or not, goes missing.

A vast literature is devoted to optimum selection approaches. Unfortunately, JMP's documentation does not provide information on which kernel function it uses and only states on how it computes the default bandwidth selection for the univariate case (*JMP 13 Quality and Process Methods*, p. 241). Nonetheless, the bandwidth parameter (in JMP named as *Kernel Std*) can be adjusted by the user and a few examples are explored in Section 3.2.1.

3.2. Implementation via *Fit Y by X* Platform

The bivariate kernel density estimation is available in JMP from the *Fit Y by X* platform. This context offers the user a large set of options and customisability, through which elaborated visualisations can be generated with a fine control over the resulting output. In this section, I present the procedure required for implementing a kernel density estimation and in section Section 3.2.1 a few use cases are explored to illustrate a few of the available

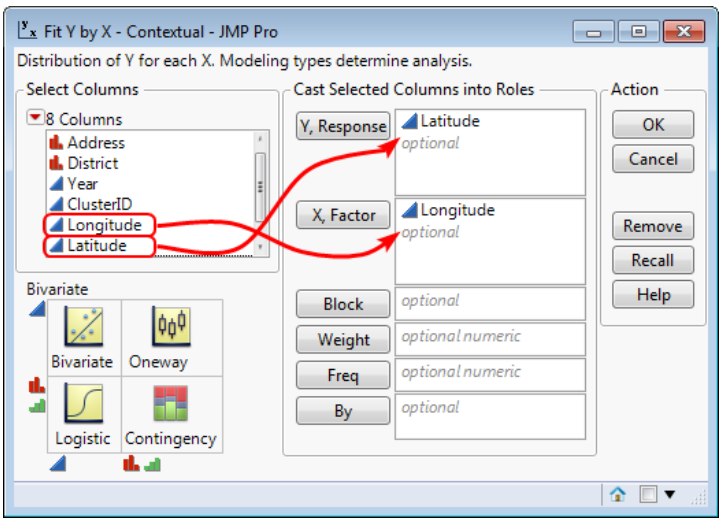


Figure 3.2. Initiating the bivariate kernel density estimation via the Fiy Y by X platform.

options and customisability.

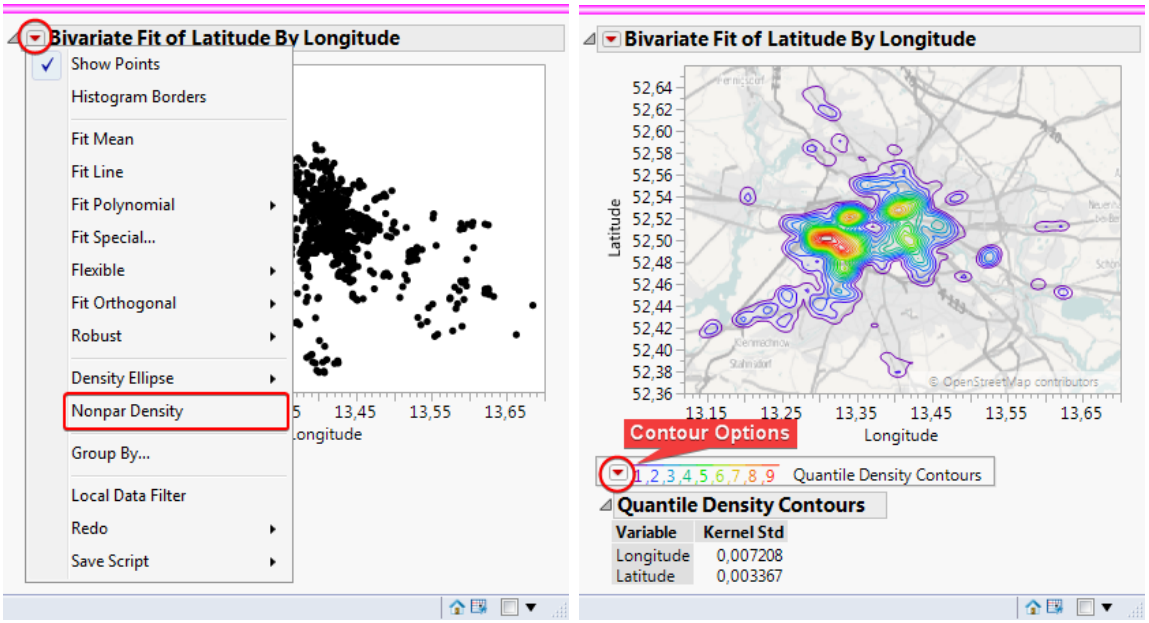
With the Stolpersteine Data Table opened, the implementation can be initiated as depicted in Fig. 3.2:

1. *On Data Table Toolbar*: Select Analyse > Fit Y by X (or click on shortcut),
2. drag *Longitude* to role *X, Factor*,
3. drag *Latitude* to role *Y, Response*,
4. (when necessary: drag *Frequency variable* to *Freq*,) and
5. click OK.

The user is firstly presented with a scatter plot as in Fig. 3.3 a). For the actual density estimation,

6. select *("Bivariate...")* > Nonpar Density. (Optionally: hold `SHIFT` to change default "points for side of grid".)
7. select *("Bivariate...")* > Show Points (to hide/show markers) and
8. add a background map as discussed in Section 2.3.

Note that the values for Stolpersteine are not aggregated in this data set, meaning that each row contains information for only one observation. In case the frequency of the analysed variable was aggregated over unique addresses, the user would need to set the aggregated data column to the *Freq* role, as indicated in step 4).



a) Nonpar Density implementation **b)** Intermediate results

Figure 3.3. Implementation and intermediate results of non-parametric density estimation over a background map within the *Fit Y by X* platform.

Regarding the calculation methods, JMP's documentation does not provide exact information on the algorithm used. It is not very clear for the user how some of the available settings will affect overall results. The user is left to guess that some sort of binning algorithm is in place, which is further discussed under Section 3.2.1.

With the described steps completed, the user is presented with a density estimation plot over a background map as displayed in Fig. 3.3 b). The default settings create contour lines for quantiles in 5% intervals. Which means that about 95% of the points are covered by the outermost line (in purple), 90% by the next one, and so forth until the innermost contour line (in red), containing around five percent of the points. Furthermore, the options available under the *Quantile Density Contours* menu (see Fig. 3.3 b)) offers the user more ways of visualising and exploring the data. Some of the available options and a brief description of their behaviour are the following:

Kernel Control Shows or hides the kernel controller, with which the user can separably adjust the bandwidth parameter of each variable. Refer to Fig. 3.1 for a theoretical and Fig. 3.5 for an practical illustration of this option.

5% Contours Shows or hides the 5% contour lines.

Contour Lines Shows or hides all contour lines.

Contour Fill Fills the areas between the contour lines.

Select Points by Density Selects points between the lower and upper probabilities. This is useful if the user, for instance, wants to further explore patterns among observations that lie in high-density areas versus the ones in low-density.

Color By Density Quantile Colours the markers according to their density estimation. This might be useful if the user intends to keep all markers visible alongside the contour lines. The applied colours remain active under other contexts, which can be very useful when searching for correlation between variables.

Save Density Quantile Creates a new column to the data frame containing the density quantile estimation for each observation, which is also useful for further exploration of the data.

The above customisation settings can cover several use cases. In Section 3.2.1 I explore a few of the options available to illustrate some of the exploratory and visualisation potential offered by the non-parametric density estimation via the *Fit Y by X* platform.

3.2.1. Use Cases and Comparisons

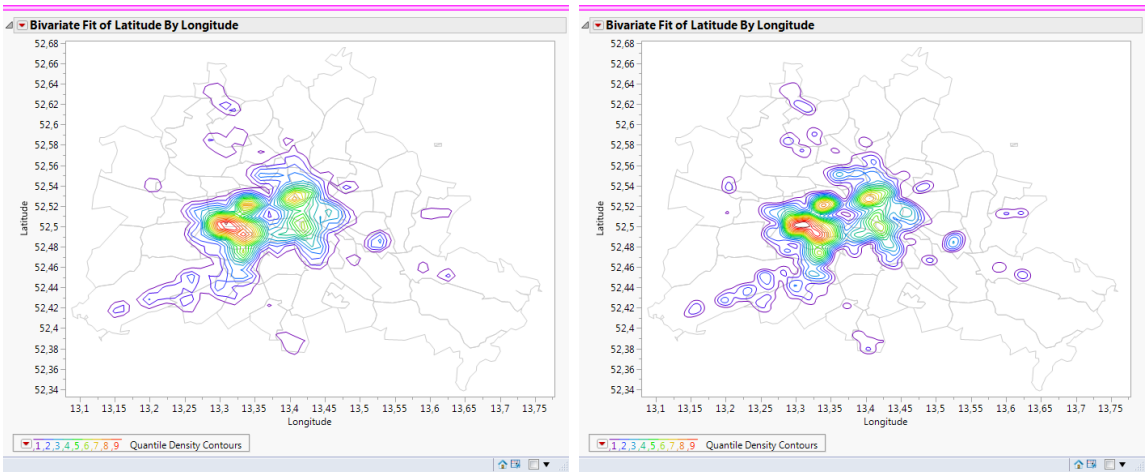
To illustrate the effect of a few applicable options, in this subsection I present a side-by-side comparison among a few of the different settings available. I first examine the variation caused by the choice of number of bins and bandwidth parameter. Further, I explore one use case that uses a mixture of the available options in order to illustrate the customisability potential accessible to the user.

Grid Size Comparison

As discussed above, JMP's documentation does not provide profound information the algorithm implemented for its kernel density estimation (KDE) procedure. Thus, it is not very clear for the user how this setting will affect overall results. The documentation also does not state what the default value of 51 means (see *JMP 13 Quality and Process Methods*, p. 121) and which possible values this setting can take. There are two common grid implementations in KDE procedures. One relates to efficiency optimisation, where the data points are assigned to a given number of discrete values, thus forming a grid. This is done in order to reduce the amount of evaluations needed when computing the estimation (see Section 3.1 from Fan and Marron, 1994). On the other hand, this grid setting might only refer to the visualisation output. That is, after performing the estimation calculations for each data point (binned or not), the software creates a grid of discrete points and computes estimation for each point in this grid, based on which the

software can plot the density contour lines. In both cases, the higher the grid size, the more fluid will the contour lines appear. The default number of 51 seems also to be the minimum allowed. If the user inputs a lower number, the output will look exactly the same, although no warning or error message is displayed. In order to change the default grid side, the user can hold the `SHIFT` key while selecting *Nonpar Density*, as described in step 6) from the implementation instructions on page 12.

