BACHELOR THESIS

by Paul Fink

October 2, 2009

K–Ward–microaggregated data in linear models

An analytical approach to obtain unbiased estimators under anonymized data

Supervision:
Prof. Dr. Thomas Augustin

Department of Statistics
Ludwig–Maximilians–University
Munich

Contents

Abstract

This thesis deals with the effect of microaggregated data obtained by using k–Ward–Algorithm on estimators of linear regression. Therefore, at first an overview over the strategy of k–Ward–microaggregation is given in chapter 2, describing the underlying idea. Further, improvements concerning disclosure control and computing time are discussed.
The first section of chapter 3 focuses on estimators, in case the data set is aggregated according to all regressors of the regression model. It is proposed that the estimators of the slope and intercept coefficients remain unbiased in this particular case. Furthermore, it is possible to obtain an unbiased estimator of the underlying true error variance in case the response is sorted concomitantly and an upper bound of it otherwise.
Not as obvious as before are the results in the second section of chapter 3, in which the case of a regression is illustrated when the response, regressors and other model–excluded variables are employed in the group forming process of k–Ward–Algorithm. It proves to be difficult to give a consistent estimator of any parameter as the assumption of a latent underlying sorting variable causes problems. First in determining such a variable, and secondly if such one exists the covariance between the regressors itself and the response can only be bounded allowing no clear bound of the least–square estimator for the slope coefficients. Only in case the group–sizes are assumed as random it is possible to determine a consistent estimator, requiring a known sorting variable.
In chapter 4 the theoretically achieved results are tested in simulations. The outcome does not allow to refute the conclusions stated in the first section of chapter 3.

Chapter 1

Introduction

A huge volume of data is produced by national and international organizations in order to obtain information about a specific topic. These produced data can be classified in two different categories, *macrodata* and *microdata* sets. While the first are mainly statistical tables, created by aggregating microdata, the latter contain individual records. As these are usually critical personal information they have to be kept safe to prevent both misuse and disclosure. Mostly these data are used once and are deleted afterwards. However, there is an interest of scientists in these data, as they usually provide a large number of observations and appropriately drawn samples.

In order to use this data for scientific research they have to be anonymized, guaranteeing that the individual records are not disclosable by the researchers. However, as such an *absolute anonymized* microdata set would provide almost the same amount of information as the according makrodata set, the concept of *factual anonymity* is applied, meaning that '*an allocation of the individual data is possible only with an excessive amount of time, expenses and manpower*'[1]. A variety of anonymization methods holding that concept can be applied to a microdata set ranging from information reduction to data perturbing methods. To the latter one belongs the microaggregation technique which is appropriate for continuous data. The main idea of this technique is to form groups of observations and afterwards replace the individual records by their group average value. In the process of forming groups the data set is sorted according to a predetermined criterion, but re–ordered afterwards in the same way, so that in the end the microaggregated data set is in the same order as at the beginning.

This thesis concentrates on the microaggregation method k–Ward–microaggregation, which was developed by Mateo-Sanz and Domingo-Ferrer in 1998. The advantage of this method is the minimization of the information loss, i.e. the variance within the groups and thus allowing precise estimators in regression models in comparison to the estimators on the original data.

In chapter 2 the algorithm to microaggregate is presented and described. Furthermore it is illustrated that the above presented advantage could be a dis-

[1] Law on Statistics for Federal Purposes §16 (6) (2007)

advantage leading to severe disclosure risk. As a solution another algorithm, implementing the original k–Ward–Algorithm, is roughly described.

The chapter 3 focuses on obtaining least–square estimators in linear regression models in case of k–Ward–microaggregated data. The first section deals with the case that all regressors are taken into account at the microaggregation process. The following section deals with the more problematic case in which the variables used for the microaggregation are various variables including response, regressors and other variables, excluded from the regression model. Although there is not a straightforward solution to obtain consistent estimators, it is demonstrated that under some circumstances and assumptions it is still possible.

An overview of simulations is given in chapter 4, testing the hypotheses generated in the chapter 3.

Finally in chapter 5 a conclusion of this thesis and a prospect for further research in this context are presented.

Chapter 2

k–Ward-Microaggregation

2.1 Concept of K–anonymity

In order to obtain a microaggregated data set a variety of methods can be applied to the original data whose final output holds the standard of *K–anonymity.* *K–anonymity* means, that for each variable in the microaggregated data set every value occurs at least k times per variable. This makes it more difficult for an attacker to identify a single observation.

To achieve this, the original data set is partitioned into disjoint groups, containing at least K values each, so called k–partition, and in a second step the original values are replaced by their group average. In the process of forming groups all methods aim to group only the most similar values together, though holding the property of *K–anonymity.*

First of all, a universal notation is needed: Let X denote the original data set containing n observation vectors, with each p continuous variables. The i–th observation vector is written as x_i. After the partition the original values are assigned to g disjoint groups $G_1, \ldots, G_g$. The number of observations in G_k is denoted by n_k. The vector (length: p) with the aggregated means in the k–th group is hereby $\bar{x}_k$.

Finding an *optimal* partition requires a measure of the information loss which arises naturally in the process of forming groups, except for artificial cases. Mateo-Sanz and Domingo-Ferrer (1998) proposed to use the within–groups sum of squares

$$SSE := \sum_{k=1}^{g} \sum_{x_i \in G_k} (x_i - \bar{x}_k)' (x_i - \bar{x}_k) \tag{2.1}$$

as a measure for the information loss. The strategy in minimizing *SSE* seems to be appropriate as the empirical covariance matrix S_{xx} can be divided into the variance within (SSE) and between (SSA) the groups:

$$SST := nS_{xx} = \underbrace{\sum_{k=1}^{g} n_k (\bar{x}_k - \bar{x})' (x_i - \bar{x}_k)}_{SSA} + SSE \ , \tag{2.2}$$

where $\bar{x}$ is the vector of the means computed from the original data set.
A standardized definition of the information loss L is as followed:

$$L := \frac{SSE}{SST} \ , \tag{2.3}$$

with $0 \leq L \leq 1$. As SST is a fixed value for a given data set, minimizing SSE assures that the most similar values are grouped together. Equivalent to the above minimizing is a maximizing of SSA since L can also be written as

$$L = \frac{SSE}{SST} = \frac{SST - SSA}{SST} = 1 - \frac{SSA}{SST} \ . \tag{2.4}$$

This can be summarized by the K–partition-problem

Definition 1 (K–partition–problem)**.** *Partition the observation in the original data set into disjoint groups such that*

1. *each group consists of at least K observations (to assure K–anonymity), and*
2. *the variance within the groups (measured by SSE) is minimized.*

Definition 2 (Optimal K–partition)**.** *A partition solving the K–partition–problem is called* optimal K–partition.

Similar to finding a partition with minimal within group variance is the grouping process in statistical cluster analysis. However, the constraint there is a fixed number of groups in the obtained data set, whereas the K–partition–problem requires a minimum group–size. Although these two approaches seem not to be matchable, Mateo-Sanz and Domingo-Ferrer (1998) proposed a method striking a balance between those: k–Ward–Algorithm.

This method is further explained in Section 2.2. Section 2.3 focuses on the disclosure issue of the k–Ward–Algorithm, and describes an improved *secure-k–Ward–Algorithm* by Li et al. (2002).

