Stockholms universitet

Statistical generalization and applications of a robust, fast and fully automated density based clustering method for big data

Anton Holm

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2022:3
Matematisk statistik
Juni 2022

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Mathematical Statistics
Stockholm University
Master Thesis **2022:3**
http://www.math.su.se

Statistical generalization and applications of a robust, fast and fully automated density based clustering method for big data

Anton Holm*

June 2022

Abstract

In recent years, machine learning has taken a larger place in data analysis. The aim is often to predict some response variables based on other explanatory variables. When only the explanatory variables are available, it is still possible to do inference on the data. One way to do so is by performing clustering. Essentially, clustering is a method where data points with similar characteristics are grouped while keeping data points with dissimilar characteristics far from each other. There are currently several different branches of clustering and this thesis will be focusing on density-based clustering. A comparison between a kernel density estimator and a k-nearest neighbor (kNN) density estimator is performed, showing the strength and robustness of using a kNN approach. The main goal of this thesis is the construction of a fully automated density-based clustering method. The method is statistically robust to clusters with varying shape and density, works fast on large data sets, is easy to understand and interpret even for non-statisticians, and only relies on a single parameter. The method is tested on a generated data set showing promising results. Lastly, future improvements are discussed, suggesting the use of fuzzy clustering and substitution of Euclidean distance by graph-based distance in efficiently identifying clusters with non-linear shape.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: anton.holm.klang@gmail.com. Supervisor: Chun-Biu Li.

Contents

1 Introduction

Machine Learning has seen a rapid rise in analyzing data across several fields over the last couple of decades. The applicability is widespread and new methods are developed continuously. One segment of Machine Learning used in almost every field is clustering. Clustering is the procedure of grouping similar data points together while keeping groups of dissimilar points far from each other. Often the similarity is based on some features or spatial location of the data. There exist many different branches of clustering such as hierarchical clustering, graph-based clustering, and partitional clustering. This thesis will however discuss another branch called density-based clustering.

One of the first approaches to density-based clustering was introduced by Wishert (1969) as a method for removing chaining effects in hierarchical clustering. In short, the number of neighbors within a given distance of a point was used in order to remove noise in the data. The first clustering method focused on density was however introduced by Hartigan (1975). Hartigan proposed to link all points within some distance threshold r to each other. A cluster would then be classified as a contour around a set of fully connected points with large enough densities $\rho(x)$. Most often, different density-based clustering algorithms differ within three parts. How is the density $\rho(x)$ calculated, how is connectivity between points defined, and how does the algorithm for finding connected components work?

At the Science for Life Laboratory, there exists an ongoing project trying to segment tissues into domains based on cell-type compositions. Most previous works within cell inference have previously been done using single-cell sequencing, i.e. looking at the order of the DNA base for individual cells to distinguish between different cell types. More recent work is trying to perform the same type of inference on *in-situ* samples e.g. Partel and Wahlby (2021), Petukhov *et al.* (2021), and Park *et al.* (2021). When using *in-situ* samples, the cells are kept intact within their natural habitat, e.g. within intact tissue, in contrast to single-cell sequencing where the cell has been extracted to be examined. By examining some tissue from a biopsy, the placement of different genes in the tissue can be extracted resulting in a transcriptomic data set. However, in doing so, the knowledge of which cell each gene belongs to is lost. Therefore, this thesis aims to develop an intuitive, simple, and automatized clustering algorithm for application to transcriptomics data. As will be seen throughout this thesis, due to the simplicity and robustness of the method on large data sets, the application of this clustering algorithm is not confined to genomics.

The outline of this thesis is as follows. Section 2 will discuss different density estimators. In Section 3, the density clustering algorithm will be introduced. Section 4 will combine the previous sections to construct the entire algorithm specific to large data sets. Section 5 will compare the different density estimators and evaluate the clustering algorithm. Finally, Section 6 will bring forth some conclusions and future improvements.

2 Density Estimation

One of the fundamental parts in statistics is the use of a probability density function $f_X(x)$ for the random variable X with the property $P(a \leq X \leq b) = \int_a^b f_X(x)dx$ for the univariate continuous case. Real-world data rarely follows some theoretical density distribution exactly and in most cases, it is impossible to find this distribution. In these cases, would the density of the data be required, an estimation is necessary.

When some information concerning the underlying distribution of a sample is available it is possible to perform a parametric approach to estimate the density. As an example, assume there exists some sample where the family of distribution is known, e.g. an exponential distribution is a reasonable assumption. This assumption could for example emanate from physics or nature. Under such assumptions, it is possible to estimate the parameters of the underlying distribution using the sample. In this thesis, the focus will be on a non-parametric approach, where no assumptions of the underlying distribution are required.

2.1 Kernel Density Estimation

One commonly used non-parametric method for density estimation is based on using kernels. As mentioned by Shether (2004), Kernel Density Estimation (KDE) has widespread use across several fields. The use of KDE is further prominent in Park *et al.*, (2021) and appears in various research within genetics. The KDE for a sample $x_1, x_2, ..., x_n$ at some point x is defined as

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}),$$

where K is a kernel satisfying $\int K(y)dy = 1$ and h is the bandwidth of the Kernel. One common choice of kernel is a Gaussian kernel where $K(y) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{y^2}{2})$. In practice, the estimated density of a point is decided by the number of other points (neighbors) within the bandwidth h as a function of the kernel. Often the further away points lie from the original point, the less impact it has on the estimated density as can be seen in Figure 1.

The KDE is most commonly used in order to get a smooth estimation of the density. It is also advantageous when the density of the data should be modeled in a specific way. As an example, consider the Gaussian kernel as described previously. In this case, only distance between points is taken into consideration when estimating the density. The direction should not affect the estimation at all. Depending on the goal of the estimation, a suitable kernel should be implemented.

Something that could be even more important than a suitable choice of the Kernel is the choice of bandwidth. The size of the bandwidth has a great influence on the outcome of the estimator. If the choice of bandwidth is too large, the estimation will be over-smoothed, possibly removing some important

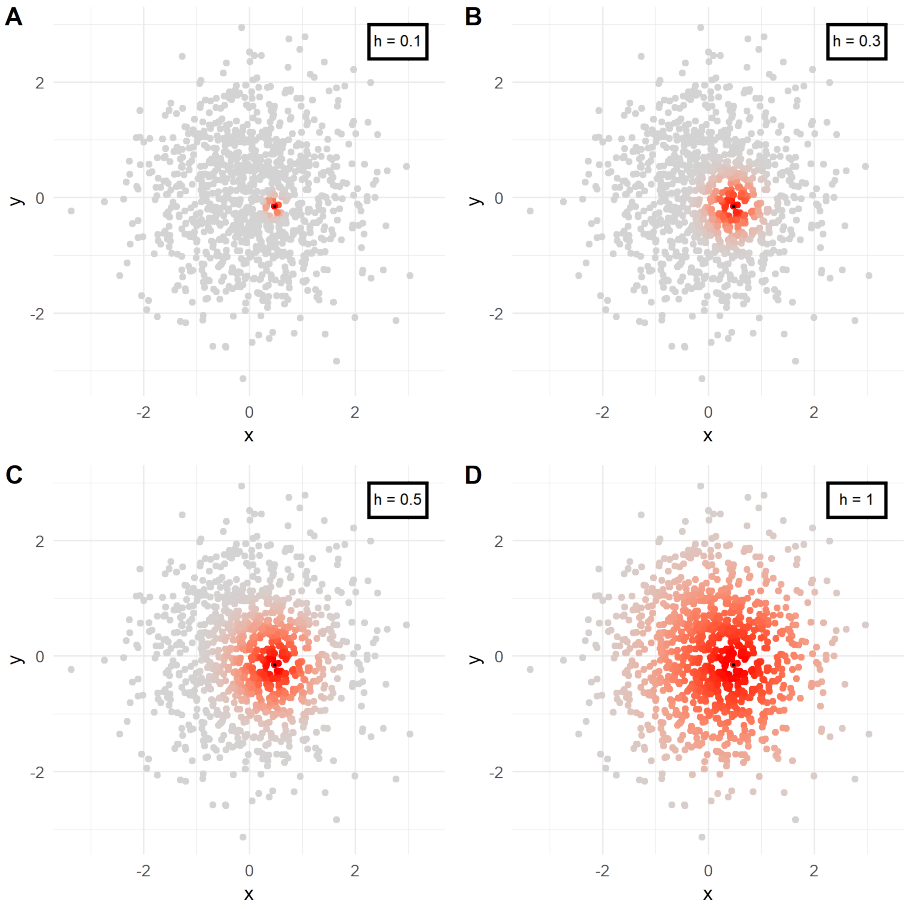

Figure 1: An illustration of using a Gaussian kernel to estimate the density of a 2-dimensional standard Gaussian sample of 1000 observations. Here h is the bandwidth of the Kernel. The image shows the impact each point has in estimating the density of the black point based on the size of the bandwidth. If a neighbor is close to the black point, it has a larger contribution shown by a more fragrant red color. The further away a neighbor lies to the black point, the less impact it has shown by a less fragrant red color until it is completely absent for neighbors far away. The larger the bandwidth is, the more points are taken into consideration.

characteristics of the data. On the other side, choosing a bandwidth that is too small could lead to under-smoothing, implementing fabricated characteristics upon the data. One choice of the bandwidth h as suggested by Sheather (2004) would be to minimize the Asymptotic Mean Integrated Squared Error (AMISE). The details and derivations of the AMISE can be found in the Appendix. In practice, the aim is to choose a bandwidth h which minimizes the bias and variance of the estimator. An example of over-and under-smoothing can be seen in Figure 2

A further complication in choosing the size of the bandwidth occurs when the homogeneity of a data set comes into question. When the density of points varies a lot, there exists no one-size-fits-all solution in deciding the bandwidth. When this is the case, an adaptive bandwidth needs to be implemented.

2.2 k-Nearest Neighbor Density Estimation

Loftsgaarden and Quesenberry (1965) laid the groundwork for a k-nearest neighbor (kNN) approach to estimating the density function. This is a non-parametric approach where the neighbors of a data point are used in estimating the density. The fact that this method is non-parametric means there is no need to decide the bandwidth. Instead, the number of neighbors k is predetermined. The major benefit of a kNN approach is the adaptive bandwidth. When regions become more sparse, neighbors are further away and the bandwidth becomes larger. In more dense areas, neighbors are closer meaning the bandwidth is smaller. There are many variations for kNN density estimation, and some of the more common methods will be explained in this section.

2.2.1 Preliminary Estimator

The most common method of kNN density estimation derives from Kraskov *et al.*, (2004) with over 3000 citations. The paper focuses on estimating the mutual information but in doing so, introduces an intuitive and simple density estimator using kNN.

Let X be some continuous random variable within some space with distance function $d_{ij} = ||x_i - x_j||$ between all pairs of observations of X. Assume there exist some density function $\mu(x)$ to be estimated. Denote ϵ_i as two times the distance from point x_i to its k-nearest neighbor using Euclidean distance. Let $P_i(\epsilon)$ be the probability mass function of the ϵ-ball around x_i where

$$P_i(\epsilon) = \int_{||x_i - \epsilon|| < \epsilon/2} \mu(\xi) d\xi.$$

Let $p_k(\epsilon)$ be the probability distribution for the distance between a point and its k-nearest neighbor. In other words, $p_k(\epsilon)d\epsilon$ is the probability of having $k-1$ points at smaller distance than $\epsilon/2$ from x_i, one point within distance $r \in [\epsilon/2, \epsilon/2 + d\epsilon/2]$ and $N-k-1$ points further away. From the multinomial distribution it is derived that the probability of the k-nearest neighbor having $\epsilon/2$ distance to a point x_i is

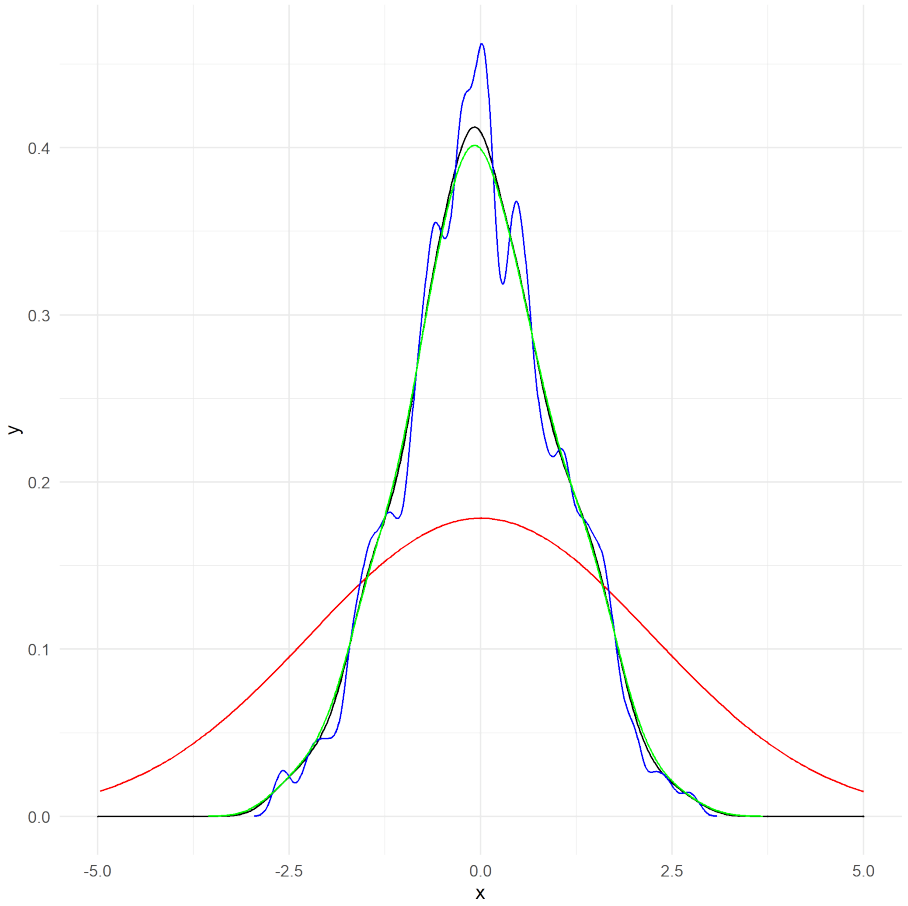

Figure 2: An example of KDE on a sample of 500 observations from a standard normal distribution (Black line). Here the y-axis represents the density of the distribution. A Gaussian kernel was used with different sizes of bandwidth. When $h = 0.1$ (blue) there is under-smoothing, when $h = 2$ (red) there is over-smoothing, and for $h = 0.3$ (green) the estimation is quite close.