With the default value of 51, the output is displayed in Fig. 3.4 a). The visual results look very pleasing but it can produce some jagged contours lines in some areas. For the data at hand, increasing the number of bins to 200, with the results shown in Fig. 3.4 b), generates *more fluid* lines and it still computes almost instantly. While the impact is not very drastic, the user can observe that the effect is not only a cosmetic one, but also some of the quantile contours disappear or take slightly distinct shapes in some areas. Note that other parameters are kept constant at their default values and a boundary map instead of a background map was used for better emphasis on the contour lines.



a) default value: 51 **b)** user-defined value: 200

Figure 3.4. Grid-size comparison: A higher value produce visibly smoother contours.

Kernel smoothing parameter comparison

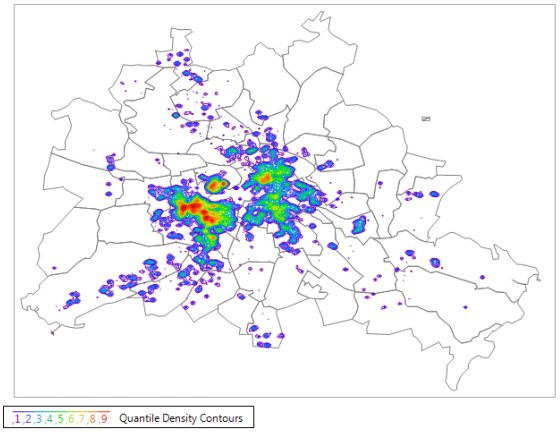
Another very important option available to the user that can drastically affect the overall estimation and visualisation results encompasses the bandwidth parameters. Within JMP, these parameters can be adjusted separably for each variable via the *Kernel Std Slider* after activating the *Kernel Control* option as described in page 13. The effect of adjusting the *Bandwidth* parameter can be intuitively interpreted as changing the standard deviation of the chosen kernel function, but this nomenclature is not commonly used in

other applications. For that reason, I maintain the *Bandwidth Parameter* terminology in this paper.

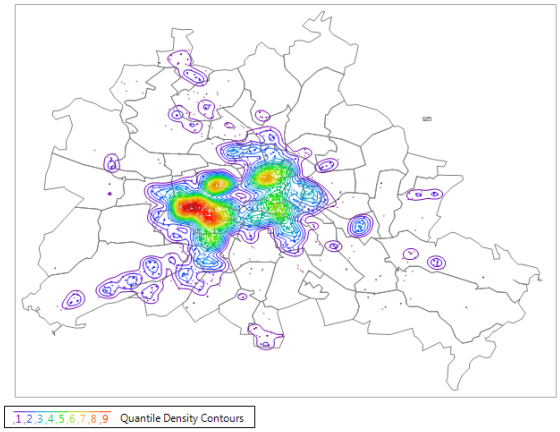
Several sophisticated techniques are available for smoothing parameter selection (for a in-depth comparison refer to Scott, 2015, pp. 190-193; and Silverman, 1986, pp. 84-88). Silverman (1986, p. 44) also makes the case for a *subjective* approach for the parameter selection, where the user makes usage of the estimated visualisations with different smoothing parameters and chooses one according the one's prior knowledge about the data structure and other related information. The visual results of the estimation are more easily distinguishable in the univariate case, but it can also be used with bivariate variables. As a rule-of-thumb, the user should aim at the smallest smoothing parameter before it starts to create spurious *spikes* in the estimation results. This is easier said than done, and when dealing with geographic data, in contrast to other data that follow a more or less random generating process, it might be difficult to distinguish if certain *spikes* are indeed spurious or relative to observations clustered in small areas.

An illustration of the effects caused by different smoothing parameters and an use case of the intuitive selection approach are depicted in Fig. 3.5. From Fig. 3.5 a) to d), the smoothing parameters applied take increasing values, while keeping a proportional ratio between Latitude and Longitude. The estimation using default values are shown in Subfig. b). Unfortunately, JMP's documentation does not state how the default values are calculated[1]. The parameters on Subfig. a) is one fourth of the default ones from b). In c), the values 50 percent higher and in d) five times as high. The result in Subfig. a) illustrates an under-smoothed estimation, where several *spikes* appear indicating an overly precise estimation in some areas that most likely does not represent the real (unknown) density. On the other hand, d) depicts an over-smoothed estimation, where the *spikes*, spurious or not, are lost in the estimated results. When comparing Subfig b) to c), it is difficult to affirm either way. While there are still some *spikes* in a few areas distant to the city centre, the underlying geographic and urban characteristics might explain some of these clustered areas. The results in Subfig. d) solves most of those *spikes* but some (non-spurious) information might be lost along the way. Either way, the results don't differ significantly from each other. It is also important to note that the reader can *smoothen* the estimation from an under-smoothed one, but not the opposite. That means from Subfigs. b) or c), and even a) the reader might be able to mentally

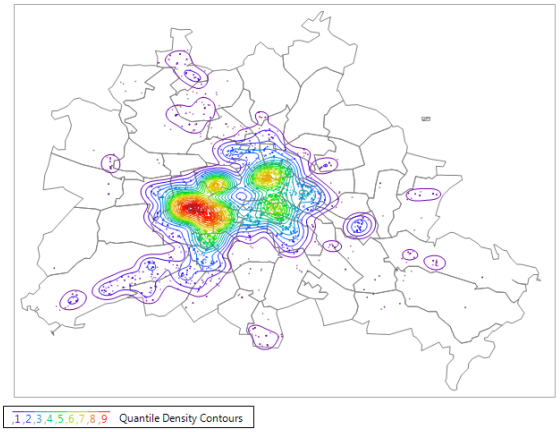
[1]In a 2009 post in JMP's community forum, a unnamed staff explains that default values estimation method are not included in the documentation due to its complicatedness. See: `https://community.jmp.com/t5/Discussions/Default-Kernel-Std-in-Smooth-Curve-fit-Distribution-Platform/m-p/768`. Last accessed: 30.10.2017



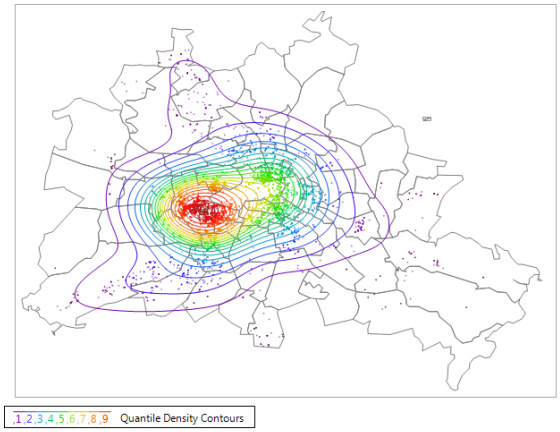
		Kernel Std
a)	Latitude	0.001802
	Longitude	0.000842



	(default values)	Kernel Std
b)	Latitude	0.007208
	Longitude	0.003367



		Kernel Std
c)	Latitude	0.010812
	Longitude	0.005051



		Kernel Std
d)	Latitude	0.03604
	Longitude	0.01684

Figure 3.5. Kernel density estimation with different bandwidth parameters. The parameters were set in relation to the default values in b), so that:
a): $0.25 \times default$; c): $1.5 \times default$ and d): $5 \times default$ *values*.

project an estimation similar that in d). But from the results in d) it is not possible to infer a more precise density of the underlying data.

It is also a valuable feature that the user can set the bandwidth parameter for each of the variable separably when dealing with spatial analysis under the WGS 84 coordinate system, considering its non-equidistant scaling in each axis, which is also exacerbated in areas far from the equator.

Note that for this visualisation, the data points were left visible, and the option *Colour by Density Quantile* were applied in order to better compare the estimation results to the underlying data points. Also, the background map was replaced by the plain boundary lines in order to emphasise the estimation contours, but the user might find useful, after conducting the estimation, to reintroduce a background map for closer analysis of the urban and geographic characteristics, in order to make a better educated decision on which result or results to keep.

Use case for data filtering and emphasising on dense areas

To illustrate a common use case that presents the reader with an overview of the available customisability, in this subsection I explore a scenario with the objective of comparing the distribution of observations of two distinct groups. Further, assuming an use case where the user wishes to give emphasis only to highly dense areas and produce a less cluttered visualisation, I discuss the steps necessary for such implementation. More precisely, I create two distinct groups and implement for each a density estimation where only the 0.5 to 0.9 deciles contours are visible, as depicted in Fig. 3.6.

The contour lines are customisable through *Customize Graph* window [*On Graph frame*: Right-click > Customize...]. Under the *Quantile Density Contours* options, the user can change the colour, style and width for each of the 19 originally created contours (one for each quantile between 0.05 and 0.95 in 0.05 increments). To achieve the proposed results, as shown in Fig. 3.6, the width of all contour lines can be set to zero, except those representing the contours that should be emphasised. The user can also widen the lines intended to remain visible (here the width was raised to three). This can be a tedious process, since the user has to adjust the settings for each contour line separately, but after done it once, the user can save the JSL script [*("Bivariate...")* > Save Script > To Script Window] and apply the same settings to other data sets by modifying the script accordingly.

Since the data at hand has no categorical variables, on which to base the grouping, two groups - A and B - were created based on their birth years. One way of creating

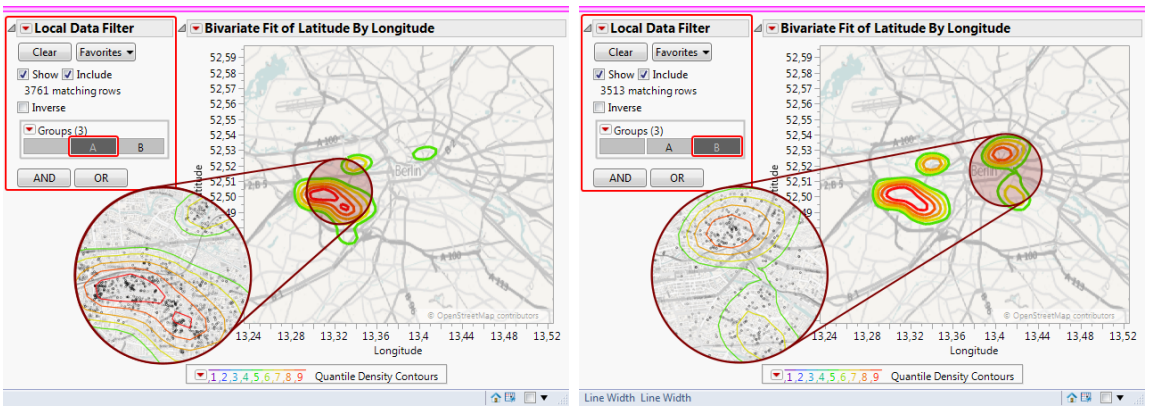


Figure 3.6. Use case of visualisation with *Local Data Filter* emphasising only highly dense areas and depiction of interactive data exploration on a street level.

groupings conditional on data columns is via a creation of a *formula* column. For that, the user can create a new column [*On Table Header*: Right-click > New Columns...] called *Groups*, set *Data Type* to *Character* and *Modelling Type* to *Nominal*, so to divide into *Group A* those whose birth year was before 1890 and *Group B* those who were born afterwards with the following formula [*Within New Column Windown*: Column Properties > Formula],

```
If( Is Missing( :Year ), Empty(), If( :Year < 1890, "A", "B" ))
```

which results in group A with 3760 observations for those born before 1890, and group B with 3514 individuals born after, apart from ten missing values. The year of 1980 was chosen rather abstractly with the only intent to have two roughly equally populated groups. But one can also interpret group A as those individuals older than roughly 50 by the time of deportation and group B with the individuals younger than that.