2.2 k–Ward–Algorithm

It is an heuristic method approximating an optimal K–partition, which adapts the clustering technique, Ward's hierarchical algorithm, with the requirements of K–anonymity.

Before going into the details of this algorithm, a brief overview on Ward's hierarchical clustering method is given, using his notation.

2.2.1 Excursus: Ward's clustering algorithm[1]

As objective function Z Ward proposed to use the within–groups sum of squares (SSE), because it minimizes the variance within the groups, hence a lower value is considered more desirable.

[1]cf. Ward (1963), page 239ff

At the beginning, the whole data set U consists of n one–element subsets: $U = \{e_1, \dots, e_n\}$. The number of subsets is reduced to $n-1$ by uniting those two subsets to a new subset, which minimize the change in the objective function's value. This results in optimal subsets at the step $n-1$, denoted by $S(p_{n-1}, n)$, where p_{n-1} is an identifier for the group in the $n-1$ subsets. Iterative applying leads to a successive groups' reduction.

To identify the optimal union the objective function has to be evaluated for all $n(n-1)/2$ possible unions in the first step. As $Z[a, b] = Z[b, a]$ only those are considered with the first index smaller then the second. The both groups with the lowest value $Z[p_{n-1}, q_{n-1}, n-1]$ (with $p_{n-1} < q_{n-1}$ used as indices) are then grouped together. However, another approach measuring the distance $d(\cdot, \cdot)$ between two subsets would be to evaluate only the increase in SSE when joining them. This is easier to compute and here given in the case for two subsets G_p (indexed by p_{n-1}), G_q (indexed by q_{n-1}) containing n_p and n_q elements with means $\bar{x}_p$ and $\bar{x}_q$ respectively[2]:

$$Z[p_{n-1}, q_{n-1}, n-1] = d(G_p, G_q) = \frac{n_p\, n_q}{n_p + n_q} \|\bar{x}_p - \bar{x}_q\|^2 \quad {}^{3} \tag{2.5}$$

After identifying the *optimal* union the new subset is formed:

$$S(p_{n-1}, n-1) = S(p_{n-1}, n) \cup S(q_{n-1}, n) \ ,$$

All other subsets remain unaffected besides changing the index of the iterative step, i.e. $S(i, n-1) = S(i, n)$ with $i \neq p_{n-1}$ and $i \neq q_{n-1}$. The group with the index q_{n-1} is then deleted.

In the next iteration step the objective function is now evaluated for all $(n-1)(n-2)/2$ possible unions. However, in this step the distances between the group merged in the previous step and the others have to be evaluated. The new distance between two groups can be calculated with distances obtained in the previous step[4]. Again the groups with the lowest value are united, giving the next optimal set of subsets:

$$S(p_{n-2}, n-2) = S(p_{n-2}, n-1) \cup S(q_{n-2}, n-1) \ ,$$

with $p_{n-2} < q_{n-2}$ and for the rest $S(i, n-2) = S(i, n-1)$ with $i \neq p_{n-2}, i \neq q_{n-2}$ and $i \neq q_{n-1}$.

In the k–th iteration step the objective function has to be evaluated for $(n-k+1)(n-k)/2$ unions. The optimal value is then $Z[p_{n-k}, q_{n-k}, n-k]$ leading to a union

$$S(p_{n-k}, n-k) = S(p_{n-k}, n-k+1) \cup S(q_{n-k}, n-k+1) \ .$$

A flowchart of the complete algorithm can be found in the Appendix A (Figure A) on page 25.

[2]The proof for the univariate case can be found in the Appendix B on page 26
[3]Fahrmeir et al. (1996), page 466
[4]A proof can be found in the Appendix B on page 26

Now, after this excursus, the adaption of this algorithm fulfilling the restrictions of K–anonymity will be explained.

First of all the objective function in Ward's algorithm is the increase in SSE measuring the distance between to subsets (see (2.5)). Secondly, the above presented proceeding allows subsets with less than k and more than $2k$ elements, even in late steps. To avoid this, there has to be a stricter rule for the process of uniting subsets.

Algorithm 1 (k–**Ward**)

---

1. *Form a group with the first (smallest) k elements of the data set and another group with the last (largest) k elements of the data set.*
2. *Use Wards method until all elements in the data set belong to a group containing k or more data elements; in the process of forming groups by Wards method, never join two groups which have both a size greater than or equal to k.*
3. *For each group in the final partition that contains $2k$ or more data elements, apply this algorithm recursively (the data set to be considered is now restricted to the particular group containing $2k$ or more elements).*

Proposition 1. *The k–Ward–Algorithm terminates after a finite number of recursion steps.*

Proof.
The first step ensures that in each recursion step the active data set is split into at least 2 groups, so the obtained partition is always *finer* than the initial. So if there are only 2 groups formed in each recursion step it needs at most $\lceil \frac{n}{2k} \rceil$ turns. The rule in the second step ensures that the in step 1 formed groups are never united because of their size. At least the last step guarantees K–anonymity by forcing a further recursion step for all groups with $2k$ or more elements. $\square$

The only difficulty appears in the determination of the k *smallest* and *largest* elements of the data set. One way could be to project the data onto a single-axis and then those are easily obtained. Another approach could be to use any distance measure, for example the Euclidean distance, in order to find the most distant elements and then group the nearest around them, according to the distant measure. However, there may be different results in the grouping process '*depend[ing] on which extreme point is taken as first*'[5].
As there may arise misunderstandings concerning the second step of the algorithm, the underlying idea is now pointed out. Before the first step the data set contains of n single element groups, i.e each observation is treated as a group. In the first step the most distance observations are discovered and then the k nearest observations are grouped around them. Thus after the first step there are $n - 2k + 2$ groups. These groups are the basis of Ward's algorithm in the

[5]Mateo-Sanz and Domingo-Ferrer (1998), page 520

second step. The underlying recursion of Ward's algorithm is applied until a k-partition is obtained. To ensure that after the second step the data set is partitioned into at least 2 groups the rule is not to merge two groups containing more or equal than k elements. The rule has to be applied since it is possible to merge the single–element groups continuously into one margin group. The rule in the second step allows thereby groups with more than $2k$ elements after the union. For each of those groups the algorithm has to be applied recursively with the concerning group as data basis.

2.3 Disclosure Risk

Microaggregation requires a lot of resources, mostly computing time, so the results should satisfy a certain level of anonymity. This seems to be guaranteed by the property of K–anonymity, although some examples can be constructed holding the property of K–anonymity, but are easily disclosurable. The risk of identifying single observations from the microaggregated data set is called *disclosure risk.* Consequently the key purpose of any microaggregation strategy should decrease this risk.
This also applies for k–Ward–microaggregation, so it has to match this property. As the strategy in k–Ward–microaggregation is to unite homogeneous groups, the disclosure risk decreases with the inhomogeneity of the data set according to the used variables. The maximal disclosure risk is when the original data set is already a k–partition and the values in each group do not differ. In this case there is no difference between the aggregated data set and the original one. Generally a tendency between disclosure risk and heterogeneity of the data set can be postulated, as an increase in heterogeneity implies an decrease of disclosure risk.