$$p_k(\epsilon)d\epsilon = \frac{(N-1)!}{(k-1)!(N-k-1)!} \cdot \frac{dP_i(\epsilon)}{d\epsilon} d\epsilon \cdot P_i(\epsilon)^{k-1} \cdot (1 - P_i(\epsilon))^{N-k-1}.$$

Under the assumption of having a constant density function $\mu(x)$ within the ϵ-ball, by use of the volume of a sphere, an approximated estimation of $\mu(x)$ is given by

$$P_i(\epsilon) \approx (\epsilon/2)^d \cdot V_d \cdot \mu(x_i),$$

where d is the dimension of X and $V_d = \frac{\pi^{d/2}}{(n/2)!}$ is the volume of the unit sphere in d dimensions. By rearranging terms and taking the expectation an estimation of $\log \mu(x)$ is

$$\log \mu(x_i) \approx \mathbb{E}[\log P_i(\epsilon)] - \log V_d - d(\mathbb{E}[\log \epsilon] - \log 2). \tag{1}$$

From the definition of expectation it follows that

$$\begin{aligned} \mathbb{E}[\log P_i(\epsilon)] &= \int_0^\infty p_k(\epsilon) \log P_i(\epsilon) d\epsilon \\ &= k\binom{N-1}{k} \int_0^1 \omega^{k-1}(1-\omega)^{N-k-1} \log \omega d\omega \\ &= \Psi(k) - \Psi(N), \end{aligned} \tag{2}$$

where $\Psi(\cdot)$ is the digamma function. A more detailed derivation of the last equality can be seen in the Appendix. Combining (1) and (2) yields

$$\log \mu(x_i) \approx \Psi(k) - \Psi(N) - \log V_d - d(\mathbb{E}[\log \epsilon] - \log 2).$$

As k and N are fixed, the only non-constant term of the density estimator is $\mathbb{E}[\log \epsilon]$. In this case, the expectation is only taken over one point, resulting in reducing the term to $\log \epsilon$. Consequently, the rankings from the estimator of $\mu(x)$ only depend on the distance from a point to its k-nearest neighbor. This removes the spatial information around a local neighborhood of a data point. As can be seen in Figure 3, this approach would not be robust and could lead to the same density estimation for two points in areas of different densities. One simple solution would be to use the distance to every nearest neighbor up to neighbor k.

2.2.2 Generalized Density Estimator

In order to utilize the spatial information of all nearest neighbors of a point, two density estimators are proposed motivated by graph theory, and more precisely, similarity graphs. From graph theory, it is known that the degree d_i of a point is defined as

$$d_i = \sum_{j=1}^{n} w_{ij},$$

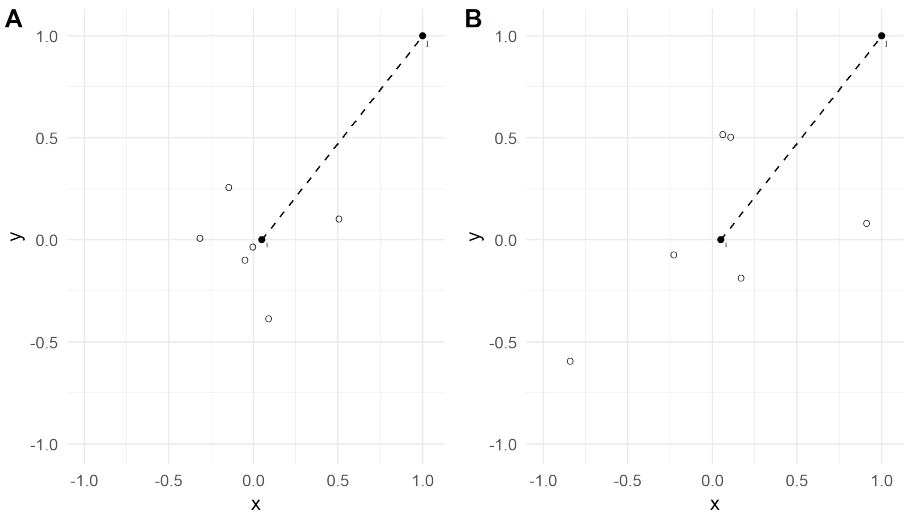

Figure 3: An illustration showing the sensitivity of only considering the distance to the k-nearest neighbor for density estimation. Here $k = 7$ and thus only the distance from point x_i to x_j is considered. The density estimation for **A** and **B** would be the same when clearly the scenario of **A** should yield a larger density estimation for x_i.

where w_{ij} is the weight between two points x_i and x_j. Note that, for a kNN graph, $w_{ij} = 0$ if x_j is not a nearest neighbor of x_i. Following this, one of the proposed density estimators is

$$\rho_i^{(1)} = \frac{1}{\sum_{j=1}^{k} w_{ij}^{(1)}} = \frac{1}{\sum_{j=1}^{k} d_{ij}} = \frac{1}{d_i^{(1)}},$$

where again d_{ij} is the Euclidean distance between point x_i and its j:th nearest neighbor, and k is the fixt number of neighbors to consider.

Another density estimator that is proposed in this thesis is

$$\rho_i^{(2)} = \sum_{j=1}^{k} w_{ij}^{(2)} = \sum_{j=1}^{k} \frac{1}{1 + d_{ij}^2} = d_i^{(2)}.$$

Note that, in the context of this thesis, these are not strictly speaking density estimators and do not signify a true density function as they would for example not sum up to 1. For the goal of this thesis, a density estimator is meant as a statistical quantity to measure the fluctuation of the density of points in a local neighborhood. For the density-based clustering method introduced in Section 3 and Section 4, this is however sufficient.

The two estimators are quite similar with small differences. For $\rho^{(1)}$, the distances to all neighbors are aggregated before the density is estimated. For $\rho^{(2)}$,

the opposite is true. A density estimation based on each neighbor separately is first calculated before being pooled together for a final density estimation. In most cases, the outcome of the estimators is similar, just on different scales. However, there exist some special cases where they differ significantly. One example would be in a very dense area where the distance to neighbors is close to zero meaning the estimation using $\rho^{(1)}$ would go towards infinity. In this case, the scale of the density between different points could be skewed, where the difference in the scale between dense areas and other areas could be too large. This is not as big of a problem when using the second estimator since it is always at most equal to k. On the other hand, say that two outliers lie very close to each other but far away from all other points. Using the second estimator would give these points a density above or close to 1 which could be larger than for points in some sparse areas. This should not be the case and is not a problem for the first estimator. Hence both estimators are considered and an analysis is later performed to compare them.

3 Density Based Clustering

At the start of learning about statistics, a lot of different methods are learned where the goal is to build a model to predict some labels. Some examples could be to, based on some characteristics (predictors), predict the price of a house, make a forecast of the weather or predict how much a customer will spend. What these examples have in common is the fact that they all have one or more labels (response variables) on which the model can be trained on. However, in many real cases, the labels do not exist or are not of interest. An example of this would be when the data only contains the characteristics of houses or customers. In these cases, one of the main analyses that are possible is called clustering.

Clustering is a method where the data is divided into clusters or groups based on their characteristics. Data with similar characteristics should belong to the same cluster and data with dissimilar characteristics should be far apart from each other. An example can be seen in Figure 4 of data over customers for some company. In this example, there are only two characteristics, the yearly salary and the number of previous purchases by the customers, but in most cases, there are many more. The customers are divided into three clusters, consumers with low salaries who rarely make purchases, consumers with high salaries who rarely make purchases, and consumers with high salaries who often make purchases. This could for example be used to know which customers to target with ads or approach in the store. The use of clustering methods is widespread, and therefore, many different methods for clustering have been developed for different areas of use. This thesis will focus on density-based clustering.

Density-based clustering methods have many advantages that suit the goal of this thesis. For example, the number of clusters does not need to be decided in advance which is important when it is impossible to know that information

beforehand. It is also possible to cluster arbitrary shapes as opposed to many algorithms which can only handle spherical clusters. It also deals with large data sets efficiently since no iterations are needed as for many other clustering algorithms. Lastly, the methodology is very intuitive and easy to understand. The most used density-based clustering method is the density-based spatial clustering of applications with noise (DBSCAN) by Ester *et al.* (1996). However, this method, similar to Section 2.1, relies on fixing the size of the bandwidth. Due to this, DBSCAN is not robust to varying density clusters. Hence, another approach based on the algorithm by Rodriguez and Laio (2014) is taken.

3.1 Clustering by fast search and find of density peaks

A more recent method in density-based clustering was established by Rodriguez and Laio (2014). They proposed a density-based clustering algorithm more robust to the choice of threshold and varying cluster densities. The underlying assumptions of the algorithm lie in the idea that cluster centers are characterized by having a larger density than their neighbors and a relatively large distance to other cluster centers.

For each data point x_i, two quantities are calculated, the local density estimation ρ_i and the distance to the closest neighbor with a larger density, denoted as δ_i. To estimate the density, a distance threshold d_c is set giving rise to the quantity

$$\rho_i = \sum_{j=1}^{n} \mathbb{1}\{(d_{ij} - d_c) < 0\},$$

where $\mathbb{1}\{\cdot\}$ is the indicator function being equal to 1 if the input is true, and 0 otherwise. In words, ρ_i is equal to the number of points within a neighborhood of radius d_c around the point x_i as can be seen in Figure 5A.

The second quantity δ_i is simply computed as

$$\delta_i = \min_{j:\rho_j<\rho_i} \{d_{ij}\},$$

where the point with the largest density by convention is given a value as the maximum distance to any point from itself. Under the assumptions of the algorithm, all cluster centers are characterized as having large δ and ρ.

For simpler cases, where the number of data points and clusters are relatively small, a manual approach is taken in finding cluster centers by the use of a decision graph as illustrated in Figure 5B. Any point that sticks out having both large δ and ρ in comparison to all other points are chosen as cluster centers. This can be seen as large gaps between points. When the cluster centers have been found, every other data point is assigned to the same cluster as its nearest neighbor with a larger density, removing any need for iterations.

In many real-world cases, finding the cluster centers in the decision graph could be difficult. When the number of clusters is too large, it is impossible to manually identify them. When that is the case, the authors propose to look at

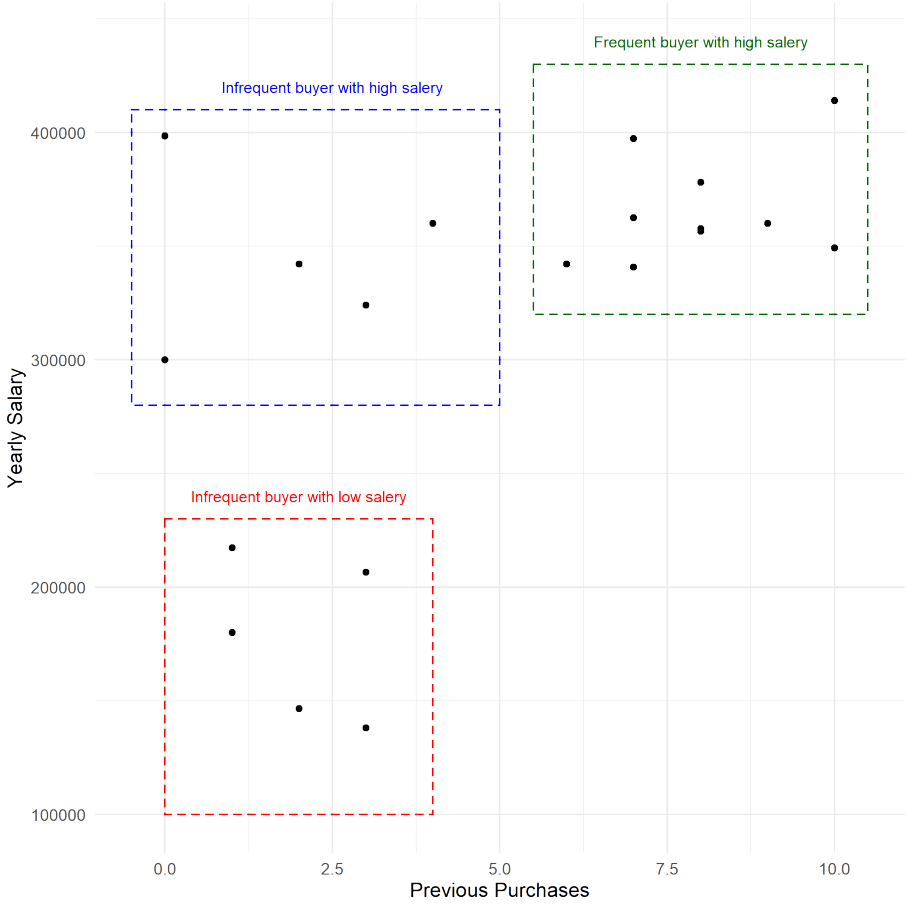

Figure 4: A simple clustering example. The data is of customers containing their yearly salary and number of previous purchases. The customers are divided into three different clusters in order to know which customers are more likely to spend money.