Within the *Fit Y by X* context, a filter for the variable *Groups* was created [*("Bivariate...")* > Local Data Filter] for comparison between the observations of Group A and B. This inserts a new pane on the left of the graph where the user can select which group to be estimated and visualised. Note that a third group for the ten missing values were also created. Multiple groups, if intended, can also be selected with Ctrl + Left-click. Note that the *Local Data Filter* feature is not exclusive to this estimation procedure. It can be also employed within the *Graph Builder* and others visualisation contexts.

From the filtering procedure, it is possible to observe a distinct pattern in the residence of those born before 1890 and those born after. While most of the data points overlaps in its highly dense areas, there are some agglomerations of younger individuals in the north-eastern area of the district Mitte, in constrast to the older individuals, which

is displays bigger clusters in the boundary between Charlottenburg and Willmersdorf districts. The ability to deeply focus on small areas on a street level resolution makes it easy to pinpoint areas of interest - such as highly dense areas - for further analysis and exploration, as depicted by the in Fig. 3.6. Note that, for the street level visualisation, it might be useful to display the data points again [*("Bivariate...")* > Show Points]. For data exploration, the *Graber* and *Magnifier* tools hidden underneath the top (pink) bar makes it easier to pan and zoom in and out, respectively. Note also that the *zoom* depicted as circles in Fig. 3.6 is not a JMP feature, but only an illustration of the interactivity available to the user.

This illustrates a rather specific usage of those settings, but it helps to highlight some of the options available to the user within the *Fit Y by X* context, from which its main functionality can be extrapolated to other common scenarios. The non-parametric estimation, together with the *Local Data Filter* and street level exploration over a background map offers the user a very high degree of customisability and fine control over data exploration and pattern recognition besides output visualisation and presentation.

3.3. Contour Plot via *Graph Builder* Platform

Another way of creating a non-parametric density estimation within JMP is available via the *Graph Builder* context. While user-friendlier than the *Fit Y by X* case, this approach offers less detailed control over the estimation procedure. Regarding the actual estimation method, JMP's documentation also does not offer in-depth information to the user. It only states that "the contours are computed using Delaunay triangulation" (SAS Institute, 2017a, p. 60). In general, this technique uses the data points for splitting an area into triangles, while ensuring that a circle circumscribing the vertices of one triangle does not overlap with vertices of any other (Daintith and Wright, 2008). This triangulation method avoids long and thin triangles and the density of the data points is inversely proportional to the area of the triangles surrounding the data points. That means, a highly dense area will produce several small triangles, while a scarce populated area creates fewer and larger triangles. Since the exact algorithm method is not disclosed to user, I only present here the actual implementation in the following pages without going any further into technical and algorithmic aspects.

With the *Graph Builder* window opened [*On Toolbar: Graph > Graph Builder*] as shown in Fig. 3.7, the user can 1) assign the longitude and latitude variables by dragging and dropping each variable to the respective horizontal and vertical axis, 2) change the

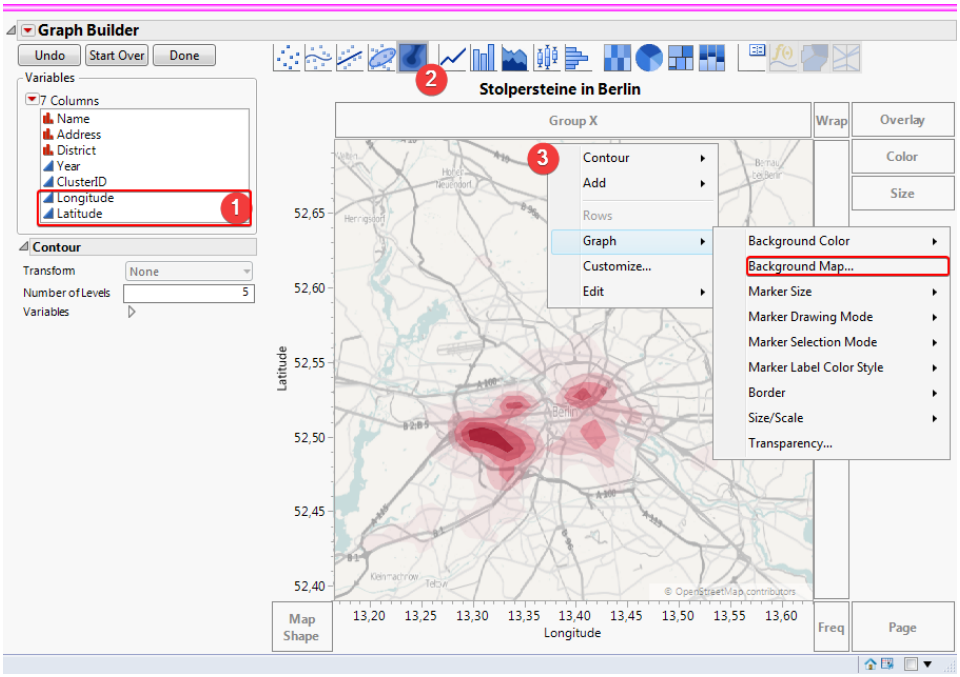


Figure 3.7. Implementation of a Contour Plot over a Background Map using Web Mapping Service functionality.

visualisation to a *Contour Plot* [Click on  icon] and 3) add a map layer - as described in Section 2.3 - to the graph [*On graph frame*: Right Click > Graph > Background Map...]. The user is then presented with a non-parametric bivariate density estimation of the Stolpersteine data over an interactive background map.

The official documentation states that the four contour levels created by default are 100%, 75%, 50%, and 25% density contours (*JMP 13 Essential Graphing*, p. 60). This information seems to be incorrect, since it is clear that a substantive percentage of the data lies outside the outermost band. Contradicting the documentation, a registered JMP staff named Xan Gregg states in the *JMP community forum*[2] that "about 10% of the data will be outside the outermost band", meaning that the outer band will contain 90% of the data. The rest of the data points are evenly distributed within each remaining contour level. That means, if the user chooses 3 levels, each band, starting from the outermost will comprise about 90%, 60% and 30% of the data. This information seems to be consistent with the achieved results and when comparing to the *Fit Y by X* method explored in Section 3.2.

Without the intent to exhaust available settings, an interesting feature of the *Graph*

[2]The discussion can be accessed on `https://community.jmp.com/t5/JMP-Blog/New-in-JMP-9-Graph-Builder-Density-Contour-Element/ba-p/29933` (Accessed: 20.09.2017). Also note that the outermost band may be difficult to visualise over the background map layer.

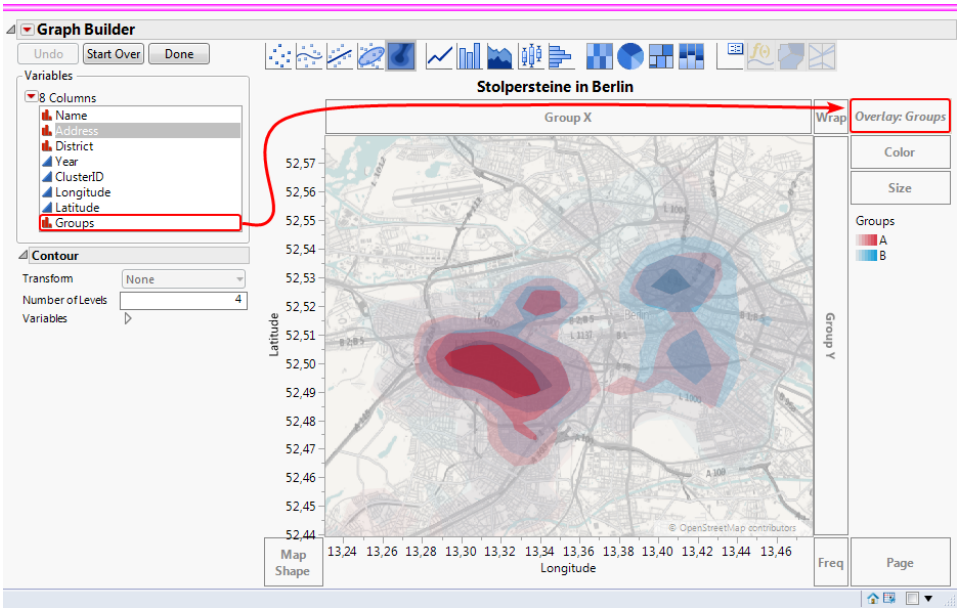


Figure 3.8. Density estimation of individuals born before 1890 (Group A, in red) and those born after (Group B, in blue) illustrating the *Overlay* function.

Builder utility, which is not available under the *Fit Y by X* method, is the ability to differentiate and visualise groups of individuals within the same graph. This can be done through the *Overlay* functionality.

With the *Groups* variable created in Section 3.2.1, the overlay functionality can be used by dragging and dropping the intended variable into the overlay box, as shown in Fig. 3.8. While most of the data overlap in the same areas, one can observe a general clustering behaviours, as also noticeable within the *Fit Y by X* context. In a more general case, this can be a powerful feature for quickly exploring differences in spatial density between two or more groups of observations based on a categorical variable. And, while it offers less detailed customisation, it can be preferable to have both estimation within the same graph for visualisation and presentation applications.