To use the k–Ward–Algorithm even in the homogeneity case, Li et al. (2002) developed a *secure–k–Ward–Algorithm.* The underlying idea is to compare the original values with its substitutes and in case their difference is too small, it is looked for a substitute that hold a given distance. In order to analyze the aggregated data set for critical substitution values they defined the *tolerance level* ε and the *security ratio* γ. With the tolerance level the minimal tolerable difference between the original and the substitution value defined, and the security ratio gives the percentage of observations with their differences greater than the tolerance level. Whereas the tolerance level has to be predefined by the user and applies for all groups, the security ratio has to be computed for every group. In the following the secure–k–Ward–Algorithm is presented[6].

[6]Li et al. (2002), page 153

Algorithm 2 (**secure–k–Ward**)

1 Initial phase: *Apply k–Ward to the input data set. Assume that the data set is partitioned into g groups with the i-th group consisting of n_i individuals $x_{ij}(j = 1, \dots, n_i)$, where $n_i \geq k$ and $\sum_{i=1}^{g} n_i = n$. Denote the average over the i-th group $\bar{x}_i(i = 1, \dots, g)$.*

2 Checking phase: *For each group i, compute the security ratio γ based on the data and the tolerance level ε. If $\gamma \geq \gamma_0$, then use the average x_i to substitute the individual values in this group and continue this checking phase for the next group; else, deliver the current group i to the intra–group optimization phase.*

3 Intra–group optimization phase: *Compute two optimal values $\overleftarrow{x_i}$ and $\overrightarrow{x_i}$ for the current group i and use them respectively instead of x_i in substitution of individuals in group i such that the information loss of substitution in this group is minimized with the constraints $\{\gamma \geq \gamma_0 \wedge \overleftarrow{x_i} \leq \bar{x}_i\}$ and $\{\gamma \geq \gamma_0 \wedge \overrightarrow{x_i} \geq \bar{x}_i\}$ respectively. Then go back to the checking phase for the next group.*

4 Inter–group optimization phase: *For each group j that has been optimized by phase 3, determine either $\overleftarrow{x_i}$ or $\overrightarrow{x_i}$ to be $\hat{x}_j$ and use $\hat{x}_j$ in substitution of values in this group such that the average over all substituted values of all groups is as close as possible to the original average $\bar{x}_i$.*

In their article Li et al. (2002) proposed to use any optimization method for the intra–group optimization phase[7] and developed a heuristic method for the inter–group optimization phase, as this optimization proves to be NP–hard[8]. The great advantage of this algorithm is the compatibility with k–Ward–Algorithm; '*that is, if the K-partition finished by k–Ward*[–Algorithm] *has no security problem in terms of the security ratio, then secure-k–Ward*[–Algorithm] *works exactly the same way as k–Ward*[–Algorithm]. *In this case, secure-k–Ward*[–Algorithm] *terminates once the checking phase is finished.*'[9]

However, the output of the secure–k–Ward-Algorithm relies on the predefined tolerance level ε and minimum security ratio γ_0. Varying these constraints can lead to quite different aggregated data sets, nevertheless the result is to be considered safe concerning the disclosure risk.

Besides the issue of disclosure control, the computing time is another aspect to mention. The article 'On Optimizing the k–Ward Micro-aggregation Technique for Secure Statistical Databases' by Fayyoumi and Oommen[10] describes how to save computing time by pre–partitioning the data set. As this aspect is not as important as disclosure control, it is not further evaluated in this thesis.

[7]cf. Li et al. (2002), page 154f
[8]cf. Li et al. (2002), page 155f
[9]Li et al. (2002), page 154
[10]Fayyoumi and Oommen (2006)

Chapter 3

The Effect of k–Ward–Algorithm on the Estimation of a Linear Regression Model

This chapter is dealing with the effect of microaggregated data — generated by using k–Ward–Algorithm — on the estimators of a linear model. This will be accomplished by a specialization of the general conclusions made by Schmid (2007, chap. 4) in case of fixed–size microaggregation. In the microaggregation process the data are sorted, leading to two different approaches in analyzing a linear model: The sorting relies either on a function of the regressors or an arbitrary variable.
As k–Ward–Algorithm sorts the data set implicitly, the concept of single–axis–sorting is adapted, assuming that the group forming process is done according to one latent sorting variable.

3.1 The sorting variable is a function of the regressors

In this case, all variables which were microaggregated by the k–Ward–Algorithm are in the linear model as regressors. Therefore the multivariate k–Ward–Algorithm had to be used.
Let G denoted the number of disjoint groups formed in this process and G_j the j–th group containing k_j elements but at least k, $j = 1, \dots r$. The data contains n observations, but it is not necessary that n is a multiple of k. The microaggregated values are written with a tilde over the variable name, i.e. the microaggregated values for x are denoted as $\tilde{x}$.
The classical linear model with its assumptions for mean and variance of the error variable is used:

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \tag{3.1}$$

$$\mathbb{E}\left(\varepsilon|X_1,\ldots,X_p\right) = 0 \tag{3.2}$$
$$\mathbb{V}\left(\varepsilon|X_1,\ldots,X_p\right) = \sigma_\varepsilon^2 \tag{3.3}$$

The estimator for β on the original data is denoted with $\hat{b}$.

According to the results of Schmid (2007, chap. 4.2), it is possible to generate a matrix D describing the structure of the microaggregation process. This has to be adapted hence he assumed fixed–size microaggregation. Then the microaggregated values can be written as [1]:

$$\tilde{X_1} = D \cdot X_1 \,, \tag{3.4}$$
$$\vdots \tag{3.5}$$
$$\tilde{X_p} = D \cdot X_p \tag{3.6}$$

Remark: The response variable Y is not aggregated in this process.
The microaggregation matrix D has the following structure:

$$D = \Pi^{-1} \cdot K \cdot \begin{pmatrix} 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \underbrace{0 \ \cdots \ 0}_{k_1} & & & \cdots & \underbrace{1 \ \cdots \ 1}_{k_r} & & \end{pmatrix} \cdot \Pi \tag{3.7}$$

with the diagonal matrix K as

$$K = diag\underbrace{\left(\underbrace{\frac{1}{k_1}, \ \ldots \ , \frac{1}{k_1}}_{k_1}, \frac{1}{k_2}, \ \ldots \ , \frac{1}{k_{r-1}}, \underbrace{\frac{1}{k_r}, \ \ldots \ , \frac{1}{k_r}}_{k_r}\right)}_{n} \tag{3.8}$$

Hereby Π has the function of ordering the data according to the result of the k-Ward-Algorithm, whereas the 2 matrices in the middle perform the actual aggregation. Finally by multiplying it with Π^{-1} the data is rearranged in their initial order. Easily to be seen is that D is symmetric and idempotent[2].

Theorem 1. *Let $X = (X_1,\ldots,X_p)'$ k–Ward–microaggregated data, and a linear regression according to* (3.1) *with assumptions* (3.2), (3.3). *Denote by $\tilde{\mathbf{X}} := (1, \tilde{x}_1, \ldots, \tilde{x}_p)$ the design matrix of the aggregated data. If $(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}$ exists, then $\tilde{b} := (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'y$ is an unbiased LS estimator of β.*

[1]cf. Schmid (2007), page 94

[2]It is proven in the Appendix B on page 28f

Proof.
$\tilde{b}$ can be written with the original data.

$$\begin{aligned}\tilde{b} &= (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'y && (3.9)\\ &= (\mathbf{X}'D'D\mathbf{X})^{-1}\mathbf{X}'D'y && (3.10)\\ &= (\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy && (3.11)\end{aligned}$$

The vector containing the actual values of ε by e. So the mean of $\tilde{b}$ becomes

$$\begin{aligned}\mathbb{E}\left(\tilde{b}|X\right) &= (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\mathbb{E}\left(y|X\right)\\ &= (\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\left(\mathbf{X}\beta + \mathbb{E}\left(e|X\right)\right)\\ &= (\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\mathbf{X}\beta + (\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D0_n\\ &= \beta\end{aligned} \tag{3.12}$$

□

The proof of this theorem follows mostly Schmid (2007) argumentation[3].