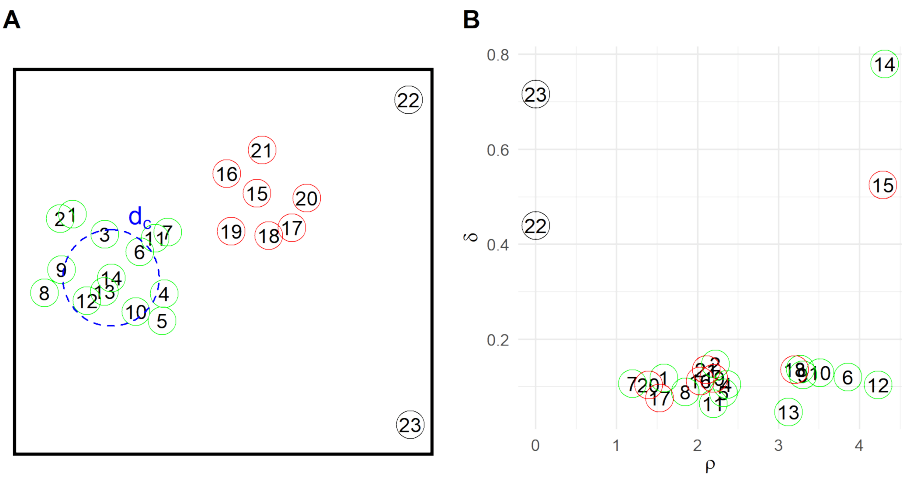

Figure 5: An adaptation of Figure 1 in Rodriguez and Laio (2014). This figure illustrates how cluster centers are found. **A)** Data containing two clusters (green and red) with two outliers (black). The size of d_c is shown with a blue dashed line. **B)** The decision graph where the user manually picks cluster centers. Points 14 and 15 are the only two points with large δ and ρ and as such become cluster centers. Points with large δ and small ρ are likely candidates for being outliers. Note that the points have been slightly shifted in the x-axis for clearer visualization and should in practice only take on integer values.

the statistic $\gamma = \rho\delta$ in descending order instead. However, this could prove to be problematic. One issue could appear when the scale of δ is much larger than ρ. In such scenarios, the density of the points loses its importance, resulting in only focusing on δ in deciding cluster centers. Furthermore, the decision still needs to be done manually, whether it be in picking out cluster centers one by one or setting some threshold of γ. This opens up the possibility for human errors and the number of clusters could be different between users. Since the selection of cluster centers is done manually, it will be very time-consuming whenever the procedure needs to be done a large number of times.

The strong element of the method lies in its speed and simplicity. As the name suggests, the algorithm is quick. In contrast to many other clustering methods, there is no need for iterations or convergence criteria, making it suitable for larger data sets. Additionally, the only parameter in need of being fixt is d_c opposed to most clustering algorithms having at least 2 hyper-parameters to be tuned. On the other hand, the choice of d_c has a great influence on the accuracy of the result and as such, is important to get right. One of the larger complications in deciding d_c appears when the size and density of clusters as well as the distance between clusters vary greatly. In parallel to section 2.1, as for the KDE, there exists no one-size-fits-all size of d_c. As seen in Figure 6, this could induce over-clustering in parts of the data where d_c is too small or under-clustering when d_c is too big.

Lastly, one part of the core of this algorithm lies in calculating δ, the distance from a point to its nearest neighbor with a larger density. Since the density estimator suggested by the authors by its definition only take on integer values, the robustness of the algorithm is questioned. There are possibilities for ties within the density estimator and as such, it is impossible to decide on which point has the larger density. This is especially fragile for sparse data where the density will often be quite small.

In light of these observations, this thesis proposes a generalized version of Rodriguez and Laios (2014) density-based clustering algorithm.

3.2 Generalized Density-Based Clustering

In this section, a generalization of the method in Section 3.1 is presented. The generalization of Rodriguez and Laios (2014) algorithm focuses on preserving the advantages of the method while making the method more robust against its weaknesses. This is done by retaining a fast algorithm with a single hyper-parameter. At the same time, a different density estimator is introduced and the method of finding cluster centers is automatized with the use of monotonic regression and kernel smoothing.

3.2.1 Monotonic Regression

Given some data set with pairwise observations (x_i, y_i) where x_i is the covariate and y_i is the response variable, it is possible to construct models explaining the data. One of the first methods that are taught in statistics is fitting a regression

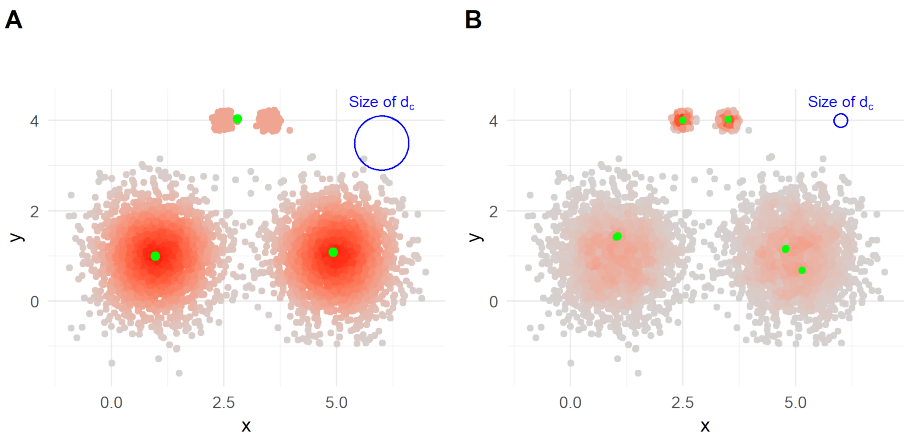

Figure 6: An illustration of one example when the fixt size of d_c in the algorithm by Rodriguez and Laio fails. The data contains two smaller Gaussians with 200 points each and two larger Gaussians with 2000 points each. A stronger red color indicates a larger estimated density. The green points are cluster centers as decided by the algorithm. **A)** In this example, the size of d_c is set based on the size of the larger Gaussians. This results in under-clustering of the smaller Gaussians where only one cluster center (green points) appears in between them. This is the result of too large of a bandwidth, resulting in some of the border points getting contributions from both Gaussians. **B)** In this example, the size of d_c is smaller, in an attempt to avoid the over-clustering in **A**. However, in this case, over-clustering of the larger Gaussians occurs.

line. In order to find a regression line, for two-dimensional data, the following is minimized

$$f(\beta_0, \beta_1) = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 \phi(x_i))^2.$$

Here β_0 is the intercept, β_1 the effect parameter (or slope), and ϕ some basis function. In some cases, for example, when only the rank of x_i is available, this parametric approach does not work. Barlow *et al.* (1972) proposed a non-parametric method, namely Monotonic Regression.

Under the assumption that all observations x_i of X are unique and ordered such that $x_1 < x_2 < ... < x_N$, the monotonic regression line is found by minimizing

$$f(z) = \sum_{i=1}^{N} (y_i - z_i)^2,$$

where it is assumed that $z_1 \leq z_2 \leq ... \leq z_N$. If $\hat{z}$ is the solution minimizing that function, then the best increasing function to fit to the data is the pair of points $(x_i, \hat{z}_i)$ (Leeuw, 2005). When finding the monotonic regression line in practice, it is used that, whenever $y_i > y_{i+1}$, breaking the monotonicity, $\hat{z}_i = \hat{z}_{i+1}$. Under these circumstances, the algorithm for finding the regression line is quite simple.

The procedure for finding the monotonic regression line can be seen in Figure 7. Whenever $y_i > y_{i+1}$, breaks monotonicity, their values are replaced by their combined average. This procedure is continued until the monotonicity is no longer broken. This procedure must come to an end where the most simple regression line would be a straight horizontal line which is clearly monotonic.

3.2.2 Nadaraya–Watson Kernel Regression

In many instances, it is difficult to calculate statistics and do inference on functions that are not smooth. Hence, there exist plenty of methods to make a function smooth whilst maintaining the important statistics and information it contains. Both Nadaraya (1964) and Watson (1964) proposed such a method now called the Nadaraya-Watson Kernel Regression.

Given some value of x, the conditional expectation $m(x) = \mathbb{E}[Y|X]$ is to be estimated, where $m(x)$ is some function. Similar to section 2.1, this is done by the use of a kernel K_h. Below is the estimator for $m(x)$.

$$\hat{m}(x_0) = \frac{\sum_{i=1}^{n} K_h(x_0 - x_i) y_i}{\sum_{i=1}^{n} K_h(x_0 - x_i)}.$$

In practice, by aggregating the values within the bandwidth and applying some kernel function, the overall trend of the function is captured. An example of this can be seen in Figure 8. There are many applications for performing such smoothing. One such example could be for finding the maximums and minimums of a function. For the function in Figure 8A, there are two problems.

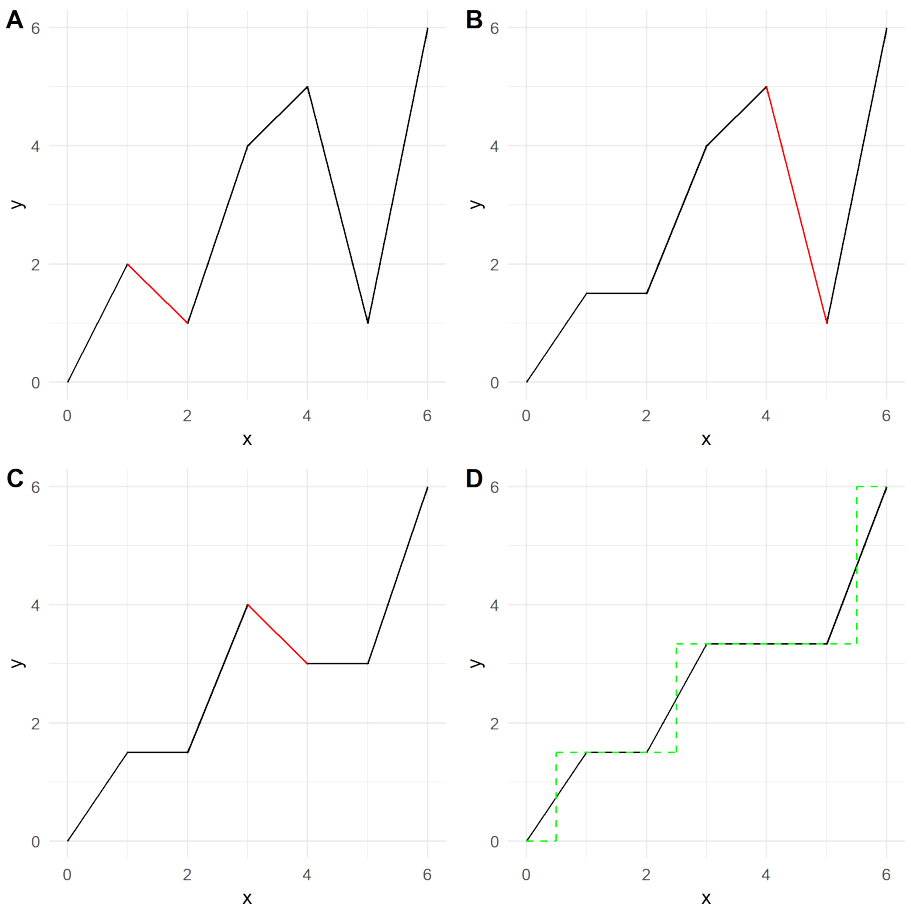

Figure 7: Workflow of finding the monotonic regression line. Here $x_i = i$ and y_i takes on some integer between 0 and 6. The black line **A** is the fitted function running through all pairs (x_i, y_i). This is altered in the following panels to generate a monotonic function. In each panel, the red line represent the earliest contradiction to having a monotonic function. Whenever $y_i > y_{i+1}$, both of their values are replaced with the average between them. This is done until no such contradiction exists. **A)** The first contradiction is in $y_2 = 2 > 1 = y_3$. Their values are replaced with 1.5 resulting in **B**. **B)** Next contradiction is $y_5 > y_6$ where the same procedure is performed. **C)** Previous corrections resulted in a new contradiction between y_4 and y_5. **D)** There are no more contradictions. In the end, the best fit line is a step-function as shown by the green dashed line.