4. Implementation of a Choropleth Map

For the procedure discussed in Chapter 3, exact geocoordinates of the data are required. Due to privacy and confidentiality concerns, however, exact geo-coded data is not often available. In these cases, the information are commonly aggregated over pre-defined boundaries before becoming accessible to the public. A common procedure for exploratory spatial analysis for such situations involves the creation and the visualisation of the data on a choropleth map. In this chapter, I present the necessary steps for a choropleth map creation, including preparation and conversion steps in Section 4.1, dealing with custom map shapes in Section 4.2 and the actual implementation in Section 4.3.

For this implementation, the Stolpersteine data is also used in order to provide the reader with a better comparison between the choropleth procedure and the results obtained via the non-parametric density estimation discussed in Chapter 3.

4.1. Data Preparation and Conversion

A choropleth map (also called thematic map) is a technique, where the data is first aggregated over disjunct areas and an uniform colour or pattern is used to depict a certain statistic on the analysed data. This technique requires not only the data to be studied, but also boundaries over which to aggregate and plot the data. For this implementation, the *Lebensweltlich orientierten Räume* is used as the boundaries, in which the Stolpersteine frequency is to be plotted.

4.1.1. Lebensweltlich orientierten Räume

The Lebensweltlich orientierten Räume (henceforth LOR) were created by the Bureau for Statistics Berlin-Brandenburg to serve as a geographical basis for planning, forecasting and observing social and demographical development in Berlin (Senate Department for Urban Development and Housing, n.d.[b]). The boundaries creation took several factors into account - such as building structures, sociological environment, traffic roads, natural barriers, population density, among many others - in order to ascertain a level of consistency among each block, thus making LORs a better tool for statisticians and urban

planners than more abstract alternatives (Senate Department for Urban Development and Housing, n.d.[b], paragraph *Das Ziel*). The LORs are structured into three hierarchical layers with growing granularity. The first layer, called prediction-areas (Prognosenraum) contains 60 cells. The second, district-areas (Bezirksregion), contains 138 and the last, the planning-areas (Planungsraum), contains 447. Each of these layers are also contained within each of the 12 districts of Berlin. So, technically, there are four hierarchical layers in total. The LOR vector files are also made publicly available by the Senate's website (ibid., section *Download*) in different file formats and coordinate systems.

There are several possible constellations between file formats and coordinate system of the vector data and the underlying data of study. In the following sections I discuss the steps necessary to convert the provided vector file as shapefile format from *UTM33 N* into WGS 84 coordinate system, but the alternative paths can be taken with little modification. It is only necessary to convert into the WGS 84 coordinate system if the user intends to integrate the choropleth map with a street-level background map, otherwise a choropleth map can be plotted in any coordinate system.

4.1.2. The *ESRI* Shapefile Format

In brief, the *ESRI* shapefile format is a specification developed by the Environmental Systems Research Institute for storing vector coordinates and related attribute information for geographical data (ESRI, 1998, p. 2). Despite its name, a shapefile format actually contains three mandatory and a few optional files. The mandatory ones are the main (.shp), the index (.shx) and the dBASE (.dbf) files. Although these three files are indicated as mandatory in the original specification, JMP only requires the main file containing the geometry vertices that form each polygon (cell) and the dBASE file containing the attribute features and keys, through which the data table containing the actual data of analysis can be linked.

The coordinate system conversion of the shapefile was done via the *rgdal* package (Bivand, Keitt and Rowlingson, 2017) for the R programming language. Further, the function `over()` of the R package *sp* (Pebesma and Bivand, 2005) was used in order to count the frequency of Stolpersteine contained in each shapefile polygon. For detailed information on the scripts, refer to Appendix A.3 on page 42. The resulting data table consists of 447 observations and five columns. A district column, three columns containing the unique index numbers to each LOR layer and one column with the Stolpersteine count.

4.2. Creating Custom Map Shapes

Generally, the creation of a choropleth map within JMP requires two sets of data tables. One is a series of geocoordinates - the *XY-Table* - for each vertices that will form the polygons upon which the aggregated data will be drawn. The other is a set of indexed locations - the *Names-Table* - matching each entry to a specific polygon in the XY-Table through an unique ID-number. Both data tables and how relate to each other is shown in Fig. 4.2 on page 26. In order to convert the shapefile into a format readable by JMP the user can follow the two steps detailed below. The instructions are based on *JMP 13 Essential Graphing* (Chapter 15, pp. 325-328), where further explanations on these and other options are also available.

1. Converting the .shp shapefile to a <prefix>-XY.jmp file:
   a) Open the .shp file from JMP.
   b) Save the <prefix>-XY.jmp file under:
      - On Windows: C:/Program Files/SAS/JMP/13/Maps
      - On Mac: /Users/<user name>/Library/Application Support/JMP/Maps
2. Converting the .dbf shapefile to a <prefix>-Name.jmp file:
   a) Open the .dbf file from JMP.
   b) Create a index column with unique value for each row named *Shape ID* and make sure it is the first (leftmost) column.
      - *On Toolbar: Cols > New Columns > Initialize Data > Sequence Data.* Also change *After last column* to *Before first column.* Finally, *Click OK.*
   c) Set *Map Role* definition for the variable containing the identical entries as in the data table of analysis, as shown in Fig. 4.1. Here, the PLR-keys are under the column "SCHLUESSEL".
      - *On SCHLUESSEL column: Right-click > Column Info... > Column Properties > Map Role > Shape Name Definition.*
   d) Save the <prefix>-Name.jmp file in the same location as stated in point 1.b).

It is important to note that the prefix for *-XY.jmp and *-Name.jmp data tables must be identical for the matching procedure to work. Further, it is not strictly necessary to save the Data Tables in the above stated location. In this case, when already working from the actual Data Table of analysis, the user has to manually assign which Data Table

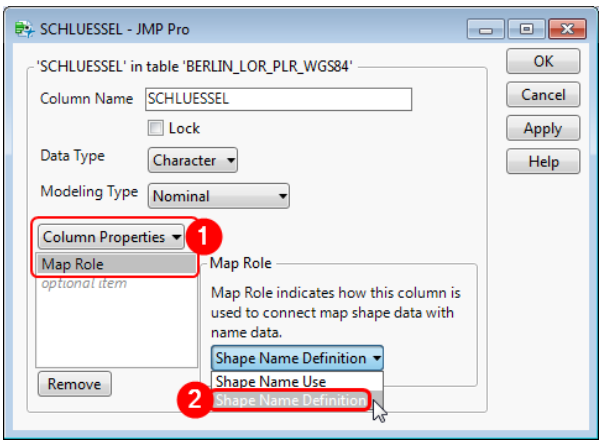


Figure 4.1. Setting Map Role definition for column SCHLUESSEL

and Column to look up for the boundaries, through the *Shape Name Use* settings under the Map Role properties. This can be also useful to avoid conflicting boundary names, since the user can specify exactly which Data Table to look up. But it is recommended to save the converted shapefile in the above location in order to use the *Boundaries* function of the *Background Map* context, which automatically loads up boundaries saved in the default folder.

Another aspect to note is that the shapefile provided by the Senate Department for Urban Development and Housing displays two minor issues. Firstly, the names for a high number of LOR boundaries are partly missing, as marked in yellow in Fig. 4.2. Secondly,

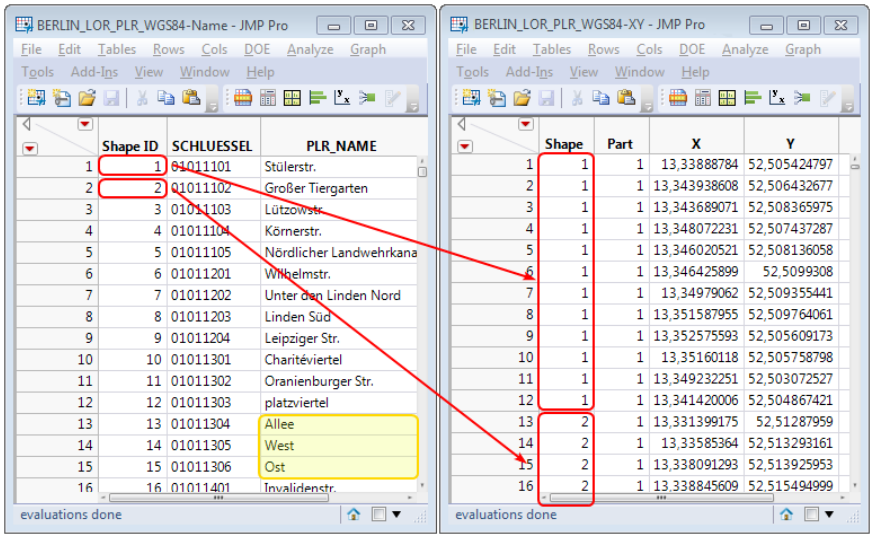


Figure 4.2. Illustration of the relation between the *-Name and *-XY Data Tables. The Shape ID of each polygon in the *-Name table points to their corresponding Shape in the *-XY table. Emphasis (in yellow) to a few errors in the source shapefile.

there are two extra unnamed boundaries in every LOR-shapefile, that ends up drawing a small polygon North-east from Berlin. These issues are present in the unmodified shapefiles in multiple coordinate systems, therefore it was not due to the file format or coordinate system conversion. In any case, none of these issues prevents the choropleth map implementation that is conduct in the next section.

4.3. Choropleth Map Implementation

With the underlying shape boundaries already in place, the actual choropleth map can be easily implemented via the *Graph Builder* platform. In this context the user can 1) set the intended LOR boundary by dragging and dropping it from the *Variables* box into the Map Shape Zone and 2) drag Stolpersteine into the *Colour* Zone, as displayed in Fig. 4.3. Further, the user can change the Axis Scale Type from linear to Geodesic [*On X and Y Axes*: Double-click] to remove the distortion due to the geographical projection or accordingly adjust the *Aspect ratio* setting under the *Map Shapes* options.