Normally, not only the value of the estimators but also their variance is of interest. It appears that microaggregation comes along with a loss of efficiency, implying a greater variance of $\tilde{b}$. This was derived in the case of single–axis–sorting and fixed–size microaggregation by Lechner and Pohlmeier (2003)[4].

Theorem 2. *Let $X = (X_1, \ldots, X_p)'$ k–Ward–microaggregated data, and a linear regression according to* (3.1) *with assumptions* (3.2), (3.3). *Denote by* $\operatorname{Cov}\left(\tilde{b}|X\right)$ *and* $\operatorname{Cov}\left(\hat{b}|X\right)$ *the covariance matrices of $\tilde{b}$ and $\hat{b}$ given $X_1, \ldots, X_p$, respectively. Then $\tilde{b}$ has a greater variance than $\hat{b}$, in the sense that the matrix* $\operatorname{Cov}\left(\tilde{b}|X\right) - \operatorname{Cov}\left(\hat{b}|X\right)$ *is positive semidefinite.*

Proof.
$\operatorname{Cov}\left(\tilde{b}|X\right) - \operatorname{Cov}\left(\hat{b}|X\right)$ is positive semidefinite if and only if $\left(\operatorname{Cov}\left(\tilde{b}|X\right)\right)^{-1} - \left(\operatorname{Cov}\left(\hat{b}|X\right)\right)^{-1}$ is positive semidefinite. With $\operatorname{Cov}\left(\hat{b}|X\right) = \sigma_\varepsilon^2\left(\mathbf{X}'\mathbf{X}\right)$ and $\operatorname{Cov}\left(\tilde{b}|X\right) = \sigma_\varepsilon^2\left(\mathbf{X}'D\mathbf{X}\right)$, it follows:

$$\begin{aligned}\left(\operatorname{Cov}\left(\tilde{b}|X\right)\right)^{-1} - \left(\operatorname{Cov}\left(\hat{b}|X\right)\right)^{-1} &= \frac{1}{\sigma_\varepsilon^2}\left(\mathbf{X}'\mathbf{X} - \mathbf{X}'D\mathbf{X}\right)\\ &= \frac{1}{\sigma_\varepsilon^2}\mathbf{X}'\left(I_n - D\right)\mathbf{X}.\end{aligned} \tag{3.13}$$

As $(I_n - D)$ is an idempotent and symmetric matrix, it follows for any vector $v/neq0$

$$v'\mathbf{X}'\left(I_n - D\right)\mathbf{X}v = v'\mathbf{X}'\left(I_n - D\right)\left(I_n - D\right)\mathbf{X}v = w'w \leq 0, \tag{3.14}$$

where $w := \left(I_n - D\right)\mathbf{X}v$. □

[3]cf. Schmid (2007), page 95
[4]Lechner and Pohlmeier (2003), pages 3, 27

Concerning an estimator for σ_ε^2, it is possible to obtain a upper bound for an unbiased estimator.

Theorem 3. *Assume the sorting variable to be a function of $X_1, \ldots, X_p$. Denote by $\tilde{\mathbf{X}} := (1, \tilde{x}_1, \ldots, \tilde{x}_p)$ the design matrix of the aggregated data and by $\tilde{\varepsilon}'\tilde{\varepsilon} := (y - \tilde{\mathbf{X}}\tilde{b})'(y - \tilde{\mathbf{X}}\tilde{b})$ the estimator of the residual sum of squares based on the aggregated data. Then $\tilde{\varepsilon}'\tilde{\varepsilon}/(n-p-1)$ is the upper bound for an unbiased estimator of σ_ε^2.*

Proof.
The proof follows the same steps as made in Schmid (2007).

$$\begin{aligned}
\tilde{\varepsilon}'\tilde{\varepsilon} &= (\tilde{y} - \tilde{\mathbf{X}}\tilde{b})'(\tilde{y} - \tilde{\mathbf{X}}\tilde{b}) \\
&= y'y - \left((\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy\right)'\mathbf{X}'Dy - y'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy \\
&\quad + (\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'DD\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy \\
&= y'y - y'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy - y'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy \\
&\quad + y'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'Dy \\
&= y'(I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D)y \\
&= y'Qy
\end{aligned} \tag{3.15}$$

where $Q := I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D$. It is easy to see that Q is a symmetric and idempotent matrix[5], and thus

$$\begin{aligned}
\tilde{\varepsilon}'\tilde{\varepsilon} &= y'Qy \\
&= (\mathbf{X}\beta + \varepsilon)'(I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D)(\mathbf{X}\beta + \varepsilon) \\
&= (\beta'\mathbf{X}' - \beta'\mathbf{X}'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D + \varepsilon' - \varepsilon'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D)(\mathbf{X}\beta + \varepsilon) \\
&= \beta'\mathbf{X}'\mathbf{X}\beta - \beta'\mathbf{X}'D\mathbf{X}\beta + \varepsilon'\mathbf{X}\beta - \varepsilon'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\mathbf{X}\beta \\
&\quad + \beta'\mathbf{X}'\varepsilon - \beta'\mathbf{X}'D\varepsilon + \varepsilon'\varepsilon - \varepsilon'D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\varepsilon \\
&= \beta'\mathbf{X}'(I_n - D)\mathbf{X}\beta + \varepsilon'\mathbf{X}\beta - \varepsilon'D\mathbf{X}\beta + \beta'\mathbf{X}'\varepsilon - \beta'\mathbf{X}'D\varepsilon \\
&\quad + \varepsilon'(I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D)\varepsilon \\
&= (\mathbf{X}\beta(I_n - D))'(\mathbf{X}\beta(I_n - D)) + \varepsilon'\mathbf{X}\beta - \varepsilon'D\mathbf{X}\beta + \beta'\mathbf{X}'\varepsilon - \beta'\mathbf{X}'D\varepsilon \\
&\quad + \varepsilon'Q\varepsilon \\
&= (\mathbf{X}\beta(I_n - D))'(\mathbf{X}\beta(I_n - D)) + \varepsilon'(\mathbf{X}\beta - D\mathbf{X}\beta) + (\beta'\mathbf{X}' - \beta'\mathbf{X}'D)\varepsilon + \operatorname{tr}(Q(\varepsilon\varepsilon')).
\end{aligned} \tag{3.16}$$

As

$$\begin{aligned}
\operatorname{tr}(Q) &= \operatorname{tr}\left(I - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1} \cdot \mathbf{X}'D\right) \\
&= \operatorname{tr}(I) - \operatorname{tr}\left(\mathbf{X}'D \cdot D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\right) \\
&= \operatorname{tr}(I) - \operatorname{tr}(\mathrm{I}_{p+1}) \\
&= n - (p+1)
\end{aligned} \tag{3.17}$$