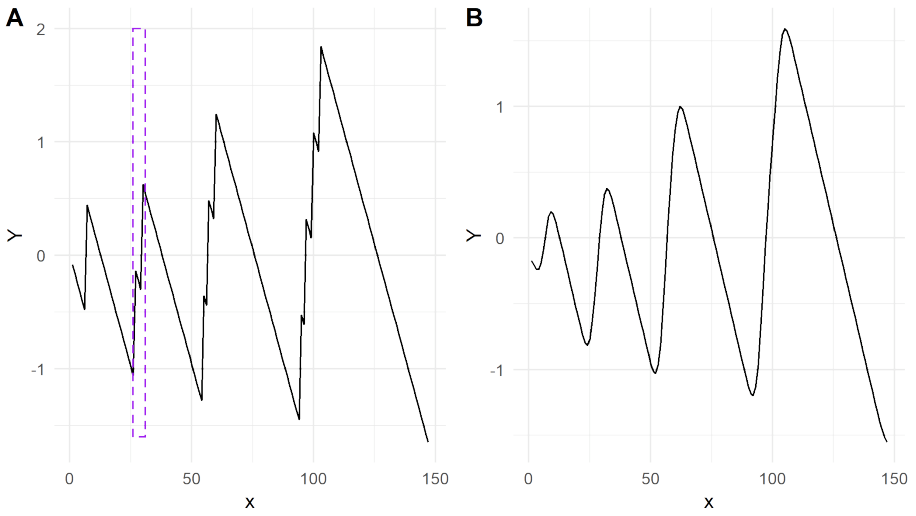

Figure 8: An illustration showing the use of Nadaraya-Watson Kernel Regression to smooth a function. **A)** This shows the shape of the true function. The size of the bandwidth of the kernel is showcased by the purple dashed rectangle. All y values within this rectangle are aggregated using some kernel and the value replaces the old value of the center point within the bandwidth. **B)** The result after applying kernel regression.

First of all, each maximum and minimum contains sharp points where the function clearly is not differentiable. Hence, it would not be possible to find the maximum or minimum by the use of conventional methods such as setting the derivative to zero. Secondly, there exist 4 distinct local maximum values of the functions and several less distinct local maximums. If only the 4 major local maximums should be considered, smoothing has to be applied as illustrated by Figure 8B.

It is, just as in section 2.1, important that the size of the bandwidth is correctly chosen. If the bandwidth is too large there will be over-smoothing which could remove some of the 4 distinct maximums in Figure 8. If the bandwidth is too small there could be under-smoothing where the less distinct maximums would remain or more could be introduced. If a fixt bandwidth is to be used anyways, it could be determined by using the h that minimizes the AMISE as explained in Section 2.1.

3.2.3 Generalization of Clustering Algorithm

In this thesis, there are two major generalizations to the algorithm in Section 3.1, a different density estimator and automatization of the decision graph. Using a kNN density estimator introduces an adaptive neighborhood size instead of the fixed d_c. This reduces the issues of using a fixed size of d_c as mentioned in Section 3.1. The adaptive neighborhood helps in making the algorithm more robust on data with varying densities and sizes of clusters. The kNN-density estimator additionally removes the possibilities of ties. In practice, k should be chosen to represent a reasonable-sized neighborhood around a point. If k is chosen too small, the neighborhood also becomes too small and introduces a large variance. If k instead is chosen too large, the neighborhood becomes too big and introduces a lot of bias.

A schematic showing the process of automating the choice of cluster centers from the decision graph is shown in Figure 9. First of all, a monotonic regression line $f(\log \rho)$ is constructed as in Section 3.2.1, using $\log \rho$ as a covariate and $\log \delta$ as the response variable. The monotonic regression line is fit in the log scale since ρ and δ have more of a linear relation in the log-scale than in the original scale. This can easiest be seen in Figure 5. In real-world cases, in the example of Figure 5, there would exist more outliers. Since the outliers have a very large δ and small ρ, there would be a decreasing exponential trend in the decision graph as points get closer to some cluster, decreasing their δ quickly while slightly increasing the value of ρ. This regression line is then placed on the decision graph as a decision line (see Figure 10). The decision line can be moved vertically with the following relation

$$\begin{aligned} \log \delta_i &= f(\log \rho_i) + c \\ \iff \delta_i &= e^{f(\log \rho_i)} \cdot e^c. \end{aligned} \tag{3}$$

By changing the value of the constant $\alpha = e^c$ the decision line is moved up and down (see Figure 9A). As the decision line is moved vertically on the

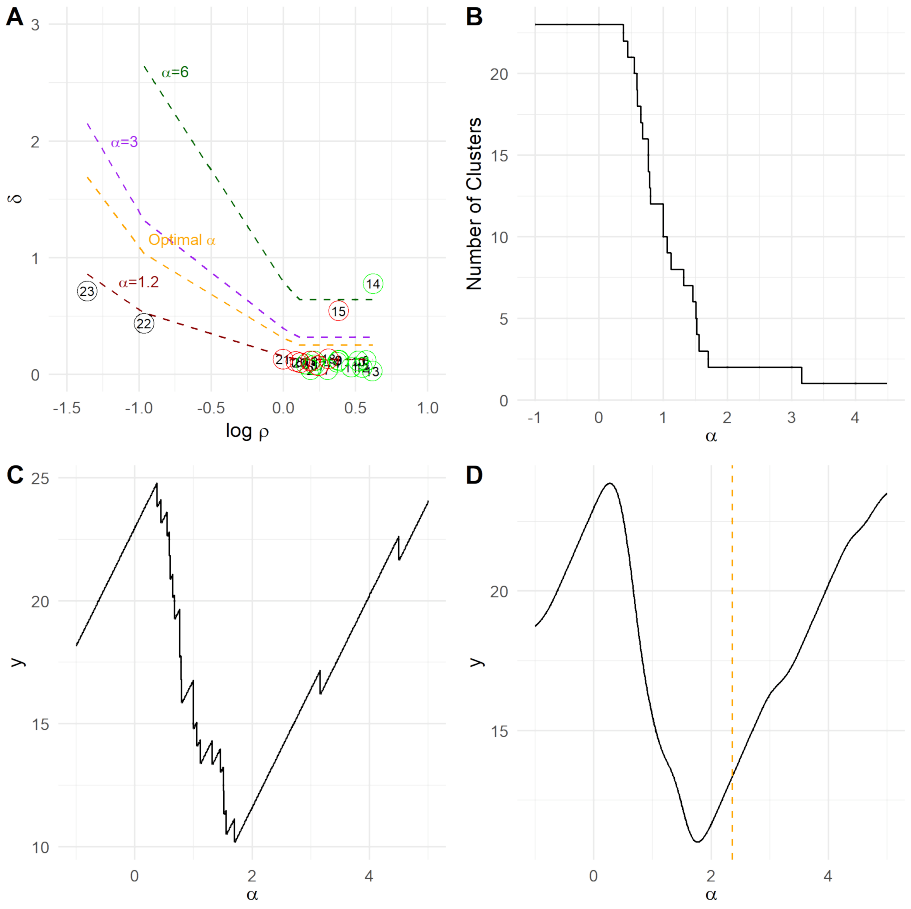

Figure 9: A schematic showing the automated process of finding cluster centers. **A)** A monotonic regression line $f(x)$ is fit on the data with pairwise entries $(\log \rho_i, \log \delta_i)$ and then applied on the decision graph as a decision line. For better visualization, the decision graph use δ over $\log \delta$ where the regression line is transformed as by equation 3. The regression line is moved up and down by changing the constant α in the equation $\delta_i = e^{f(\log \rho_i)} \cdot \alpha$. **B)** Any points above the decision line are chosen as cluster centers. As α change, the number of clusters also change. This graph shows the number of clusters as a function of the constant α. **C)** The step-function in **B** after linear detrending. This simply mean that $y = C - \alpha * b$ where C is the number of clusters in **B** and b is the ordinary least square estimation of the function in **B**. Now the start of a plateau is signified by a local minimum and the end of a plateau by a local maximum. **D)** Finally, the Nadaraya-Watson kernel regression is used on the projected step-function for smoothing. The 20% quantile of the longest upward trend is taken as the optimal α (orange dashed line).

decision graph, the cluster centers are any points lying above the decision line. This approach yields two new quantities, different values of the constant α and the number of clusters as a function of α (see Figure 9B). This creates a step-function where large gaps between points in the decision graph are shown as long plateaus. Since large gaps between points in the decision graph signify profound cluster centers as described in Section 3.1, this also means that long plateaus signify profound cluster centers.

One way to decide on the optimal value of α could be to simply pick a value within the longest plateau since this represents the largest gap in the decision graph. One way to find this is by linear detrending, i.e. projecting the step-function on its own linear regression line as seen in Figure 9C. Now the goal is to simply find the longest distance between a local minimum and its subsequent local maximum. In Figure 9A this would work. However, that is not always the case and is the reason why the smoothing step in Figure 9D is needed.

Imagine that in Figure 9A, $\delta_{14} = 2$. The largest gap would then instead be between point 15 and 14, meaning the longest plateau in Figure 9B would be for number of clusters equal to 1. This is clearly wrong since there are two clusters. By applying smoothing, the two plateaus for 1 and 2 clusters are merged into one as seen in Figure 9D. By choosing one of the smaller values of α in the new, merged line, the problem is resolved. For this thesis, the optimal value of the constant α is chosen as the lower 20% quantile of the longest upward trend. If the constant α is chosen too big, some cluster centers may be missed. Since it is impossible to add back missed cluster centers later, but possible to remove improper ones in a later stage, it is preferred to pick too many cluster centers.

The use of monotonic regression instead of e.g. linear regression is needed to prevent the inclusion of outliers (points with large δ and small ρ) as can be seen in Figure 10. If the linear regression line would have been used, point number 23, an outlier, could be picked up as a cluster center. Using monotonic regression greatly reduces this risk as illustrated by the figure.

4 Density Based Clustering for Large Data

In this section, a description of the full algorithm will be explained when used on large data such as *in-situ* transcriptomics data. A simplified flowchart showing the steps of the algorithm can be seen in Figure 11 and a more detailed algorithm can be seen in Algorithm 1.

The first step of the process is to compute the density estimation ρ. A comparative analysis between the KDE approach and the kNN approach was made and can be seen in section 5. Based on these results, the kNN approach described in section 2.2.2 is recommended. The value of k is set to 7 to maintain reasonable-sized neighborhoods.

One of the core parts of the method lies in choosing the appropriate δ. However, when the data set is too big, it is not feasible to calculate the distance between every pair of data points. Hence, a modification for large data sets is needed. To compute δ, the feature space is divided into a grid as can be seen in

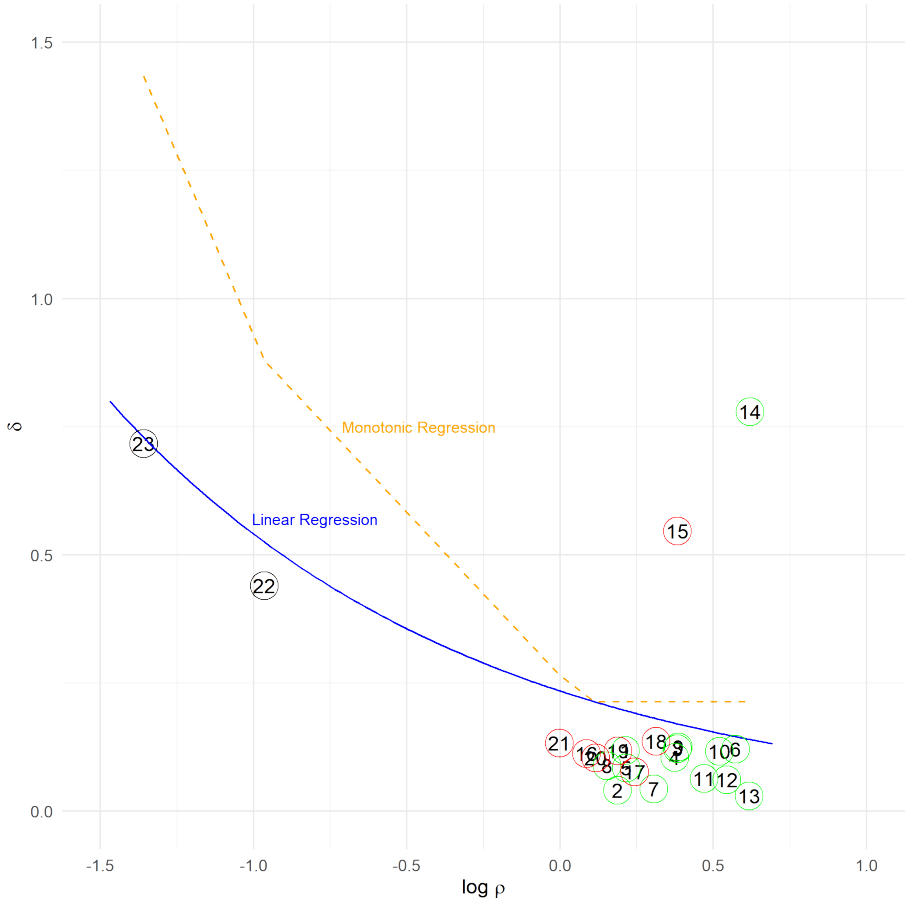

Figure 10: A decision graph for the generalized method similar to the decision graph in Figure 5. Here the density estimator is based on a kNN approach. A monotonic regression line and a linear regression line have been fit to the response $\log \delta$ and covariate $\log \rho$ and then applied to the decision graph as a decision line. Note that δ is used over $\log \delta$ as per equation 3 for easier visualization. The cluster centers are any points lying above the decision line. The decision line placement is chosen automatically as shown in Figure 9. As can be seen, it is more likely to pick up outliers using linear regression than monotonic regression which is why the monotonic regression approach is recommended.