With the choropleth map employed, JMP offers a wide range of descriptive statistics and customisation options. Firstly, the user should choose which summary statistics to be displayed from the *Map Shapes* options (marked in blue). Here the user can choose from Sum, Mean, Median, Range and Variance to name a few. Note that since there is only one entry per boundary in the *Plannungsraum* layer (the most granular one), some of the summary statistics options are equivalent (such as Mean or Sum) or inappropriate (such as Range or Variance). These options take into effect when using aggregated data

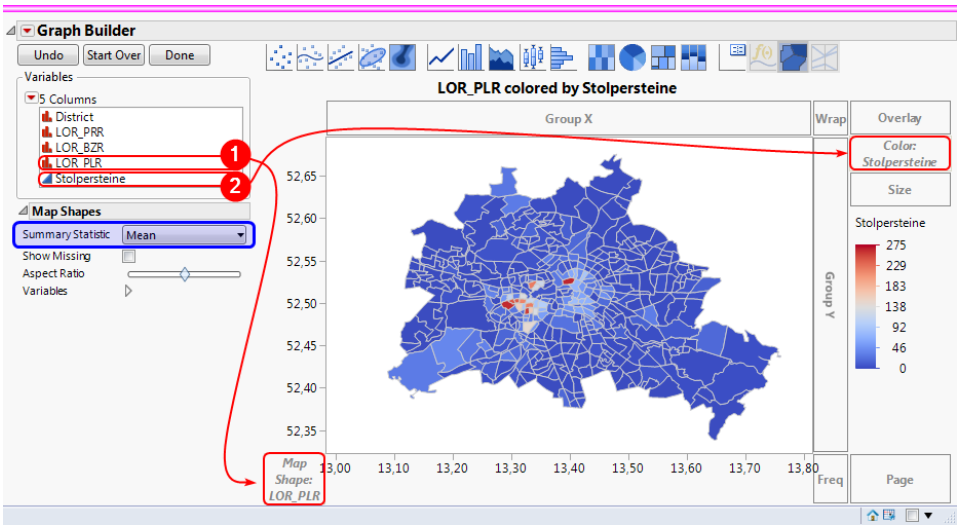


Figure 4.3. Choropleth Map implementation in the Graph Builder context.

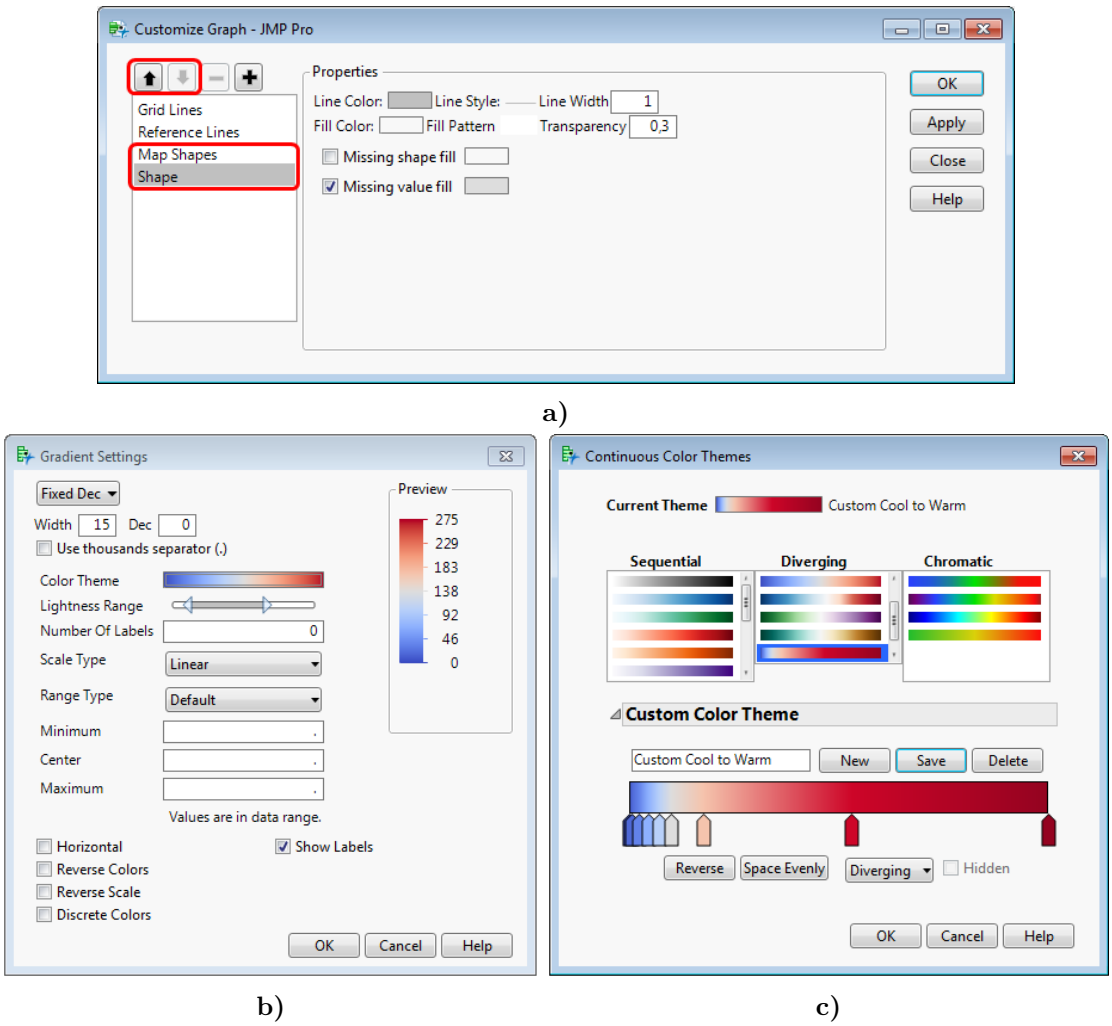


Figure 4.4. Available customisation options for the (a) graph environment, (b) legend and (c) colour theme settings

over multiple entries, as it is the case with the other two LOR layers. Moreover, JMP offers a wide range of customisation options with its *Gradient Settings* [*On Legend Zone: Right-click > Gradient...*] and the *Continuous Colour Theme* contexts [*From Gradient Settings window: Click on Colour Theme gradient box*], as shown in Figs. 4.4 b) and 4.4 c).

It is also often desirable to have multiple boundary layers drawn in the same graph. The user might want to have a detailed layer with numerous cells and at the same time keep a coarser layer with fewer cells for easier spatial identification of the depicted locations. Within JMP, the user can drawn up to two layers containing distinct boundaries. An illustration of such use case is depicted in Figs. 4.5 e) and 4.5 f). This is achieved by adding the *Boundaries* features from the *Set Background Map* context [*On graph frame*: Right-click > Graph > Background Map...]. Note that, after adding a boundary layer

to the map, the user has to make sure it is visible by changing the layer ordering via the *Customize Graph* window [*On graph Frame*: Right-click > Customize...]. The added layer *Shape*, responsible for the boundary lines should be drawn *after* the *Map Shapes* layer, which is responsible for the actual choropleth mapping. This is done via the up and down arrow buttons as shown in Fig. 4.4 a). Further options are accessible under the *Properties* section. For the following implementation, I set the transparency level to 0.3 of the *Shape* layer for better distinction between the boundaries of each LOR layer.

4.3.1. Scaling and Colour Theme Options

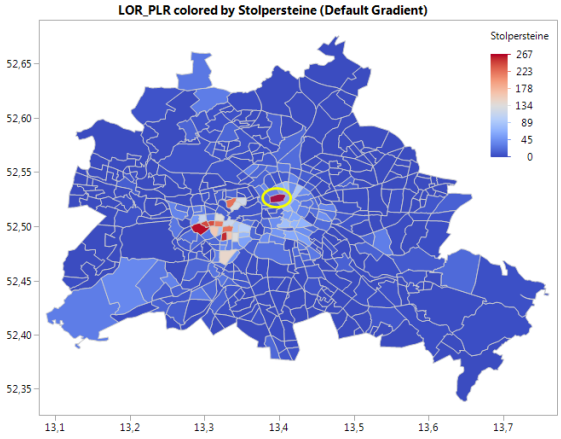
While most options are primarily of aesthetic interest, some of the colour and theme settings can affect the visualisation and also possible inferences considerably. Given the data characteristics, which ranges from 0 to 267 Stolpersteine per cell and half of them contains three or less stones, it is difficult to differentiate across sparsely populated boundaries with the default settings. This issue can be mitigated through the *Gradient Settings* window [*On Legend Area*: Right-click > Gradient...]. The user can, for example, set a new Custom Colour Theme with a stronger gradient contrast gravitating to the lower levels, as shown in Fig. 4.4 c). Alternatively, the user can change the *Scale Type* from Linear to Log. Using a base ten logarithmic scale emphasises the contrast in the lower range. As a side effect, it can become difficult to distinguish between areas where the Stolpersteine are highly concentrated. See Fig. 4.5 for a few comparison examples. Subfig. a) depicts the default results using the *Cool to Warm* gradient theme. Further, b) demonstrates the results using the custom gradient theme with a linear scaling. Moreover, c) illustrates the log scaling with the default *Cool to Warm* theme. And Subfig. d) depicts the exact geocoordinates for comparison purposes.

It is important to note that the scaling and gradient options can drastically change the overall results. These adjustments can be useful when searching for patterns in the data but also makes it easy for the user to create profoundly misleading visualisations, intentionally or not.

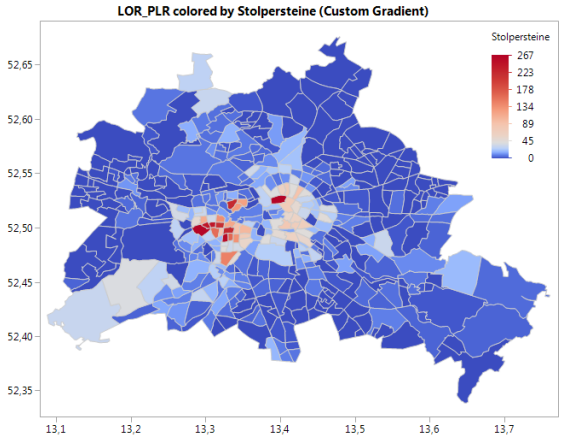
4.3.2. Boundary Selection Effect

Another important aspect to consider when creating a choropleth map is the effect caused purely by the choice of boundaries, which can also affect the resulting interpretation.