[5]It is shown in the Appendix B on page 29

it is obtained with (3.16) that

$$
\begin{aligned}
\mathbb{E}\left(\tilde{\varepsilon}'\tilde{\varepsilon}\right) = & \underbrace{\mathbb{E}\left(\left(\mathbf{X}\beta(I_n - D)\right)'\left(\mathbf{X}\beta(I_n - D)\right)\right)}_{\geq 0} + \underbrace{\mathbb{E}\left(\varepsilon'(\mathbf{X}\beta - D\mathbf{X}\beta)\right)}_{=0} \\
& + \underbrace{\mathbb{E}\left((\beta'\mathbf{X}' - \beta'\mathbf{X}'D)\varepsilon\right)}_{=0} + \mathbb{E}\left(\operatorname{tr}\left(Q\left(\varepsilon\varepsilon'\right)\right)\right) \\
& \geq \sigma_{\varepsilon}^2 \operatorname{tr}(Q) \\
& \geq \sigma_{\varepsilon}^2 (n - p - 1).
\end{aligned}
\tag{3.18}
$$

□

This result provides a measure of the maximal residual variance. In the special case that both the original design matrix $\mathbf{X}$ and the value of β is known, it is possible to derive an unbiased estimator for σ_{ε}^2.

However, an unbiased estimator for σ_{ε}^2 can be developed if the response variable is aggregated concomitantly, meaning that it is not used in the forming and sorting process of the k–Ward–Algorithm.

Theorem 4. *Let $X = (X_1, \ldots, X_p)$ be microaggregated by k–Ward–Algorithm and y concomitantly aggregated but not employed in the group forming process and denote the design matrix of the aggregated data by $\tilde{\mathbf{X}} := (1, \tilde{x}_1, \ldots, \tilde{x}_p)$ and $\tilde{\varepsilon}'\tilde{\varepsilon} := (y - \tilde{\mathbf{X}}\tilde{b})'(y - \tilde{\mathbf{X}}\tilde{b})$ the estimator of the residual sum of squares based on the aggregated data, then $\tilde{\varepsilon}'\tilde{\varepsilon}/(G-p-1)$ is an unbiased estimator of σ_{ε}^2, whereas G is the number of groups, formed by the algorithm.*

The proof follows mostly the steps of the previous, but with the stronger assumption it becomes less complicated.

Proof.

$$
\begin{aligned}
\tilde{\varepsilon}'\tilde{\varepsilon} &= (\tilde{y} - \tilde{\mathbf{X}}\tilde{b})'(\tilde{y} - \tilde{\mathbf{X}}\tilde{b}) \\
&= y'(D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D)y \\
&= y'Q^*y
\end{aligned}
\tag{3.19}
$$

where $Q^* := D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D$. Q^* is a symmetric and idempotent matrix. Moreover, $Q^*\mathbf{X} = 0$ [6], and thus

$$
\begin{aligned}
\tilde{\varepsilon}'\tilde{\varepsilon} &= y'Q^*y \\
&= (\mathbf{X}\beta + \varepsilon)'Q^*Q^*(\mathbf{X}\beta + \varepsilon) \\
&= (\mathbf{X}\beta + \varepsilon)'Q^{*\prime}(Q^*\mathbf{X}\beta + Q^*\varepsilon) \\
&= (\mathbf{X}\beta + \varepsilon)'Q^{*\prime}Q^*\varepsilon \\
&= (\mathbf{X}\beta)'Q^{*\prime}Q^*\varepsilon + \varepsilon'Q^{*\prime}Q^*\varepsilon \\
&= \beta'\mathbf{X}'Q^{*\prime}Q^*\varepsilon + \varepsilon'Q^*Q^*\varepsilon \\
&= \beta'(Q^*\mathbf{X})'Q^*\varepsilon + \varepsilon'Q^*\varepsilon \\
&= \varepsilon'Q^*\varepsilon \\
&= \operatorname{tr}\left(Q^*\left(\varepsilon\varepsilon'\right)\right).
\end{aligned}
\tag{3.20}
$$

[6] It is shown in the Appendix B on page 29

As

$$\begin{aligned}\mathrm{tr}\,(Q^*) &= \mathrm{tr}\left(D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\cdot\mathbf{X}'D\right)\\ &= \mathrm{tr}\,(D) - \mathrm{tr}\left(\mathbf{X}'D\cdot D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\right)\\ &= \mathrm{tr}\,(D) - \mathrm{tr}\,(\mathrm{I}_{p+1})\\ &= G-(p+1)\end{aligned} \tag{3.21}$$

it is obtained with (3.20) that

$$\begin{aligned}\mathbb{E}\,(\tilde{\varepsilon}'\tilde{\varepsilon}) &= \sigma_\varepsilon^2 \mathrm{tr}\,(Q^*)\\ &= \sigma_\varepsilon^2\,(G-p-1)\,.\end{aligned} \tag{3.22}$$

□

This results provides useful information: First of all it is possible to obtain an unbiased estimator of the true residual error variance σ_ε^2 by multiplying the "naive" variance estimator

$$\tilde{s}_\varepsilon^2 := \frac{1}{n-p-1}(\tilde{y}-\tilde{\mathbf{X}}\tilde{b})'(\tilde{y}-\tilde{\mathbf{X}}\tilde{b}) \tag{3.23}$$

with $(n-p-1)/(G-p-1)$.
Secondly, by not knowing of any microaggregation and thus using the naive variance estimator of σ_ε^2 one would overestimate the true error variance.

3.2 The sorting variable is an artificial and arbitrary variable

In this section the consequences of k–Ward–microaggregation on estimators are described, in case the concomitantly sorting of the data set is not only a function of the model variables, but relies on response, regressors and the error term. This assumption allows a generalization of the above results, however, in this part an asymptotic theory is considered. The sorting variable further denoted as H is a function of the model variable and an arbitrary term:[7]

$$H = c_yY + c_1X_1 + c_2X_2 + \ldots + c_pX_p + \varphi \quad . \tag{3.24}$$

The vector $c = (c_y, c_1, \ldots, c_p)$ determines the coefficients, i.e the weights, for each variable on H whereas φ is an model arbitrary error term, agglomerating the effect of the not included variables. This general construct can be used for the case of k–Ward–microaggregation, whereas all coefficients do not equal 0, because all variables in the dataset are used for the microaggregation process. However, it is not clear how to derive the coefficients, as the grouping and sorting is done implicitly. Nevertheless, it is clear that the result of the previous chapter cannot be used, as the value of response depends also on the value of the regressors and even more on excluded variables.
As to the assumptions concerning the linear regression model the original coefficient vector $\beta^* = (\beta_0, \beta_1, \ldots, \beta_p)'$ is redefined to the slope coefficients $\beta = (\beta_1, \ldots, \beta_p)'$. After a consistent estimation of β, it becomes obvious how to

[7]cf. Schmid (2007), page 98

derive β_0 and σ_ε^2. Since this section aims at an asymptotic theory the assumptions of $\mathbb{E}\left(\varepsilon|X_1,\ldots,X_p\right)=0$ and $\mathbb{V}\left(\varepsilon|X_1,\ldots,X_p\right)=\sigma_\varepsilon^2$ are weakened to the uncorrelation between $(X_1,\ldots,X_p)$ and ε.
Thus an estimator for the slope coefficients can be obtained by

$$\tilde{b}=\tilde{S}_{xx}^{-1}\tilde{s}_{xy} \tag{3.25}$$

To explore how and if it is possible to obtain consistent estimators the following lemma is needed[8]. The inverse regressions are needed to correct the correlations between the response and the regressors over the sorting variable H.