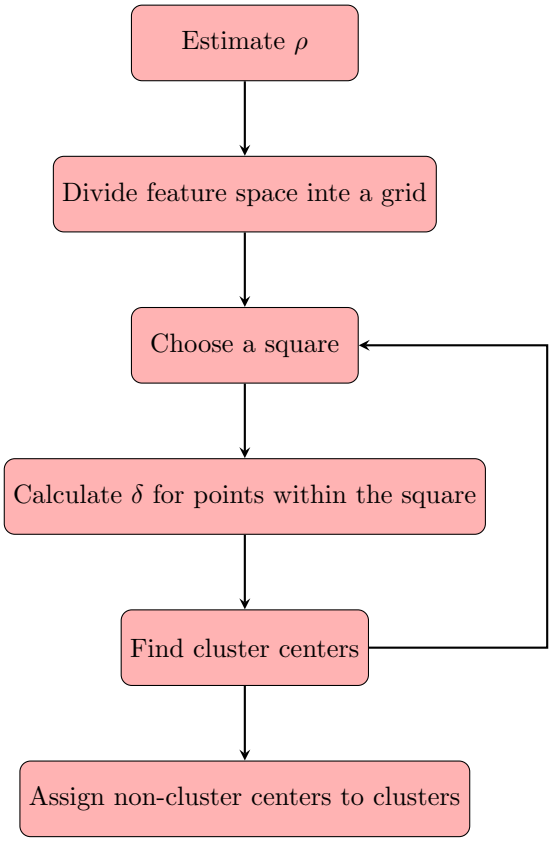

Figure 11: Simplified flowchart showing how the automated algorithm works for large data. First, the density is estimated for every data point. Next, the feature space is divided into a grid as in Figure 12. For each square within the grid, δ is calculated and cluster centers are found automatically by following the method from Section 3.2.3. When all squares are processed, clustering is performed as in Section 3.1.

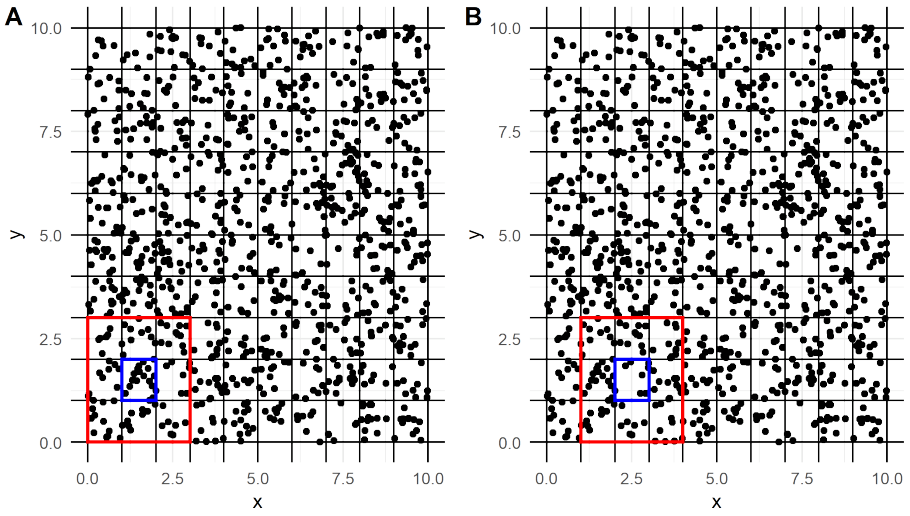

Figure 12: An illustration of grid-based use of the method in Section 3.2.3. In order to calculate δ for large data, the feature space is divided into a grid. Computations of δ and the automatized process of finding cluster centers as described in Section 3.2.3 are applied within each square one at a time. **A)** The value of δ is only calculated for points within the blue square using points within the red square followed by finding cluster centers. **B)** The same process as in **A** is done for the subsequent square. Note that, for points within squares at the edge, there are only neighboring squares in some directions. Computing δ and ρ for these squares would be biased and as such, the method of finding cluster centers is not applied to border squares.

Figure 12. The size of each square is $L/3 \times L/3$, where L is the average distance between cluster centers. The value of L is estimated by previous works and expertise within the field the data is taken from. If L is too small, the estimation of δ will not be robust and for large clusters, it is possible to get multiple cluster centers. If L is too big, the computational time is not sustainable.

Calculations for δ and finding cluster centers are done one square at a time. For every point within a square, δ is computed by only using data within that square and neighboring squares. For that same square, cluster centers are found by the automatic process described in Section 3.2.3, before moving on to the next square in the grid. These calculations are however not made for the squares at the edges. This restriction is made since when evaluating points in squares at the edges of the feature space, there will not exist any points on one side of the square. This makes the estimations biased. In a real-world example, the boundary of a tissue slice should not be evaluated anyhow. This is because the tissue has been cut, and as such, any cells near the boundary risk being incomplete. Finally, every non-cluster center is assigned to the same cluster as its nearest neighbor with a larger density.

This method is fully automated with no need for manual input by the user. Furthermore, it is a simple and intuitive algorithm that only requires one parameter k to be tuned and inserted into the algorithm. The method is quick due to having no iterative process like many other clustering methods and can easily be interpreted. Since it can find clusters of arbitrary and varying shapes, can manage large data, and is robust to varying densities of clusters the method works well for *in-situ* transcriptomics data.

Algorithm 1: The Algorithm

Input: Coordinates, k
Output: Clusters

Estimate densities ρ_i for every point x_i
Divide feature space into a grid with squares $\{j, k\} \in [0, J] \times [0, K]$
for $j = 1, 2, ..., J$ **do**
  for $k = 1, 2, ..., K$ **do**
    Only using points within squares $[j-1, j+1] \times [k-1, k+1]$;
     Estimate δ_i for points within square $\{j, k\}$
    Automatically find cluster centers within square $\{j, k\}$ using ρ_i
     and δ_i
  end
end
for $i = 1, 2, ..., N$ **do**
  Give each cluster center a unique cluster ID
  if x_i *is not a cluster center* **then**
    x_i is assigned the same cluster as its nearest neighbor with larger
     density ρ
  end
end

5 Results

In this section, a comparison between different density estimations is performed. Furthermore, an analysis of the algorithm in section 4 is given.

5.1 Data

In order to evaluate the method in Section 4 and check the assumptions made in previous methods, two different data sets will be used. One generated data set for evaluating the performance of the method and one transcriptomics data set. The *in-situ* transcriptomics data set named osmFish by Linnarsson Lab, comes from a part of a mouse brain and was used by Park *et al.* (2021) in evaluating their method. This data set will be used to check whether the assumptions of previous methods can be expected to be true, mainly the normalization process described in Section 5.4.1.

5.1.1 Simulated Data

For real-life *in-situ* transcriptomics data, it is unknown what cluster each gene belongs to. This makes it difficult to evaluate whether or not the clustering by a method is correct. Hence, a simulated data set Gaussian Mixture Model with 250 Gaussians (GMM250) is used. Doing so opens up the ability to better evaluate the performance of the method. To make a realistic comparison of the

model, the data set needs to resemble the real data. Therefore, a sample is taken from a GMM where each Gaussian represents a cell and each data point illustrates a gene.

The number of Gaussians in the GMM250 data is set to 250, with a total of 57047 data points distributed over a field of size 400×400 μm. The size of the field was set such that both sparse and dense areas were to be generated while ending up with a moderate amount of overlapping clusters. The size and shape of each Gaussian are randomly decided by generating variances for x and y from a uniform distribution within an interval from $\sqrt{3}$ to $\sqrt{11.6}$ μm. The mode of each Gaussian was randomly generated from a two-dimensional uniform distribution within the square $[0, 400] \times [0, 400]$. The number of points generated for each Gaussian was based on its size with some added randomness in the form of Gaussian noise. The data set is illustrated in Figure 13. The methodology and parameters for generating the data set were all based on characteristics from the data in Section 5.1.2.

Figure 14 show the position of each data point around the true mode of the Gaussian it belongs to. Every Gaussian has been projected such that its mode lies on the origin. This figure shows that the vast majority of points lie quite close to the mode of its Gaussian component and the density of points fades further from the mode. This is to be expected when simulating data from Gaussian distributions. Furthermore, it can be seen that the distribution of the data points is quite symmetric around the origin, illustrating that the data points are generated fairly and were not biased towards any particular direction. This is further shown in the top-right table showing that the mean position in both directions is close to 0 and hence not biased in any particular direction. The standard deviation is on average around 2.9, similar to the data used by Park *et al.* (2021), and as such, using the kernel bandwidth suggested by that paper should be close to the optimal bandwidth size for this data set.

5.1.2 Transcriptomics Data

The osmFish data set by Linnlarsson Lab consist of 1.976.659 observations of 33 different types of genes within a specific region of a mouse brain. This data will be used in order to check the assumptions made in the normalization method described in Section 5.4.1.

5.2 Analysis of Density Estimators

In previous attempts of performing clustering on *in-situ* transcriptomics data such as in the work of Park *et al.* (2021), KDE is used. As mentioned in section 2.1, there exists no one-size-fit-all bandwidth when the size of clusters varies a lot. The data gathered from images of tissues will almost always contain cells of varying size and shape and as such, using a KDE could prove problematic. Hence, a comparison between the KDE and a kNN density estimator is performed.

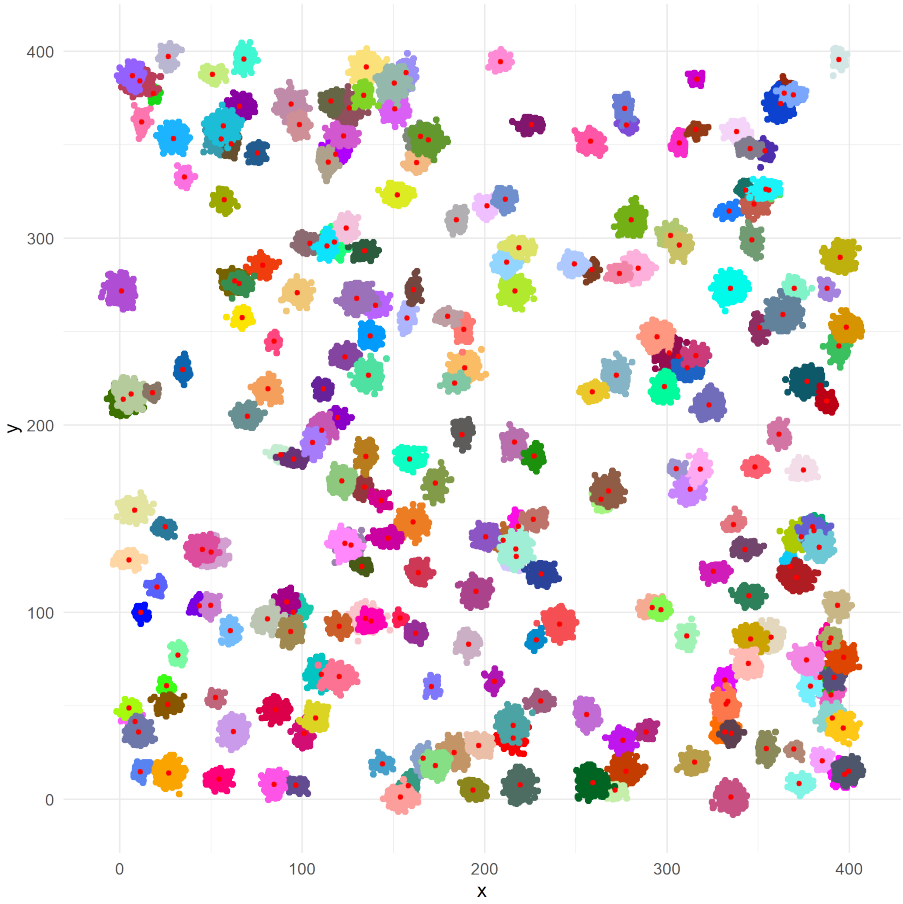

Figure 13: The GMM250 data set. The data was generated from a Gaussian Mixture Model containing 250 Gaussian components and 57047 data points. Each cluster is characterized by a different color and the mode of the Gaussians is shown as a red dot.