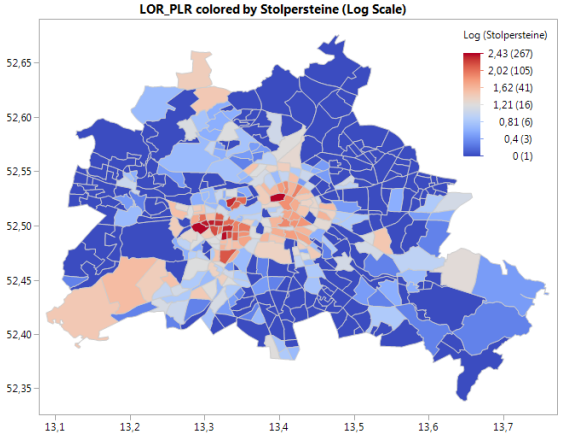
To visualise this effect, compare the LOR boundary marked by the yellow ellipse in the three distinct layers in Figs. 4.5 a), 4.5 e) and 4.5 f). In Subfig. a), using the *Plannungsraum* layer, the most granular with 447 cells, the marked cell is the most dense



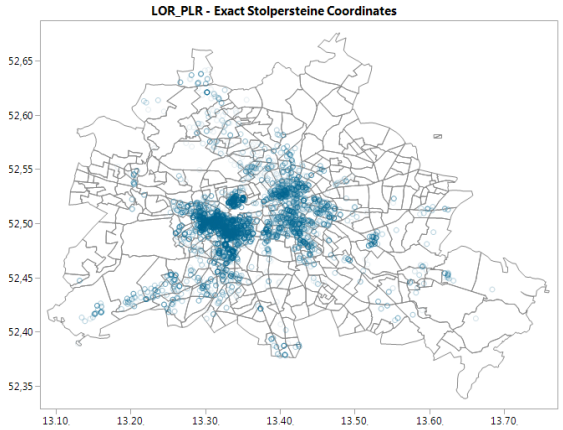
a) Default result using the Cool to Warm gradient



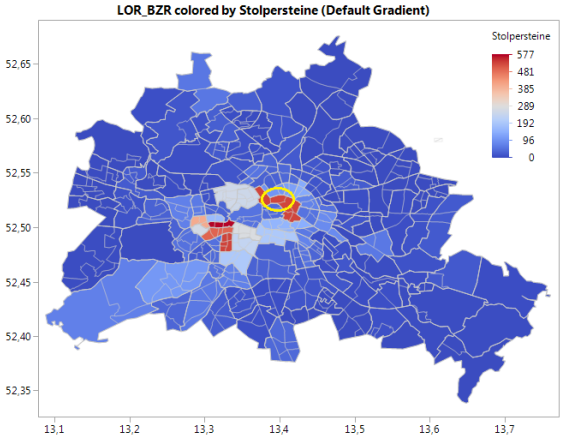
b) Custom gradient theme with a linear scaling



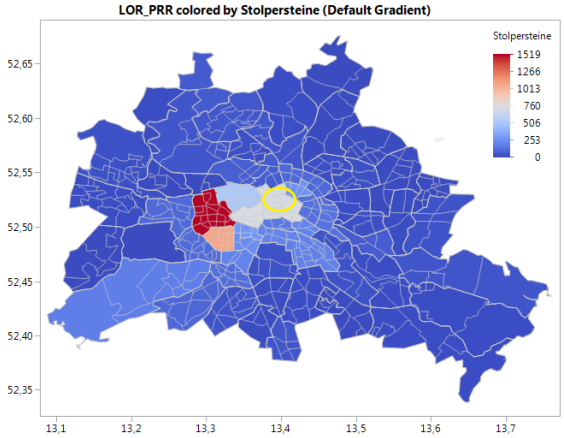
c) Log scaling with the Cool to Warm theme



d) Stolpersteine exact geocoordinates



e) Cool to Warm gradient on Bezirksregion layer



f) Cool to Warm gradient on Prognosenraum layer

Figure 4.5. Comparison between gradient, scaling and layers settings depicting the same data. Figs. a) to c) illustrate the themes and scaling effects. Figs. a), e) and f) illustrate how certain areas are affected by the boundary choice.

with 267 Stolpersteine and, therefore, depicted with the darkest shade of red. In e), using the *Bezirksregion* layer, the marked cell is aggregated with another five cells from the previous layer and together they sum to 533 Stolpersteine, which is the third most dense cell among the 138 polygons that constitute this layer and coloured accordingly. Finally, in Subfig. f), the marked cell is aggregated with 16 neighbouring PLR-cells (or three BZR-cells) forming this cell in the least granular layer, the *Prognosenraum*. These aggregated cells contain 728 Stolpersteine which is roughly half of the most dense cell in this layer and, therefore, drawn in the corresponding shade of grey.

This illustrates the effect due to the implicit assumption that the areas represented each polygon contains homogeneous data and it exemplifies how the boundaries definition alone can change the overall results and interpretation of a choropleth implementation. Further side effects based only on the boundary choices refers to the fact that the polygons rarely have roughly equal areas and this might shift the observers attention to the bigger cells.

It is important to note, however, that the used data does not fit well with a choropleth mapping implementation. In general, this procedure is better indicated when the boundaries are intrinsically related to the object of study, such as voting results over electoral districts, or when the data is previously normalised, such as in exploring unemployment rate differing across cells. When counting absolute numbers and when these values diverge significantly, it is difficult to circumvent such scaling and visualisation issues without creating distortions in some other sense.

5. Results Comparison

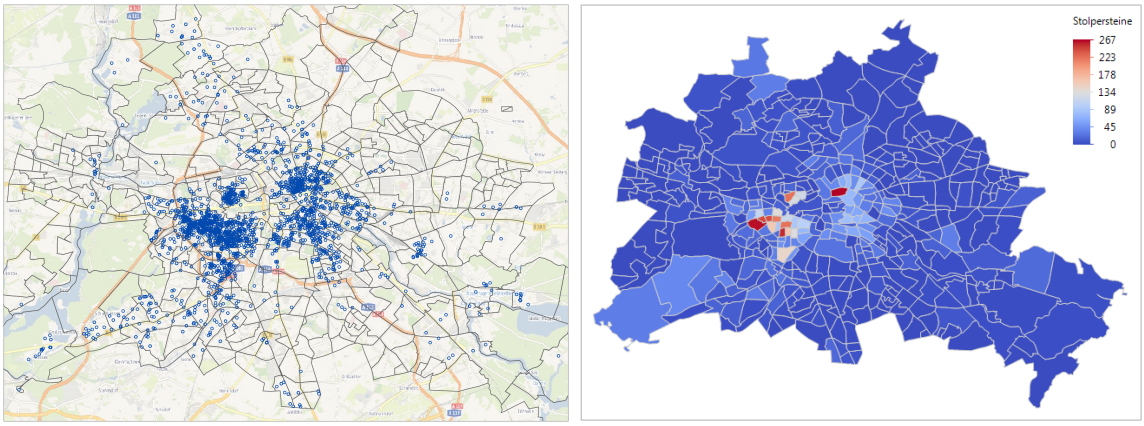
The kernel density estimation discussed in Chapter 3 allows the user to gain great insight about the underlying data distribution. In Chapter 4, an alternative method for density estimation via choropleth mapping for cases where the exact geocoded data is not available was presented. In this chapter I briefly compare the achieved results and discuss an alternative to the choropleth map under a scenario where exact geocoordinates are not available.

For the implementation, a kernel density estimation is conducted based on the centroids of each LOR-cell, weighted by the amount of Stolpersteine inside each polygon. To calculate the centroids, the `coordinates()` function from the R package *sp* was used. The R script generates a csv file that can be imported into JMP. The resulting data table contains four columns. One with the relevant layer index, two with the longitude and latitude values and finally one with the amount of Stolpersteine in each boundary. The used script can be found in under Appendix A.4.

The results displayed in Fig. 5.1 depict in a) the exact location of the Stolpersteine, in b) the choropleth results as implemented in Chapter 4 and in c) the kernel density estimation based on the exact geocoded location as applied in Section 3.2.1. Note that in figure c) the exact data from low density areas are kept visible for easier assessment on the resulting contour lines. The results in Fig. 5.1 d) illustrate a scenario where the exact location is not available and a kernel density estimation is conducted based on centroids of the Plannungsraum (PLR) layer, which is the most granular with 447 cells. Each centroid is marked with an $\times$ in this and subsequent figures. A visual comparison of figures c) and d) reveals that a procedure conducted over aggregated data can also produce acceptable estimation results. Clearly, some degree of precision goes lost due to the data aggregation, but still rich information on the main aspects of the underlying data, such as areas of highest density and their variation across the city, can be inferred reasonably well from the estimation results shown in figure d). The main deviations in this output when comparing to the original data reveals to be on bigger cells, where the centroids can shift the estimation output considerably. This, in turn, shows the potential of plotting the estimation of a background map. The user can easily check if the results correspond with the underlying geography and, when necessary, mentally

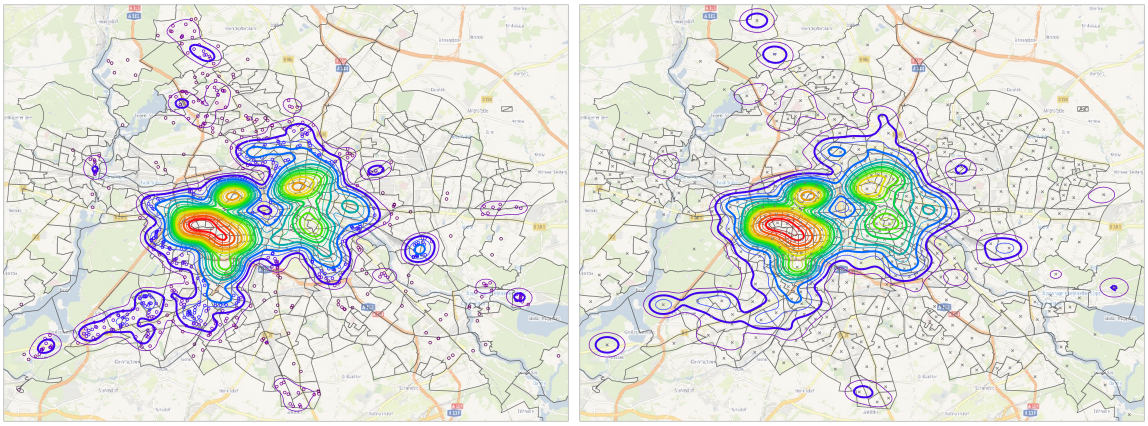
shift the spurious contours to a more coherent area. Note that for both implementations the bandwidth parameters were set as 0.0108 and 0.0051 for longitude and latitude, respectively.

As the boundaries layer get less granular, as it is the case when using the Bezirksregion (BZR) layer containing 138 cells, the estimation results suffers considerably. The results, as depicted in Fig. 5.1 e), are less precise and display signs of spurious spikes. Nonetheless, key aspects of the data can still be retrieved if the intended application does not require a very high degree of precision and a higher smooth parameter can be applied in order to alleviate some of the spurious spikes, as shown in Fig. 5.1 f). In this case the bandwidth parameters were raised to 0.0180 and 0.0084 for longitude and latitude, respectively.