Lemma 1. *Consider the set of inverse linear regressions*

$$X_i=\alpha_i+\gamma_i H+\delta_i\ , \qquad i=1,\ldots,p \tag{3.26}$$

$$Y=\alpha_y+\gamma_y H+\delta_y\ , \tag{3.27}$$

with $\mathbb{E}\left(\delta_i\right)=0,\ i=y,1,\ldots,p,$ *and assume* $(\delta_y,\delta_1,\ldots,\delta_p)$ *to be independent of* H. *Then the following probability limits exist:*

a) $\operatorname*{plim}_{n\to\infty}\tilde{s}_{hh}=\sigma_{hh}$

b) $\operatorname*{plim}_{n\to\infty}\tilde{s}_{xh}=\sigma_{xh}$

c) $\operatorname*{plim}_{n\to\infty}\tilde{s}_{yh}=\sigma_{yh}$

d) $\operatorname*{plim}_{n\to\infty}\tilde{S}_{xx}=\tilde{\Sigma}_{xx}$

$$\left(1-\frac{1}{2k-1}\right)\frac{\sigma_{xh}\sigma_{xh}'}{\sigma_{hh}}+\frac{1}{2k-1}\Sigma_{xx}\leq\tilde{\Sigma}_{xx}\leq\left(1-\frac{1}{k}\right)\frac{\sigma_{xh}\sigma_{xh}'}{\sigma_{hh}}+\frac{1}{k}\Sigma_{xx}$$

e) $\operatorname*{plim}_{n\to\infty}\tilde{s}_{xy}=\tilde{\sigma}_{xy}$

$$\left(1-\frac{1}{2k-1}\right)\frac{\sigma_{yh}}{\sigma_{hh}}\sigma_{xh}+\frac{1}{2k-1}\sigma_{xy}\leq\tilde{\sigma}_{xy}\leq\left(1-\frac{1}{k}\right)\frac{\sigma_{yh}}{\sigma_{hh}}\sigma_{xh}+\frac{1}{k}\sigma_{xy}$$

Proof. See Appendix B, page 30 ff.

Concerning the parts d) and e) of the above lemma, which allows only the construction of bounds for the true covariance between the regressors and the response, then it is not obvious how to derive neither a consistent estimator nor bounds of such one. Hence the relations do not longer hold when multiplied in order to obtain the probability limit of $\tilde{b}$. As example for the upper bound with group–size $k_g\equiv k, g=1,\ldots,r$:

[8]cf. Schmid (2007), page 99f

$$\operatorname*{plim}_{n\to\infty} \tilde{S}_{xx} \leq \left(1-\frac{1}{k}\right)\frac{\sigma_{xh}\sigma'_{xh}}{\sigma_{hh}} + \frac{1}{k}\Sigma_{xx} \tag{3.28}$$

$$\operatorname*{plim}_{n\to\infty} \tilde{s}_{xy} \leq \left(1-\frac{1}{k}\right)\frac{\sigma_{yh}}{\sigma_{hh}}\sigma_{xh} + \frac{1}{k}\sigma_{xy} \tag{3.29}$$

$$\begin{aligned}\not\Rightarrow \operatorname*{plim}_{n\to\infty} \tilde{b} = \operatorname*{plim}_{n\to\infty} \tilde{S}_{xx}^{-1}\tilde{s}_{xy} \leq &\left(\left(1-\frac{1}{k}\right)\frac{\sigma_{xh}\sigma'_{xh}}{\sigma_{hh}} + \frac{1}{k}\Sigma_{xx}\right)^{-1}\cdot \\ &\cdot\left(\left(1-\frac{1}{k}\right)\frac{\sigma_{yh}}{\sigma_{hh}}\sigma_{xh} + \frac{1}{k}\sigma_{xy}\right)\end{aligned} \tag{3.30}$$

Since this approach does not lead to a bounded estimator, one might think of k as an random variable (denoted with upper case letter K), having a distribution function, and its expected value $\mathbb{E}(K)$. As in Lemma 1 e) and f) the probability limit is used for determination of the covariance matrix, the above is useful in evaluating a fixed estimator for the covariance matrices.

Lemma 2. *Under the assumptions made in Lemma 1 and considering the group–size k as random variable K with its expected value $\mathbb{E}(K) := \mu_k$, then the probability limits in Lemma 1 e) and f) can be written as*

e) $\operatorname*{plim}_{n\to\infty} \tilde{S}_{xx} = \tilde{\Sigma}_{xx} = \left(1-\frac{1}{\mu_K}\right)\frac{\sigma_{xh}\sigma'_{xh}}{\sigma_{hh}} + \frac{1}{\mu_K}\Sigma_{xx}$

e) $\operatorname*{plim}_{n\to\infty} \tilde{s}_{xy} = \tilde{\sigma}_{xy} = \left(1-\frac{1}{\mu_K}\right)\frac{\sigma_{yh}}{\sigma_{hh}}\sigma_{xh} + \frac{1}{\mu_K}\sigma_{xy}$

Proof. See Appendix B, page 33 ff.

Remark: As this uses an asymptotic theory, the estimator $\bar{k}$ can be used for $\mathbb{E}(K)$ due to the law of large numbers.
With Lemma 2 it should be possible to obtain a consistent estimator by strictly following Schmid's theorem 3 and 4[9]. However, as easily to obtain consistent estimators might seen, it bears one major problem: The artificial sorting variable can only be evaluated in some special cases. As the sorting in k–Ward–Algorithm is done implicitly, is difficult to extract a single sorting variable. Furthermore, with the two different distance measures, in the first step the Euclidean and in the second the increase in SSE, the criterion values for joining 2 groups cannot be used for such a variable even more as there are non–single element groups merged. The same problem arises in case recursion steps are needed, as the merging criterion's value is initially set to zero at each recursion start.

A naive sorting variable is after the data set is grouped to sort the groups and the within observations randomly. Then assign each observation in the ordered data set a number between 1 and n in the way that all observations in one group are connected and then taking the assignment as sorting variable.
The problem of this naive sorting variable is the fact that it does not provide an actual distance between observations as all *distances* are equidistant by

[9]see Schmid (2007), page 100ff

construction. Furthermore, the variance of this variable does not provide an information gain.

Another problem arises in determining the actual coefficients $c_y, c_1, \ldots, c_p$, as the effect of each variable in the process of grouping/sorting on an artificial sorting variable is not obvious. As the weighted squared distances between two observations are used, the sorting variable might even rely on squared variable impact as in

$$H = c_y Y^2 + c_1 X_1^2 + c_2 X_2^2 + \ldots + c_p X_p^2 + \varphi \quad .$$

However, finding an appropriate sorting variable this would lead to another research thesis or article.

As a conclusion of this chapter can be said that in case the model–response is not involved in the process of k–Ward–Algorithm, the estimators for the slope coefficients remain unbiased, though less precise, but the estimator of the error variance becomes greater. This can be bounded in case the response is not microaggregated, and exactly given if the response is sorted and microaggregated concomitantly.
The worse case is that the response is included in the microaggregation process. With the theory of an artificial sorting variable it is theoretically possible to show that under some assumptions it is possible to obtain an unbiased estimator. However, the actual evaluation remains tricky, as the optimal sorting variable has not been identified yet.