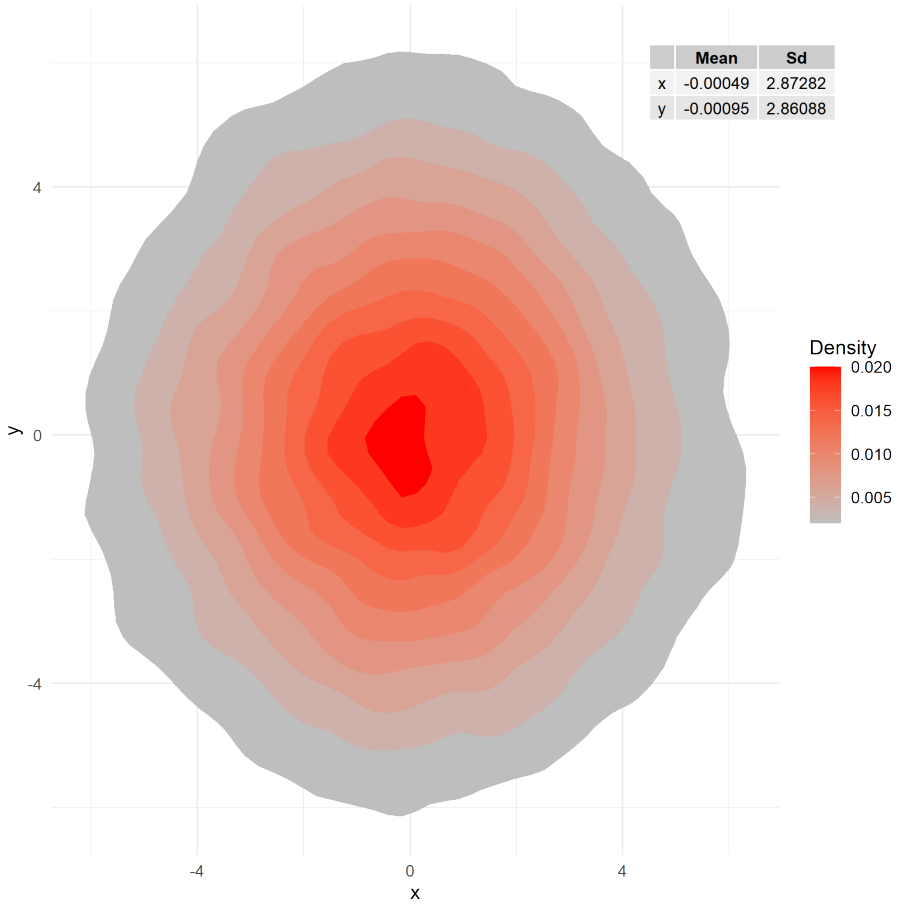

Figure 14: Distribution of the position of all data points from every Gaussian in the GMM250 data set in relation to its respective mode. Each of the 250 Gaussians has been projected such that each mode is positioned at the origin. This illustrates that the data points are generated around the true mode in a fair manner. The density indicates how many data points lie in that area. The data points are not more likely to lie in any particular direction than another. The smaller table in the top right corner shows the mean and standard deviation of the position of the data points. Since the mean is close to 0, it further strengthens the reasoning that the data points are generated around the mode fairly and not biased in any direction. The standard deviation of the Gaussians is on average around 2.9 in both directions and as such the bandwidth size of 2.5 as suggested by Park *et al.* (2021) is a good choice for the KDE.

Both methods were evaluated by first performing a point-wise density estimation of the GMM250 data. For the KDE, a Gaussian kernel was used where the size of the bandwidth was set to $2.5\mu m$ as suggested by Park *et al.* (2021). The kNN estimators used are the same as in section 2.2.2 with $k = 7$. Since the main interest of the algorithm is the position of the cluster center, the evaluation of the density estimators was done by examining the distance of the estimated mode to the true mode of the Gaussian. The direction of the estimated mode compared to the true mode was also examined to detect any possible bias in the position of the estimated modes.

To evaluate the position of the estimated mode in relation to each true mode, the methodology in Algorithm 2 was implemented. In practice, the search area around a true mode is expanded until there is a clear decrease in density. The estimated mode is then the data point with the largest density estimation within that search area. The Euclidean distance and spatial location between the true- and estimated modes are then calculated.

Algorithm 2: Estimated Mode Search

Input: Coordinates, Estimated Density, True Modes
Output: Estimated Modes

Denote M as the number of Gaussians in the data
Denote N as the number of data points
for $m = 1, 2, ..., M$ **do**
  Let z_m be the true mode of Gaussian m
  Let $\rho_{(i)}$ be the estimated density of the i-nearest neighbor $x_{(i)}$ of z_m
  Let S be the number of consecutive neighbors with smaller density than any previous neighbor
  Let $\Phi_m = \rho_{(1)}$ be the density of the current estimated mode of Gaussian m.
  Set $i = 2$
  while $S < 30$ **do**
    if $\rho_{(i)} > \Phi_{(m)}$ **then**
      $\Phi_{(m)} = \rho_{(i)}$
      $S = 0$
      $i = i + 1$
    end
    else
      $S = S + 1$
    end
  end
  Let the estimated mode of Gaussian m be the neighbor with the corresponding density estimation Φ_m
end

The distribution of the distance and spatial location of the estimated modes

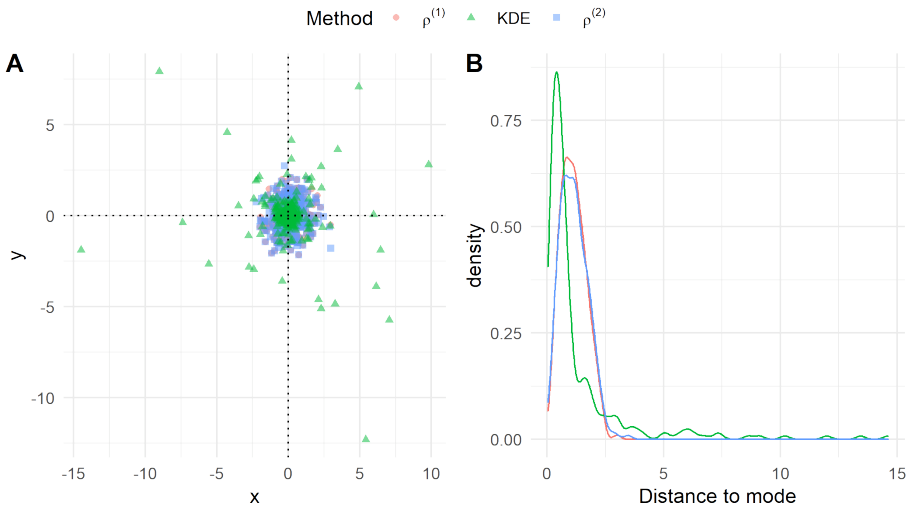

Figure 15: Analysis of density estimators. **A)** The spatial placement of each estimated mode in relation to the true mode of the GMM250 data set. The position of the estimated mode was found as described in Algorithm 2. **B)** The distribution of the distance of the estimated mode to the true mode. It is quite clear that using KDE results in many estimated modes lying far away from the true mode of the Gaussians.

in relation to their respective true mode can be seen in Figure 15. Figure 15A clearly shows that both kNN-density estimators perform very similarly to each other. It further shows that the location of the estimated modes towards the true modes seems to not be biased in any direction. However, the result of the KDE differs quite significantly. It is possible to see that a large amount of the estimated modes using KDE lie closer to the true mode in comparison to the kNN estimator. This is most likely the result of clusters where the kernel and the bandwidth of the kernel fit well with the underlying distribution. However, many of the estimated modes lie far away from the true mode as illustrated by the small but continuing peaks in Figure 15B. This problem arises when the bandwidth no longer fits the cluster size.

In many cases of the generated data, there are smaller clusters close to each other. In this scenario, when the bandwidth is too large, using the KDE results in merging the two clusters, ending up with a cluster center lying in between the two clusters as illustrated previously in Figure 6. This is further clear when noticing that the number of unique estimated modes using KDE is 228. Since the number of Gaussians in the data is 250, clearly a number of the Gaussians share an estimated mode. Using a kNN approach results in 249 unique modes. The kNN approach is showcased to be more robust than a KDE for cells of varying sizes and hence a kNN approach is suggested.

5.3 Analysis of Clustering of GMM250

In this section, an analysis of the complete method as described in Section 4 is performed. In light of the robustness of using a kNN approach as seen in Section 5.2 for estimating densities, this approach will be used. Both of these estimators performed very similarly, but since $\rho^{(1)}$ had a slight edge, this estimator is used.

To evaluate the effectiveness of the method, it was applied to the GMM250 data set. This resulted in finding 203 out of 250 clusters. A majority of the missed clusters were due to heavy overlapping between several clusters as can be seen in Figure 16. The remaining missed clusters lie at the boundary of the feature space and are shown with a black color. These clusters are not evaluated as mentioned in Section 4. Furthermore, in a real-world example, the number of clusters lying at the boundary of the feature space will be a very small fraction of the total number of clusters. If the clusters around the boundary of the feature space are disregarded, over 90% of the data is correctly clustered, where the only data wrongly clustered are from overlapping clusters. The final result of clustering is shown in Figure 16.

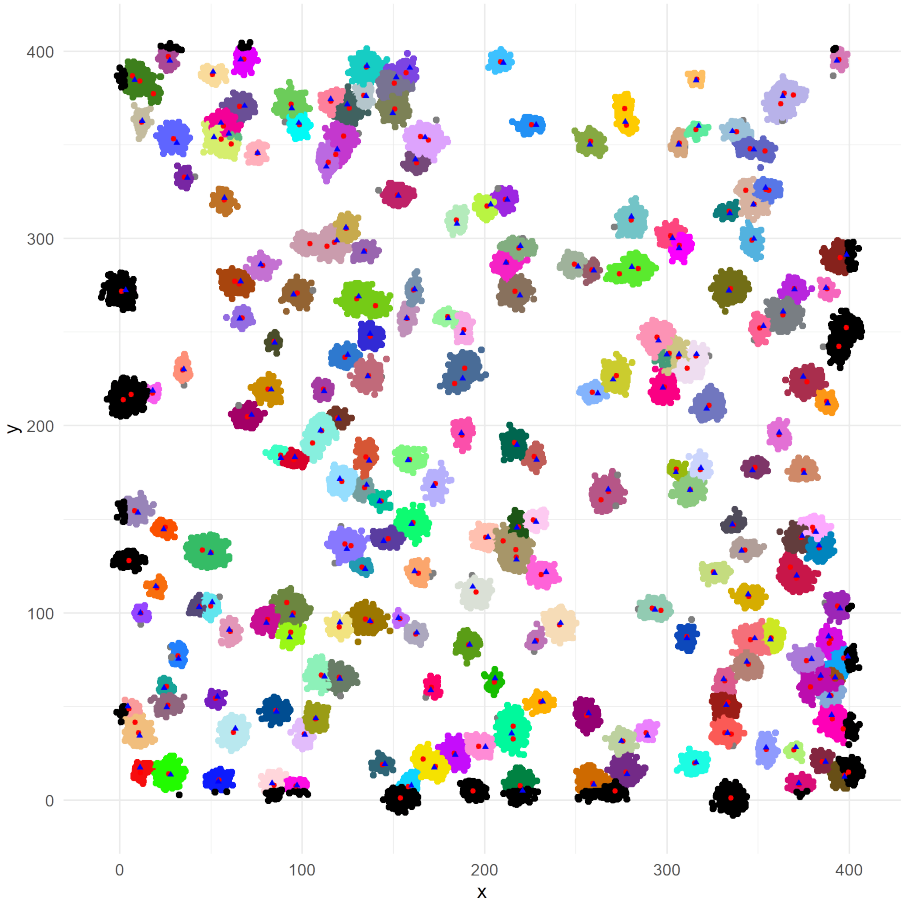

Figure 16: Result of using Algorithm 1 on the GMM250 data set. Each cluster is characterized by its own color where the true mode is shown as a red dot and the estimated mode as a blue triangle. The algorithm finds 203 clusters where a majority of missed cluster centers lie outside the grid, resulting in points belonging to those cluster centers not getting assigned to any cluster (Black colored points). The remaining missed cluster centers are due to largely overlapping clusters.

5.4 Analysis of Clustering Algorithm

This section will present one of the more recent methods for performing inference on *in-situ* transcriptomic data named Spot-based Spatial cell-type Analysis by Multidimensional mRNA density estimation (SSAM) by Park *et al.* (2021). The assumptions, robustness, interpretability, and the simplicity of this method will be compared to those for the method in Section 4.

5.4.1 Normalization

In most analysis, when the clustering of cells is performed, the gene counts in each cell is normalized due to bias in analyzing the tissue. There are different normalization methods, and one of the standards is *sctransform* by Hafemeister and Satija (2019). The method is based on the assumption that, when the clustering is completed, each cluster can be seen as a single cell sample, and as such, methods from previous works on single-cell analysis can be used.

For a cell j and some gene-type i, denote y_{ij} as the count for gene i in cell j, and x_j as the total number of genes in cell j. The authors then make the assumption that $\log y_{ij}$ and $\log_{10} x_j$ have a linear relationship and as such, the expected gene count y_{ij} can be modeled using a Generalized Linear Model (GLM) with log-link function and negative binomial error distribution. This results in the relation

$$\log \mathbb{E}[y_{ij}] = \beta_{0i} + \beta_{1i} \log_{10} x_j,$$

where β_{0i} is the intercept and β_{1i} is the effect parameter for gene-type i. The parameters of each gene type are decided such that there is a best fit across all cells. Furthermore, the dispersion parameter of the negative binomial distribution r_i for each gene type is estimated using the data. For a given gene-type i, this parameter is assumed to be independent of cell size and the total number of genes within cells.