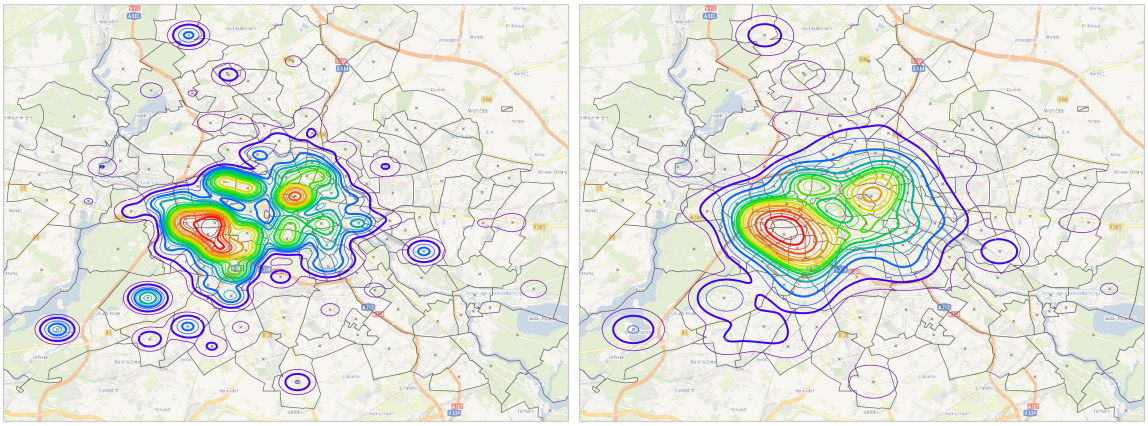
a) Exact Stolpersteine location

b) Choropleth based on PLR layer



c) KDE based on exact geocoordinates

d) KDE based on PLR centroids



e) KDE based on BZR centroids

f) KDE based on BZR centroids (higher width)

Figure 5.1. Comparison between Choropleth and KDE results including when exact geocoded data is not available illustrates the case where the KDE output is acceptable if boundaries are granular enough.

6. Conclusion and Outlook

Exploratory analysis is a important starting point when searching for patterns in a dataset. JMP reveals to be an powerful tool for exploratory spatial data analysis with its intuitive and interactive interface. When working with exact geo-coded data, the non-parametric estimation within the *Fit Y by X* context offers the user access to a flexible estimation and exploration environment. The ability to adjust the bandwidth parameters separately for the X and Y variable reveals very useful, specially when dealing with data in the WGS 84 coordinate system with its varying longitudinal and latitudinal scaling. Furthermore, the ability to integrate a street level interactive map stimulates quick and ad hoc verification of prior beliefs or unexpected results. The resulting outputs are also visually pleasing and can be directly exported for presentational purposes without the necessity of extensive adjustments. The *Graph Builder* context presents a user-friendlier interface for graph creation, but the available density estimation procedure in this context is less flexible.

When dealing with spatial data aggregated over predefined areas, a choropleth mapping procedure is often used to portray the density and distribution of the data. Within JMP, the user can import and work with customized shapefiles. The creation process is moderately complicated, but the implementation procedure is rather simple after the boundaries are set up. Although it is expected that some information is lost when data is aggregated, the high sensitivity to scaling and visualisation adjustments indicates that the achieved results might not only be less informative but also misleading. Furthermore, the implicit assumption that the data in each cell is homogeneous might not hold depending on the boundaries characteristics and granularity. To circumvent this issue, I briefly examined the results achievable by conducting a kernel density estimation based on the LOR centroids, using the Plannungsraum (PLR) and the Bezirksregion (BZR) layers. Based on the PLR layer, the most granular with 447 cells, the achieved output reveals to be very similar to the estimation based on exact geocoded data, with very small loss of precision. The less granular the layer, as it is the case using the BZR layer with 138 cells, the less precise and eventually misleading the output might become. The positive results using the PLR layer might also have been slightly overstated, due to the fact that the Stolpersteine data tends to concentrate on central areas, where the LOR cells are smaller.

This, in turn, tends to generate smaller biases when compared to bigger areas, usually located distant from the centre. These preliminary results, however, could be extended with different datasets and under more robust statistical tests in order to find adequate boundary definitions that increases estimation precision and minimizes potential bias, while observing privacy and confidentiality concerns.

Appendix A.

Appendix

A.1. WMS server implementation and comparison

For the results shown in Fig. 2.3 under Section 2.3 the following URL and layer settings were used:

(a) Aerial Image from 2015 (FIS-Broker)
URL: `http://fbinter.stadt-berlin.de/fb/wms/senstadt/k_luftbild2015_rgb`
Layer: 0

(b) Colour OpenStreetMap (terrestris)
URL: `http://ows.terrestris.de/osm/service`
Layer: OSM-WMS

(c) Aerial Image from 1928 (FIS-Broker)
URL: `http://fbinter.stadt-berlin.de/fb/wms/senstadt/k_luftbild1928`
Layer: 0

(d) Grayscale OpenstreetMap (terrestris)
URL: `http://ows.terrestris.de/osm-gray/service`
Layer: OSM-WMS

(e) Berlin Map from 1940 (FIS-Broker)
URL: `http://fbinter.stadt-berlin.de/fb/wms/senstadt/berlin1940`
Layer: 0
Notes: This map covers only relatively small area over the centre of Berlin

(f) Built-in OpenStreetMaps (JMP)
URL and layer not required.

Note also that, in order to better emphasise the data points, the marker colour was changed appropriately and the transparency level set to 0.7 on map images (Figs. b), d), e) and f)). Such changes can be done via the *Customize...* window [on Graph frame: Right-click > Customize...].

A.2. R scripts geocoding Stolpersteine addresses

For this code implementation I divided the process into three parts. The first part deals with the preparation of the data, including making sure there are not repeated addresses with slightly different names, such as "ss" and "ß" or even lower and upper case. Additionally, given that multiple observations refer to the exact same address, a column with a unique identifier for each address was added, so that each observation needed to be geocoded only once. In the end, a smaller csv file containing only the unique address and their index number is saved for the geocoding process in the second step.

stolpersteine01Preparation.R

```
# file format: UTF-8

# read csv file
df <- read.csv("./input/stolper_07.09.17.csv",
               header = T, sep = ",", encoding = "UTF-8",
               strip.white = TRUE, na.strings = "",
               stringsAsFactors=FALSE)

# rename column 3 from "District." to "District"
# and add missing distric name
colnames(df)[3] = "District"
df$District[is.na(df$District)] <- "Schmargendorf"

# make new column with address and ortsteil
df$address.full <- paste(df$Address, df$District,
                         "Berlin, Deutschland", sep=" - ")

# change all to lower case and umlauts to latin characters.
df$address.full <- tolower(df$address.full)
df$address.full <- sub("str\\.", "strasse", df$address.full)
df$address.full <- sub("pl\\.", "platz", df$address.full)
df$address.full <- sub("ü", "ue", df$address.full)
df$address.full <- sub("ö", "oe", df$address.full)
df$address.full <- sub("ä", "ae", df$address.full)
df$address.full <- sub("é", "e", df$address.full)
df$address.full <- sub("è", "e", df$address.full)
df$address.full <- sub("ß", "ss", df$address.full)

# sort by address
df <- df[order(df$address.full), ]
```

```
31
32 # Add unique index
33 df <- transform(df, ClusterID = as.numeric(
34                 interaction(df$address.full, drop=TRUE)))
35
36 max(df$ClusterID) # = 2843 unique addresses
37 # write ongoing output into csv file
38 write.csv(df, file = "./input/dfCleaned.csv",
39           fileEncoding = "UTF-8", row.names = FALSE)
40
41 # make a small data frame with ID and full address only
42 dfSmall <- unique(data.frame("ClusterID" = df$ClusterID,
43                              "Address" =  df$address.full))
44
45 # save preparation data into csv file
46 write.csv(dfSmall, file="./input/dfSmall.csv", fileEncoding="UTF-8")
```

In the second step is where the geocoding process takes place. The following lines of code are a little more complex and surely not the most sophisticated[1], but it accomplishes the intended results. Some complexity is added, due to the fact that Google's API user conditions sets a daily query limit at 2500. With that in mind, I created a way of saving the ongoing results for retrieval on the following day or if there should be an internet connection problem.

stolpersteine02Geocode.R

```
1  library(ggmap)
2
3  # set location for ongoing-results file
4  ongoing_results_file <- "./output/ongoing_results.rds"
5
6  # open ongoing if exists, otherwise open dfSmall from preparation
7  if (file.exists(ongoing_results_file)) {
8      dfSmall <- readRDS(ongoing_results_file)
9      startIndex <- nrow(dfSmall) - sum(is.na(dfSmall$lon))
10     print("using ongoing file as dfSmall")
11 } else {
12     dfSmall <- read.table("./input/dfSmall.csv",
13     header=T, sep=",", encoding="UTF-8",
14     colClasses=c('NULL',NA,NA), stringsAsFactors=FALSE)
```

[1] A better script for a geocoding a large list of addressees, which was helpful when creating my own, can be found on: `https://www.shanelynn.ie/massive-geocoding-with-r-and-google-maps`, Accessed: 03.10.2017.

```
15
16      # create new column for lon, lat and address to be later populated
17      dfSmall$lon <- NA
18      dfSmall$lat <- NA
19      dfSmall$address <- NA
20      startIndex <- 1
21      print("using dfSmall from preparation step (1)")
22  }
23
24  # function to go through the rows and geocode each address
25  for(i in seq(startIndex, nrow(dfSmall))) {
26      id <- dfSmall$ClusterID[i]
27
28      if (is.na(dfSmall$lon[i])) {
29          #checks if row is a "missing value"
30          address <- dfSmall$Address[i]
31
32          # call geocode function and apply lat, lon to cluster ID rows
33          dfAPIResponse <- geocode(address, output="latlona",
                override_limit=T)
34          dfAPIResponse$ClusterID[1] <- id
35              dfAPIResponse$FullAddress <- dfSmall$Address[i]
36          dfSmall$lat[dfSmall$ClusterID == id] <-
37              dfAPIResponse$lat[dfAPIResponse$ClusterID == id]
38          dfSmall$lon[dfSmall$ClusterID == id] <-
39              dfAPIResponse$lon[dfAPIResponse$ClusterID == id]
40
41          # when retrieval is not successfull the column "address"
42          # is not filled, therefore the next if-else statement
43          if( "address" %in% names(dfAPIResponse) ) {
44          dfSmall$address[dfSmall$ClusterID == id] <-
45              dfAPIResponse$address[dfAPIResponse$ClusterID == id]
46          } else {
47              dfSmall$address[dfSmall$ClusterID == id] <- NA
48          }
49
50          # save ongoing results every 10th query
51          if (i %% 10 == 0 | i == nrow(dfSmall)) {
52          saveRDS(dfSmall, file = ongoing_results_file)
53          }
54      } else {
55      print(paste("id nr", id, "already geocoded"))
```

```
56      }
57  }
58  # logging in case of errors
59  w <- warnings()
60  sink("./log/warnings.txt"); print(w); sink()
61
62  # write final results into csv file
63  write.table(dfSmall, file = "./output/dfResults.csv", sep = ",",
64              fileEncoding = "UTF-8",row.names = FALSE)
```