Chapter 4

Simulations

In order to check the theoretically developed propositions, various regression models have been evaluated. This was done with the software R[1], whereby the methods used for k–Ward microaggregation are done via the R–script *kward.r*. This routine uses the precompiled shared object *kward.so*, which is able to run on most 32–bit Unix machines. However, to elude any problems it should be compiled by the user, before executing the code[2].
The data of the first regressor was generated by the addition of normal distributions with both $\sigma^2 = 16$ and means -2 and 2. The second regressor data was drawn from an uniform distribution over $[-8, 8]$. The number of observations was 500 and the minimum group–size constraint 3. For simplicity reasons, the second slope coefficient was set fixed to 2 while the first was varying from -3 to 3 by steps of 0.1. The response was created as an addition of the regressor values and its slope coefficients and a normal distributed, zero–mean error term with a variance of 16. In order to explore the differences between a non/microaggregated response and a response, microaggregated but not used for group forming, two categories of linear models were chosen. For each of this category and beta coefficient 30 models were fitted. The resulting models can be described in the following way. The notation of chapter 3 is used, β_f means that it has been kept fixed, $\tilde{\tilde{y}}$ that y is only microaggregated but not involved in grouping process.

1. $y = \beta_1 \cdot \tilde{x}_1 + \beta_{2f} \cdot \tilde{x}_2 + \epsilon$
2. $\tilde{\tilde{y}} = \beta_1 \cdot \tilde{x}_1 + \beta_{2f} \cdot \tilde{x}_2 + \tilde{\epsilon}$

Figure 4.1 illustrates the differences between the original coefficients and its estimators for the 2 models concerning variable β_1.

[1] R Development Core Team (2008)

[2] This is done by navigating into the directory containing the file *kward.f* and then running *R CMD SHLIB kward.f* on the command line. For more details see R Development Core Team (2009*d*)

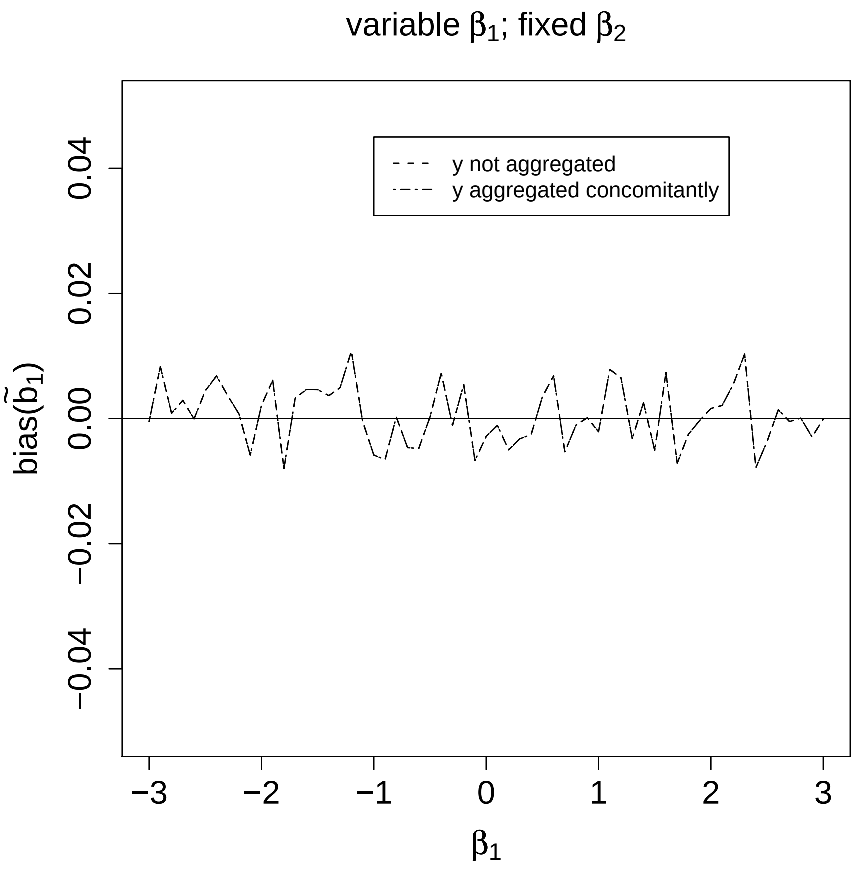


Figure 4.1: Bias of the slope estimator $beta_1$ in case the regressors are used in grouping process

As it can be seen, there is no difference in the estimator whether y is microaggregated concomitantly or not. This result was proposed in the first theorem. As the result of the estimators is the expected one, the figure 4.2 shows the results concerning the error variance. In the models, where the response was microaggregated concomitantly the estimated error variance is around 16 — the original error variance — and in the other case it is greater than 16. The quadratic form is due to the term $(\mathbf{X}\beta(I_n - D))' (\mathbf{X}\beta(I_n - D))$ which has to be subtracted of the naive variance estimator to obtain the unbiased one. The smaller the absolute value of β the smaller the difference between the estimated and the true error variance.

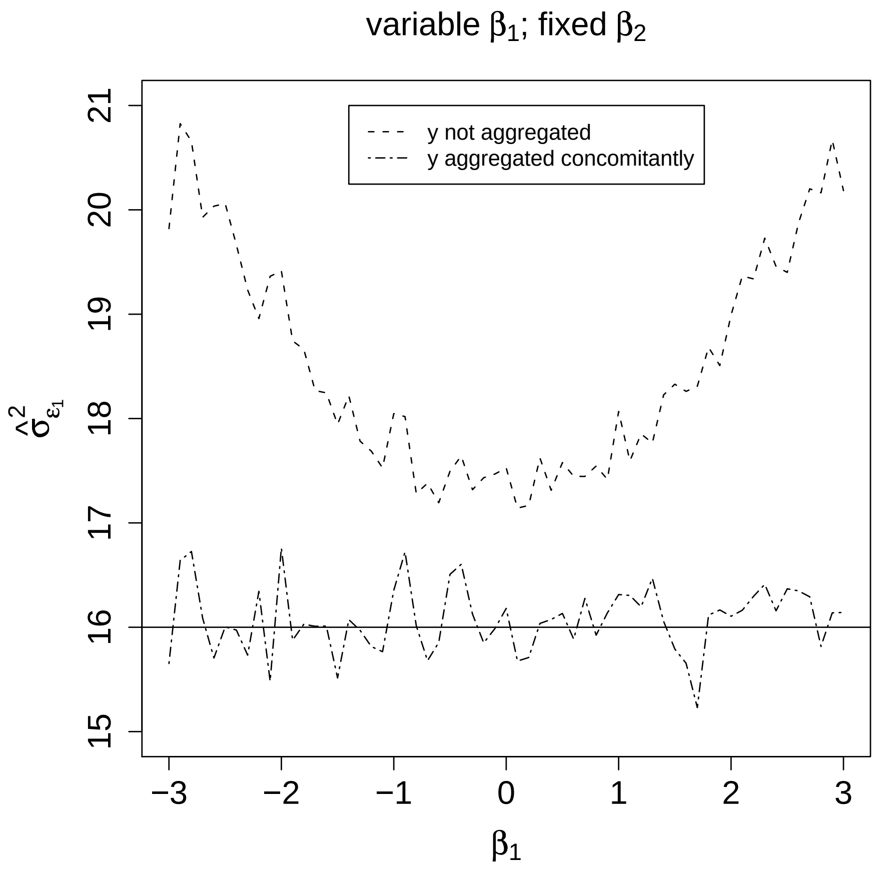


Figure 4.2: Variance of the error term in case the regressors are used in grouping process

The simulations concerning the microaggregation of the regressors have demonstrated the expected results. Although these simulations were done with a small number of repetitions and observations, the theorems 1 to 3 still hold, so a larger number of one or both will change the result only negligibly.