One problem that occurs in this normalization method is that modeling each gene individually results in over-fitting. The solution suggested by the authors is to regularize the parameters by using the Nadaraya-Watson kernel regression. In doing so, the information across genes is shared and global trends can be learned. The regularized parameter estimators are given by

$$\tilde{\beta}_0(x) = \frac{\sum_{i=1}^n K_h(x - x_i)\beta_{0i}}{\sum_{i=1}^n K_h(x - x_i)}$$

$$\tilde{\beta}_1(x) = \frac{\sum_{i=1}^n K_h(x - x_i)\beta_{1i}}{\sum_{i=1}^n K_h(x - x_i)}$$

$$\tilde{r}(x) = \frac{\sum_{i=1}^n K_h(x - x_i) r_i}{\sum_{i=1}^n K_h(x - x_i)},$$

where the kernel K_h is a Gaussian kernel with a bandwidth size of 3 times the Sheather-Jones plug-in bandwidth (Sheather & Jones, 1991). This bandwidth is a variation of the bandwidth derived in Appendix A.1. Lastly, the expected gene count in cell j of gene-type i, z_{ij}, is defined as

$$\begin{aligned} z_{ij} &= \frac{y_{ij} - \mu_{ij}}{\sigma_i} \\ \mu_{ij} &= \exp(\tilde{\beta}_{0i} + \tilde{\beta}_{1i} \log_{10} x_j) \\ \sigma_i &= \sqrt{\mu_{ij} + \frac{\mu_{ij}^2}{\tilde{r}_i}}. \end{aligned}$$

This normalization method rely heavily on the assumption that there exists a linear relationship between $\log y_{ij}$ and $\log_{10} x_j$. From discussions with the team of experts within the field at SciLifeLab, this rarely holds. It is further visible in Figure 17 where the relation between the above mentioned statistics is illustrated for 4 different gene-types. The Tbr1 gene seems to have some logarithmic relation, while the other 3 genes have some quadratic or exponential relation. Furthermore, the assumption that the dispersion parameter r_i should be the same across cells is questionable.

An important part in distinguishing between different cell-types, which is one of the goals of this project, is based on the biological heterogeneity between different cells. That is, it is important to maintain the heterogeneity of the gene composition of cells. When performing normalization on the gene composition of cells, there exists a risk of normalizing this biological heterogeneity on top of the difference from technical errors of e.g. equipment. This would be detrimental to the analysis and could yield false results. Since there currently does not exist any well established normalization methods except for *sctransform*, and this method is based on several assumptions that does not always hold, the only normalization that is made for this method is a log-transformation as suggested by the experts at SciLifeLab.

5.4.2 Spot-based Spatial cell-type Analysis by Multidimensional mRNA density estimation (SSAM)

One of the more recent methods for clustering and inference on *in-situ* transcriptomics data is SSAM, constructed by Park *et al.* (2021). This is the only density-based clustering method for this type of data and is what some of the structure and ideas of the method in Section 4 were based on. SSAM is based on 3 steps.

First, the density of the genes is estimated using only one gene type at a time. The density estimation is done using a KDE as described in Section 2.1 with a Gaussian kernel and a bandwidth of $2.5\mu m$. The gene types are then merged onto the same feature space. The feature space is then divided into pixels of size $1\mu m$ by $1\mu m$ where each pixel is a vector with one element for every gene. Each element is the sum of the density of every such gene within that pixel.

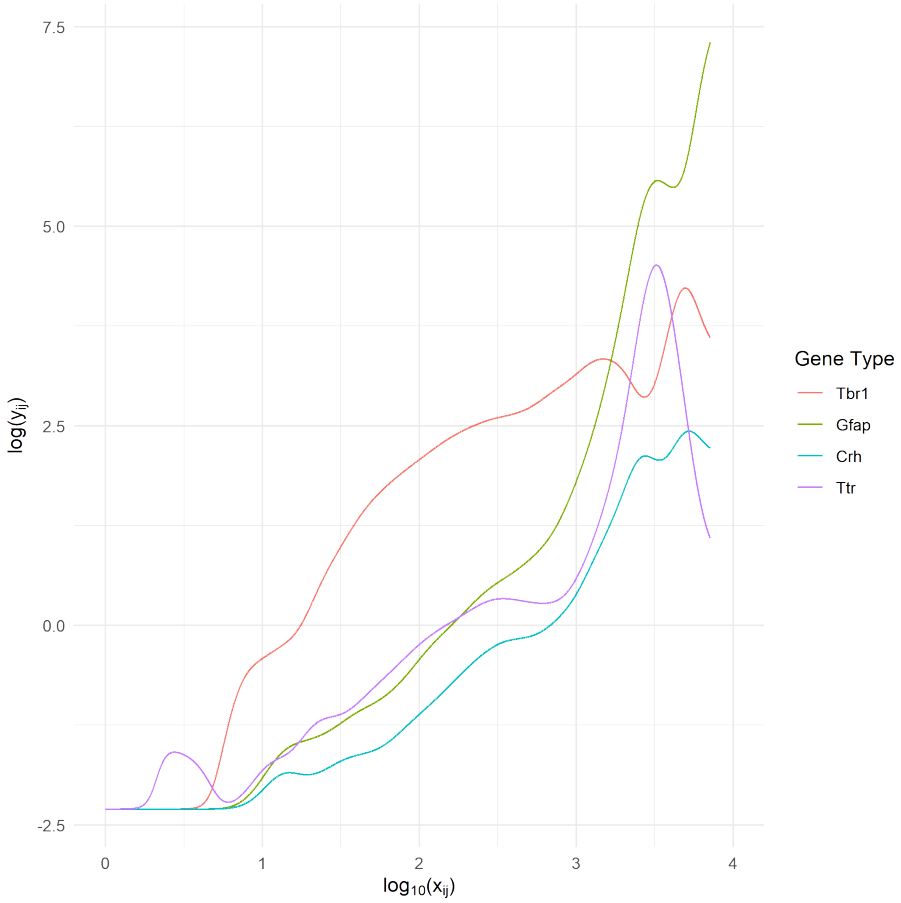

Figure 17: Testing the linearity assumption between $\log y_{ij}$ and $\log_{10} x_j$ as assumed by Hafemeister and Satija (2019) for 4 different gene-types in the osmFish data set. The x-axis shows the $\log_{10}$ of the total gene count per cell. The y-axis shows the natural logarithm of the gene count per cell for 4 different genes. A slight smoothing has been applied for better visualization of the trend using the method of Section 3.2.2. It is quite clear that the linearity assumption does not hold. The Tbr1 gene seems to have some logarithmic relation to the total gene count while the other genes have some quadratic or exponential relation. This result raises question to the robustness of *sctransform*.

Secondly, a down-sampling takes place by using a variation of max-pooling. In short, a maximum filter is applied to the pixelated feature space to dilute the image. The size of the filter is 3×3 pixels meaning, as the filter is moved across the image, every pixel within the filter copies the vector of the pixel with the largest density within this filter. The largest density is decided by the L1-norm applied to the vector of each pixel. In the end, only the pixels in the original space with a vector corresponding to a vector in the filtered space are kept as local maximums and potential cluster centers. Simplified, the down-sampling is done by selecting the pixels with the largest density while making sure no local maximum lies too close to another. These local maximums are considered to represent the gene composition of the entire cell. The normalization process as described in Section 5.4.1 is then applied to each pixel in the feature space.

Thirdly, the local maximums are clustered based on similarities between their gene expression vectors using some density-based clustering method e.g. DBSCAN where each cluster gets a unique cell-type ID. Before clustering, another down-sampling of the local maximums is performed. This process is controlled by a large number of thresholds and hyper-parameters, checking if the local maximum is in a dense enough area and contains enough genes. This down-sampling is done manually by the user. Finally, each pixel in the pixelated feature space is given a cell-type ID. This is done by comparing the gene expression vector of each pixel with those of the unique cell types using Pearson correlation. In the end, SSAM does not cluster cells but instead assigns each pixel of the image a cell type.

For SSAM to perform well, Park *et al.* (2021) assumes that cells are spherical and do not alter too much in size. This is often not the case for real data, and as such, the use of KDE for density estimation could be problematic as seen in Section 5.2 and Figure 6. Furthermore, since SSAM only performs density estimation on one gene type at a time, the data will be quite sparse and the density estimation will not be as robust. SSAM also has a lot of parameters that need to be input, e.g. bandwidth of the kernel in the KDE, the size of the filter in down-sampling, different thresholds, and any parameters needed for the clustering method of the local maximums. On top of the parameters, a manual down-sampling needs to be made by the user. This results in a rather complicated algorithm with many areas susceptible to errors, and many parameters in need of tuning whenever the result is not satisfactory. The method also relies heavily on the user to make decisions throughout the process. Furthermore, the use of *sctransform* for normalization can be questioned as seen in Section 5.4.1.

In light of this, the algorithm presented in Section 4 was constructed for clustering this type of data. This method uses a kNN density estimator instead of KDE, making it more robust to data with clusters of varying shape and density. Furthermore, the density estimation is performed on all the data instead of one gene type at a time. This makes it more robust to data with sparse regions where density estimations could be unreliable if only one gene type is used. While SSAM has a lot of parameters that need to be predetermined, the method in Section 4 only has one, k in the kNN density estimation. This gives a much simpler method that is easier to interpret. Furthermore, it minimizes

the risk of user error and makes it easier to fine-tune if the results do not look correct. The new method is also completely automatized, meaning it becomes much quicker to use and is less susceptible to human error. Furthermore, the authors mention that several parts of SSAM need to be altered when data from other tissues should be analyzed. However, the method in Section 4 is fully data-driven and directly applicable to any data set. Lastly, the normalization method in Section 5.4.1 is not used in the method of this thesis considering that the assumptions do not always hold.

6 Discussion

In this section, the primary discoveries of the thesis are reviewed and discussed. Some opportunities for improving the method in future works are also discussed.

In Section 2 different variations of estimating the density of a sample were introduced. The shortcomings of the KDE were described, mainly the inability of robust estimations for data containing areas with varying densities. Furthermore, another approach using kNN for estimating density was introduced. Section 3 contains the theory and analysis of density-based clustering. A generalization of the method by Rodriguez and Laio (2014) is stated. The generalized method is focused to preserve the speed and simplicity of the original method while making the process fully automated, removing the need for any manual decision making by the user. For Section 4, a further layer is added to the density-based clustering algorithm in order to tailor it towards large data sets, specifically *in-situ* transcriptomics data. In Section 5, different results are presented. A simulation for analyzing different density estimators was performed, showing that a kNN approach is more robust than a KDE for data with clusters of varying size and density. The method in Section 4 was analyzed on a generated data set where all clusters without major overlap were found.

The goal of this thesis was fulfilled when a completely automatized algorithm for clustering genes from *in-situ* samples into cells was constructed. The algorithm is quick, robust to large data sets, easy to interpret, simple, and only relies on a single parameter, k, for the density estimation. In contrast to previous methods, the algorithm is further robust when the data set varies a lot in densities, cluster size, and shape, and is built upon assumptions that have been confirmed to be true.

The use of Euclidean distance is often standard practice but may not always be usable. One of the major problems with using Euclidean distance to measure δ is when working with non-linear shapes. For example, if a cluster is shaped like the letter C and both end-points are dense areas, points in both of those areas will have a large density estimation ρ. Now, if the Euclidean distance between the end-points is also big, one point at each end of the shape could end up with a large δ. This results in one cluster having two cluster centers meaning over-clustering is introduced. At the same time, as seen in Figure 6A, the use of Euclidean distance can also introduce under-clustering when two clusters are close to each other. In order to prevent both of these problems,

future improvements of the method will replace Euclidean distance for a graph distance. This will be useful for smaller data sets when the similarity graph can be computed. One suggestion for the choice of graph distance is the Commute Time Distance (CTD). In short, the CTD is a measure of how much time it would take to get from a data point i to a data point j and back by performing a random walk on the graph. If there are many short paths between two points, the CTD will decrease while if the paths are long and scarce, the time increase. Hence, two points close to each other within the same dense area will have a small CTD while two points close to each other in separate dense areas will have a large CTD. This is very beneficial for clustering when clusters intertwine. For the over-clustering scenario, since this is a single cluster, there would be many short paths between the two points at the end-points of the C shape and as such, the CTD would be small. Since δ was computed as the distance between a point to its nearest neighbor with a larger density, and the point with the second-largest density would have a short distance to the point with the largest density within the shape according to the CTD, it would have a small δ and not be classified as a cluster center. For the under-clustering case, even if two clusters are close to each other, there will not be many paths crossing between them. Hence, the CTD between the points with the highest density in each cluster would be large, and as such both points would have large ρ and δ, meaning they would become cluster centers solving the under-clustering issue.

Another improvement of the method lies in the clustering. When a data point is close to the cluster center, it is clear which cluster that point belongs to. However, a large portion of points will be seen at the edges of clusters and sometimes in between two different clusters. In these cases, the assignment to clusters is not as clear. Therefore a fuzzy-clustering approach should be introduced to the algorithm. By using fuzzy clustering, the probability of a point belonging to some cluster is used instead of a yes or no as for hard clustering. This gives flexibility since points can contribute to more than one cluster.