Finally, the geocoded addresses can be merged back into the main data frame in the third step. The csv file created is then used for further analysis within JMP.

stolpersteine03Merge.R

```
1  # read results data
2  dfResults <- read.table("./output/dfResults.csv", header=T, sep=",",
3                          encoding="UTF-8", stringsAsFactors=FALSE)
4
5  ## read full input data (already cleaned)
6  df <- read.table("./input/dfCleaned.csv", header = T, sep = ",",
7                   encoding = "UTF-8", na.strings = "",
8                   stringsAsFactors=FALSE)
9
10 # join both data.frames by matching "ClusterID"
11 dfTotal <- join(df, dfResults, type = "full", by = "ClusterID")
12
13 # after checking that the address matches, drop duplicate columns
14 list_keep <- c("Name", "Address", "District",
15                "Year", "ClusterID", "lon", "lat")
16 dfTotal <- dfTotal[ , (names(dfTotal) %in% list_keep)]
17
18 # rename some columns for better consistency among oder headers
19 names(dfTotal)[5:6] <- c("Longitude", "Latitude")
20
21 # write final results into csv file for analysis in JMP
22 write.table(dfTotal, file = "./output/stolpersteine.csv",
23             fileEncoding = "UTF-8", sep = ",", row.names = FALSE)
```

A.3. R scripts for shapefile coordinate system conversion

shapefileConversion.R

```
# set layer to be converted by changing the index nr.
layer <- c("Planungsraum", "Bezirksregion", "Prognoseraum")[1]

# libraries and working directory
library(rgdal)

# CRS for input and output formats
epsg_UTM33 = "+init=epsg:25833" # change input CRS accordingly
epsg_output = "+init=epsg:4326" # change output CRS accordingly

# read ESRI shapefile
dsnPath <- "./inputDATA/LOR_BERLIN_ALL/LOR_SHP_EPSG_25833_UTM33/."
Berlinshape <- readOGR(dsn = dsnPath,
layer = paste(layer, "EPSG_25833", sep="_"))

# assign crs (proj4strig)
proj4string(Berlinshape) <- CRS(epsg_UTM33)

# transform CRS into lat/lon format
Berlinshape <- spTransform(Berlinshape, CRS(epsg_output))

#write new shapefile on WGS84 (lat lon)
writeOGR(obj = Berlinshape,
         dsn = "./outputDATA/WGS84/from_SHP_UTM33",
         layer = paste(layer, "EPSG_25833_from_UTM33", sep="_"),
         driver = "ESRI Shapefile")

# Useful resources: ref: https://stackoverflow.com/questions/36520915/
# and https://gis.stackexchange.com/questions/166876/
```

choroplethCount.R

```
# import libraries
library("sp")
library("rgdal")

# read ESRI shapefile downloaded from Berlin OPENDATA portal
BerlinShape <- readOGR(dsn = "./outputDATA/WGS84/from_SHP_UTM33/.",
                       layer = "Planungsraum_EPSG_25833_from_UTM33")
```

```
8
9  # read stolpersteine csv as data.frame
10 stolpersteine <- read.table("./stolpersteine/output/stolpersteine.csv",
11                            header=T, sep=",", encoding="UTF-8",
12                            stringsAsFactors=FALSE)
13
14 # drop NA's from data.frame
15 stolpersteine <- stolpersteine[!is.na(stolpersteine$Latitude), ]
16
17 # set Longintude and Latitude columns as coordinates, this also
18 #turns the  data.frame into a SpatialPolygonsDataFrame object
19 coordinates(stolpersteine) <- ~ Longitude + Latitude
20
21 # check proj4string of each object
22 proj4string(stolpersteine) ;proj4string(BerlinShape)
23
24 # assign WGS84 CS via proj4string
25 proj4string(stolpersteine) <- proj4string(BerlinShape)
26
27 # creates data frame with Stolpersteine contained in Berlinshape
28 df_aggregate <- over(stolpersteine, BerlinShape)
29
30 # aggregate results and turn into data.frame class
31 df_aggregate <- data.frame(table(df_aggregate$SCHLUESSEL))
32
33 # rename columns
34 names(df_aggregate) <- c("LOR_PLR", "Stolpersteine")
35
36 # set Bezirks- und Prognosenräume columns (plus district).
37 df_aggregate$LOR_BZR <- substr(df_aggregate$LOR_PLR, 1, 6)
38 df_aggregate$LOR_PRR <- substr(df_aggregate$LOR_PLR, 1, 4)
39 df_aggregate$District <- substr(df_aggregate$LOR_PLR, 1, 2)
40
41 # reorder columns
42 df_aggregate <- df_aggregate[c(5,4,3,1,2)]
43
44 # write into csv
45 write.table(df_aggregate,
46             file = "./outputDATA/stolpersteine/stolper_choropleth.csv",
47             fileEncoding = "UTF-8", sep = ",", row.names = FALSE)
48
49 # helpful resource: # https://gis.stackexchange.com/questions/110117
```

A.4. R script for retrieving centroid from LOR layers

The following script was used to retrieve the centroids from each LOR layer and aggregating the number of Stolpersteine inside each cell. This example goes over the PLR layer, but the same procedure can be done by adjusting the layer naming accordingly.

stolperLORCentroids.R

```
# import libraries
library(rgdal)
library(sp)

# read LOR_PLR from WGS84 shapefile
BerlinShapePLR <- readOGR(dsn = "./outputDATA/WGS84/from_SHP_UTM33/.",
                          layer = "Planungsraum_EPSG_25833_from_UTM33")

# get centroid coordinates and IDs of shapefile
centroidsPLR <- as.data.frame(coordinates(BerlinShapePLR))
LOR_PLR <- as.data.frame(BerlinShapePLR$SCHLUESSEL)

# create a data.frame with IDS and centroids and change column names
berlinPLRCentr <- cbind(LOR_PLR, centroidsPLR)
names(berlinPLRCentr) <- c("LOR_PLR", "Longitude", "Latitude")

# read stolpersteine csv file
fileName <- "./outputDATA/stolpersteine/stolper_choropleth.csv"
berlinStolper <- read.csv(file = fileName,
colClasses = c(rep("character", 4), "numeric"))

# aggregate (sum) over Stolpersteine by common LOR_PLR
berlinAggPLR <- aggregate(berlinStolper$Stolpersteine,
                          by = list(berlinStolper$LOR_PLR),
                          FUN = sum)

# apply name (important LOR_PLR matches for merge)
names(berlinAggPLR) <- c("LOR_PLR", "Stolpersteine")

# merge and write to file as csv
berlinAggPLR <- base::merge(berlinPLRCentr, berlinAggPLR)
write.table(berlinAggPLR,
            file = "./outputDATA/stolpersteine/berlinStolpLORPLR.csv",
            fileEncoding = "UTF-8", sep = ",", row.names = FALSE)
```

Bibliography

Beaujardiere, J. de la, ed. (2006). *OpenGIS Web Map Service Implementation Specification, Version 1.3.0*. Open Geospatial Consortium Inc., p. 85. URL: http://www.opengeospatial.org/standards/wms.

Bivand, R., T. Keitt and B. Rowlingson (2017). *rgdal: Bindings for the Geospatial Data Abstraction Library*. R package version 1.2-8. URL: https://CRAN.R-project.org/package=rgdal.

Daintith, J. and E. Wright (2008). *A Dictionary of Computing*. Delaunay triangulation. Oxford University Press. ISBN: 9780199234004.

Demnig, G. (n.d.). *Stolpersteine*. Stolpersteine Europe. URL: http://www.stolpersteine.eu/en/ (visited on 26/09/2017).

ESRI (1998). 'Shapefile technical description'. In: *An ESRI White Paper*.

Fan, J. and J. Marron (1994). 'Fast Implementations of Nonparametric Curve Estimators'. In: *Journal of Computational and Graphical Statistics* 3.1, pp. 35–56. ISSN: 1061-8600. DOI: 10.1080/10618600.1994.10474629.

Kahle, D. and H. Wickham (2013). 'ggmap: Spatial Visualization with ggplot2'. In: *The R Journal* 5.1, pp. 144–161. URL: http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf.

OpenStreetMap contributors (2017). *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org.

Pantelis Rodis (2015). *WMS Map Viewer on line*. URL: https://wms-viewer-online.appspot.com/ (visited on 07/10/2017).

Pebesma, E. and R. Bivand (2005). *Classes and methods for spatial data in R*. R package version used: 1.2.5. R News 5 (2). URL: https://cran.r-project.org/doc/Rnews/.

SAS Institute (2017a). *JMP 13 Essential Graphing*. Version 13.1. Cary, NC: SAS Institute. 352 pp. ISBN: 978-1-62960-474-9.

– (2017b). *JMP 13 Quality and Process Methods*. Version 13.1. Cary, NC: SAS Institute. 314 pp. ISBN: 978-1-62960-480-0.

Scott, D. W. (2015). *Multivariate density estimation: theory, practice, and visualization*. Second edition. Hoboken, New Jersey: John Wiley & Sons, Inc.

Senate Department for Urban Development and Housing (n.d.[a]). *Karten, Daten, Dienste - FIS-Broker*. URL: `http://www.stadtentwicklung.berlin.de/geoinformation/fis-broker/` (visited on 07/10/2017).

– (n.d.[b]). *Lebensweltlich orientierte Räume (LOR)*. URL: `http://www.stadtentwicklung.berlin.de/planen/basisdaten_stadtentwicklung/lor/index.shtml` (visited on 08/10/2017).

Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis.

Stolpersteine Art Project (n.d.). *Stolpersteine in Berlin*. URL: `https://www.stolpersteine-berlin.de/en/project/artproject` (visited on 26/09/2017).

terrestris (n.d.). *OWS terrestris Dokumentation*. URL: `https://ows.terrestris.de/dienste.html` (visited on 07/10/2017).

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Berlin, November 24, 2017

.....

(Signature of the author)