A Appendix

In the paper by Kraskov *et al.* (2004) and the paper by Sheather (2004), many of the derivations are left out. Hence, in this appendix, the derivation for the choice of bandwidth by minimizing the asymptotic mean integrated squared error (AMISE) is performed as well as a more detailed derivation of equation 2 in Section 3 which was left out by Kraskov *et al.* (2004).

A.1 Derivation of AMISE of Kernel Density Estimation

The AMISE consists of the bias and variance of the kernel density estimate. Hence the derivation of those is first shown. Denote $X_1, X_2, ..., X_n$ as an independent sample from some random variable with density f. The kernel density estimation (KDE) at point x_0 is then defined as

$$\hat{f}_h(x_0) = \frac{1}{hn} \sum_{i=1}^{n} K_h(\frac{X_i - x_0}{h}),$$

for some Kernel K with bandwidth h. The kernel K is chosen to be a unimodal distribution symmetric around 0 and as such fulfills the following statements

$$\begin{aligned} \int K(z)dz &= 1 \\ \int K(z)zdz &= 0. \end{aligned} \tag{4}$$

The bias is then defined as

$$\begin{aligned} \text{Bias}(\hat{f}_h(x_0)) &= \mathbb{E}[\hat{f}_h(x_0)] - f(x_0) \\ &= \mathbb{E}[\frac{1}{hn} \sum_{i=1}^{n} K(\frac{X_i - x_0}{h})] - f(x_0) \\ &= \frac{1}{hn} \sum_{i=1}^{n} \mathbb{E}[K(\frac{X_i - x_0}{h})] - f(x_0). \end{aligned}$$

Since $X_1, X_2, ..., X_n$ are independent and identically distributed

$$\mathbb{E}[X_1] = \mathbb{E}[X_2] = ... = \mathbb{E}[X_n].$$

Therefore,

$$\begin{aligned} \mathbb{E}[\hat{f}_h(x_0)] - f(x_0) &= \frac{1}{h} \mathbb{E}[K(\frac{X_i - x_0}{h})] \\ &= \frac{1}{h} \int K(\frac{x - x_0}{h}) f(x)dx - f(x_0), \end{aligned}$$

where the last equality follows from the definition of expectation. Furthermore, substituting $y = \frac{x - x_0}{h}$ and $dx = h \cdot dy$ yields

$$\begin{aligned}\mathbb{E}[\hat{f}_h(x_0)] - f(x_0) &= \int K(\frac{x-x_0}{h})f(x)\frac{dx}{h} - f(x_0) \\ &= \int K(y)f(x_0+yh)dy - f(x_0).\end{aligned} \tag{5}$$

By taylor expansion around the point x_0

$$f(x_0+yh) = f(x_0) - yhf'(x_0) + \frac{h^2y^2}{2}f''(x_0) + o(h^2).$$

Using this in equation 5 yields

$$\begin{aligned}\mathbb{E}[\hat{f}_h(x_0)] - f(x_0) &= \int K(y)\{f(x_0) - yhf'(x_0) + \frac{h^2y^2}{2}f''(x_0) + o(h^2)\}dy - f(x_0) \\ &= \int K(y)f(x_0)dy - \int K(y)yhf'(x_0)dy + \int K(y)\frac{h^2y^2}{2}f''(x_0)dy + o(h^2) - f(x_0) \\ &= f(x_0)\int K(y)dy - hf'(x_0)\int K(y)ydy + f''(x_0)\frac{h^2}{2}\int K(y)y^2dy + o(h^2) - f(x_0).\end{aligned}$$

Now, using the properties of the Kernel K from equation 4 simplifies this expression of the bias

$$\begin{aligned}\mathbb{E}[\hat{f}_h(x_0)] - f(x_0) &= f(x_0)\cdot 1 - hf'(x_0)\cdot 0 + f''(x_0)\frac{h^2}{2}\int K(y)y^2dy + o(h^2) - f(x_0) \\ &= f''(x_0)\frac{h^2}{2}\int K(y)y^2dy + o(h^2).\end{aligned}$$

Hence, the bias can be expressed as

$$\text{Bias}(\hat{f}_h(x_0)) = f''(x_0)\frac{h^2}{2}\mu_2(K) + o(h^2),$$

where

$$\mu_2(K) = \int K(y)y^2dy$$

is the second moment of K. An upper bound of the variance follows from similar calculations

$$\begin{aligned}\text{Var}(\hat{f}_h(x_0)) &= \text{Var}(\frac{1}{hn}\sum_{i=1}^{n}K(\frac{X_i-x_0}{h})) \\ &= \frac{1}{h^2n}\text{Var}(K(\frac{X_i-x_0}{h})) \\ &= \frac{1}{h^2n}(\mathbb{E}[K(\frac{X_i-x_0}{h})^2] - \mathbb{E}[K(\frac{X_i-x_0}{h})]^2) \\ &\leq \frac{1}{h^2n}\int K(\frac{x-x_0}{h})^2 f(x)dx,\end{aligned}$$

where the second equality follows again from the fact that $X_1, X_2, ..., X_n$ is an i.i.d. sample and the third equality follow from the definition of variance. Using the same substitution as for the bias as well as the same taylor expansion, the variance can continued be written as

$$\begin{aligned}
\text{Var}(\hat{f}_h(x_0)) &\leq \frac{1}{h^2 n} \int K(y)^2 f(x_0 + yh) h dy \\
&= \frac{1}{hn} \int K(y)^2 \{f(x_0) + yhf'(x_0) + o(h)\} dy \\
&= \frac{1}{hn} \{f(x_0) \int K(y)^2 dy + o(h)\} \\
&= \frac{1}{hn} f(x_0) \int K(y)^2 dy + o(\frac{1}{hn}).
\end{aligned}$$

Hence, an upper bound to the variance of the KDE is

$$\text{Var}(\hat{f}_h(x_0)) \leq \frac{1}{hn} f(x_0) R(K) + o(\frac{1}{hn}), \tag{6}$$

where $R(K) = \int K(y)^2 dy$. In fact, Sheather (2014) states that there is an equality for equation 6.

As n becomes large, $o(\frac{1}{hn})$ is negligible. Hence, by squaring the bias and combining it with the leading variance produce the asymptotic mean squared error (AMSE) (Sheather, 2014)

$$\text{AMSE}(\hat{f}_h(x_0)) = \frac{1}{hn} R(K) f(x_0) + \frac{h^4}{4} \mu_2(K)^2 (f''(x_0))^2,$$

and integrating the AMSE produce the AMISE.

$$\text{AMISE}(\hat{f}_h(x_0) = \frac{1}{hn} R(K) + \frac{h^4}{4} \mu_2(K)^2 R(f''),$$

where

$$R(f'') = \int (f''(z))^2 dz.$$

When performing KDE, a suitable bandwidth h needs to be chosen. One way would be to choose h minimizing the AMISE. Since the kernel K is known, the only unknown part of the AMISE is $R(f'')$. Sheather (2014) mention that the h minimizing the AMISE is

$$h_{\text{AMISE}} = (\frac{R(K)}{\mu_2(K)^2 R(f'') n})^{1/5}.$$

Since $R(f'')$ is unknown, one approach in finding h_{AMISE} is by replacing $R(f'')$ with an estimate $R(\hat{f}_g'')$ where g is some bandwidth predetermined by the user. In that case, the bandwidth h that minimize the AMISE becomes

$$h_{\text{AMISE}_g} = (\frac{R(K)}{\mu_2(K)^2 R(\hat{f}_g'') n})^{1/5}.$$

A.2 Derivation of Equation 2

This is a more detailed derivation of equation 2 in Section 2.2.1. Note that by basic derivative rules and simple calculations

$$\frac{d}{dk}\omega^{k-1}(1-\omega)^{N-k-1} = x^{k-1}(1-x)^{N-k-1}(\log(x)-\log(1-x))$$
$$\frac{d}{dN}\omega^{k-1}(1-\omega)^{N-k-1} = x^{k-1}(1-x)^{N-k-1}\log(1-x).$$

Hence, the integral in equation 2 can be rewritten as

$$\int_0^1 \omega^{k-1}(1-\omega)^{N-k-1}\log\omega d\omega = \frac{d}{dk}\int_0^1 \omega^{k-1}(1-\omega)^{N-k-1} + \frac{d}{dN}\int_0^1 \omega^{k-1}(1-\omega)^{N-k-1}.$$

The integrals on the right-hand side are the definition of the Euler integral of the first kind

$$\begin{aligned} B(x,y) &= \int_0^1 t^{x-1}(1-t)^{y-1}dt \\ &= \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}, \end{aligned}$$

where the derivation for the last equality has been shown by Artin (2015). Further note that the digamma function is defined as

$$\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}.$$

Thus, the integral becomes

$$\begin{aligned} \int_0^1 \omega^{k-1}(1-\omega)^{N-k-1}\log\omega d\omega &= \frac{d}{dk}B(k,N) + \frac{d}{dN}B(k,N) \\ &= \frac{d}{dk}\frac{\Gamma(k)\Gamma(N-k)}{\Gamma(N)} + \frac{d}{dN}\frac{\Gamma(k)\Gamma(N-k)}{\Gamma(N)} \\ &= \frac{\Gamma'(k)\Gamma(N-k)-\Gamma(k)\Gamma'(N-k)}{\Gamma(N)} + \Gamma(k)\frac{\Gamma'(N-k)\Gamma(N)+\Gamma'(N)\Gamma(N-k)}{\Gamma(N)^2} \\ &= \frac{\Gamma(k)\Psi(k)\Gamma(N-k)-\Gamma(k)\Gamma(N-k)\Psi(N-k)}{\Gamma(N)} \\ &+ \Gamma(k)\frac{\Gamma(N-k)\Psi(N-k)\Gamma(N)+\Gamma(N)\Psi(N)\Gamma(N-k)}{\Gamma(N)^2} \\ &= \frac{\Gamma(k)\Gamma(N-k)}{\Gamma(N)}\{\Psi(k)-\Psi(N-k)\} + \frac{\Gamma(k)\Gamma(N-k)}{\Gamma(N)}\{\Psi(N-k)+\Psi(N)\} \\ &= \frac{\Gamma(k)\Gamma(N-k)}{\Gamma(N)}\{\Psi(k)-\Psi(N)\}. \end{aligned}$$

By definition, $\Gamma(x) = (x-1)!$, showing the last equality in equation 2 in Section 2.2.1

$$
\begin{aligned}
k\binom{N-1}{k}\int_0^1 \omega^{k-1}(1-\omega)^{N-k-1}\log\omega d\omega &= k\binom{N-1}{k}\frac{\Gamma(k)\Gamma(N-k)}{\Gamma(N)}\{\Psi(k)-\Psi(N)\} \\
&= k\frac{(N-1)!}{k!(N-k-1)!}\frac{(k-1)!(N-k-1)!}{(N-1)!}\{\Psi(k)-\Psi(N)\} \\
&= \frac{k}{k!}(k-1)!\{\Psi(k)-\Psi(N)\} \\
&= \Psi(k)-\Psi(N).
\end{aligned}
$$

B Reference

Artin, E. (2015). The gamma function. Courier Dover Publications.

Barlow, R. E. (1972). Statistical inference under order restrictions; the theory and application of isotonic regression (No. 04; QA278. 7, B3.).

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd (Vol. 96, No. 34, pp. 226-231).

Hafemeister, C., & Satija, R. (2019). Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. Genome biology, 20(1), 1-15.

Hartigan, J. A. (1975). Clustering algorithms. John Wiley & Sons, Inc..

Kraskov, A., Stögbauer, H., & Grassberger, P. (2004). Estimating mutual information. Physical review E, 69(6), 066138.

Leeuw, J., D. (2005). Monotonic Regression. Encyclopedia of Statistics in Behavioral Science **3**, 1260-1261

Loftsgaarden, D. O., & Quesenberry, C. P. (1965). A nonparametric estimate of a multivariate density function. The Annals of Mathematical Statistics, 36(3), 1049-1051.

Nadaraya, E. A. (1964). On estimating regression. Theory of Probability Its Applications, 9(1), 141-142.

Park, J., Choi, W., Tiesmeyer, S., Long, B., Borm, L. E., Garren, E., ... & Ishaque, N. (2021). Cell segmentation-free inference of cell types from in situ transcriptomics data. Nature communications, 12(1), 1-13.

Partel, G., & Wählby, C. (2021). Spage2vec: Unsupervised representation of localized spatial gene expression signatures. The FEBS journal, 288(6), 1859-1870.

Petukhov, V., Xu, R. J., Soldatov, R. A., Cadinu, P., Khodosevich, K., Moffitt, J. R., Kharchenko, P. V. (2022). Cell segmentation in imaging-based spatial transcriptomics. Nature Biotechnology, 40(3), 345-354.

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. science, 344(6191), 1492-1496.

Sheather, S. J., & Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. Journal of the Royal Statistical Society: Series B (Methodological), 53(3), 683-690.

Sheather, S. J. (2004). Density estimation. Statistical science, 588-597.

Watson, G. S. (1964). Smooth regression analysis. Sankhyā: The Indian Journal of Statistics, Series A, 359-372.

Wishert, D. (1969). Mode analysis: a generalization of nearest neighbour which reduces chaining effects (with discussion). Numerical taxonomy, 282-311.