The simulations described below deal with the effect of k–Ward–microaggregation on the slope estimators of linear regression, in which the response is included in the grouping process. The generation of the data set is the same as in the previous simulations, but now both responses and regressors are k–Ward–microaggregated altogether, simulating that the sorting variable relies on the response, the regressors and a model–excluded variable. Here the varying β ranges also from -3 to 3 but in steps of 0.5 as the only purpose of this simulation is getting a general idea of the effect of k–Ward–microaggregation on the estimators.

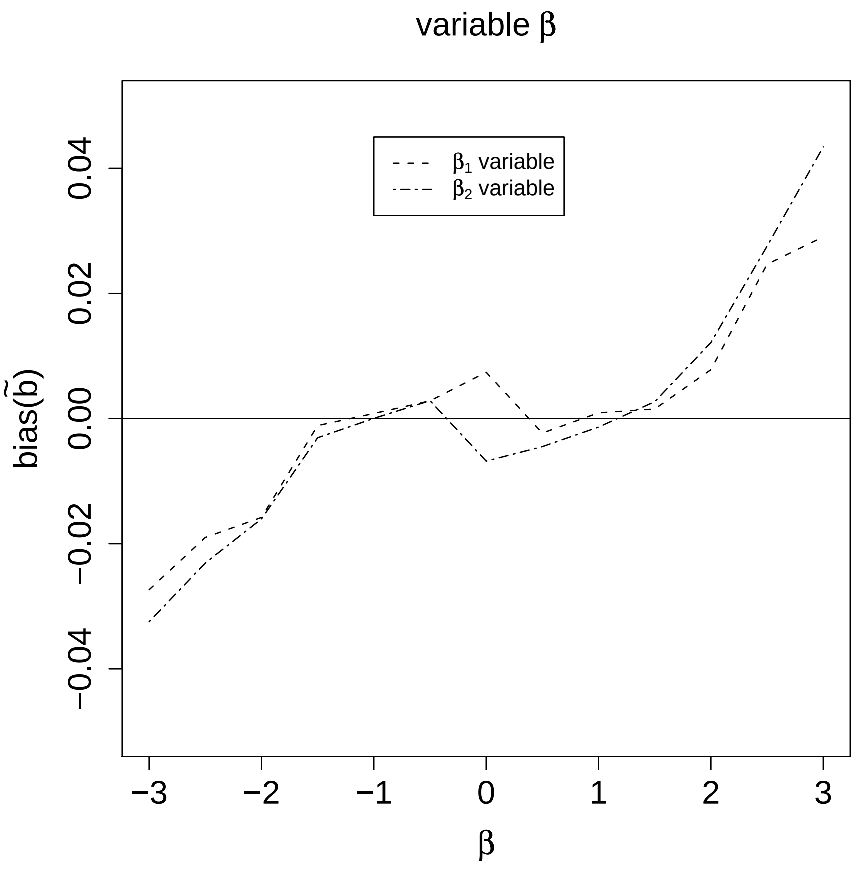
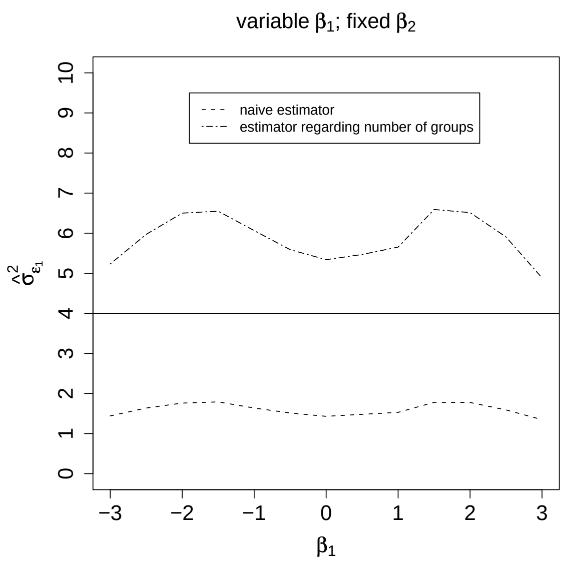


Figure 4.3: Bias of the slope estimators in case the response, regressors and a model–excluded term are used in grouping process

A tendency for both estimators can be explored in figure 4.3 as the greater the absolute value of beta, the greater the absolute value of the bias. Furthermore it tends that the bias has the same sign as β. The differences between the bias of the two estimators seems to be negligible. The same effect was noticed by Schmid (2007); Schmid et al. (2007), in their simulation study on normal distributed data, using single–axis–sorting and fixed–size microaggregation, in case the sorting variable was the response[3].

The estimators of the error variance the naive estimator (sum of squared residuals divided by $n-p-1$, here $n-3$) are compared with the obtained unbiased estimator of the previous situation (sum of squared residuals divided by $G-p-1$, here $G-3$) in figure 4.4. It appears that the naive estimator underestimates the error variance whereas the group–adjusted overestimates it. The fact that the naive underestimates the true error variance is clear, as the process of microaggregation absorbs parts of it by making the error terms more homogeneous.

[3]Schmid (2007), page 139f; Schmid et al. (2007), page 417



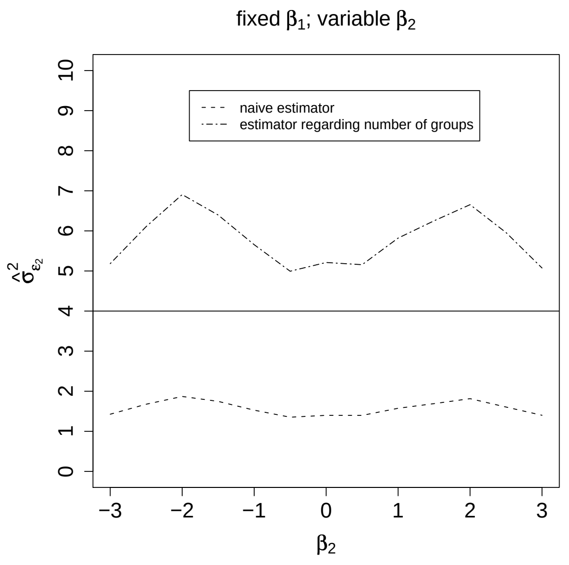


Figure 4.4: Variance of the error term in case the response, regressors and a model–excluded term are used in grouping process

The peaks at -2 and 2 may result from the first variable which was constructed by addition of normal distributions with means at -2 and 2. The naive estimator smooths this effect. If the group–adjusted constantly overestimates the underlying true error variance then this estimator proves to be an upper bound it. However, more simulations have to be made in order to empirically verify or falsify this statement.

Chapter 5

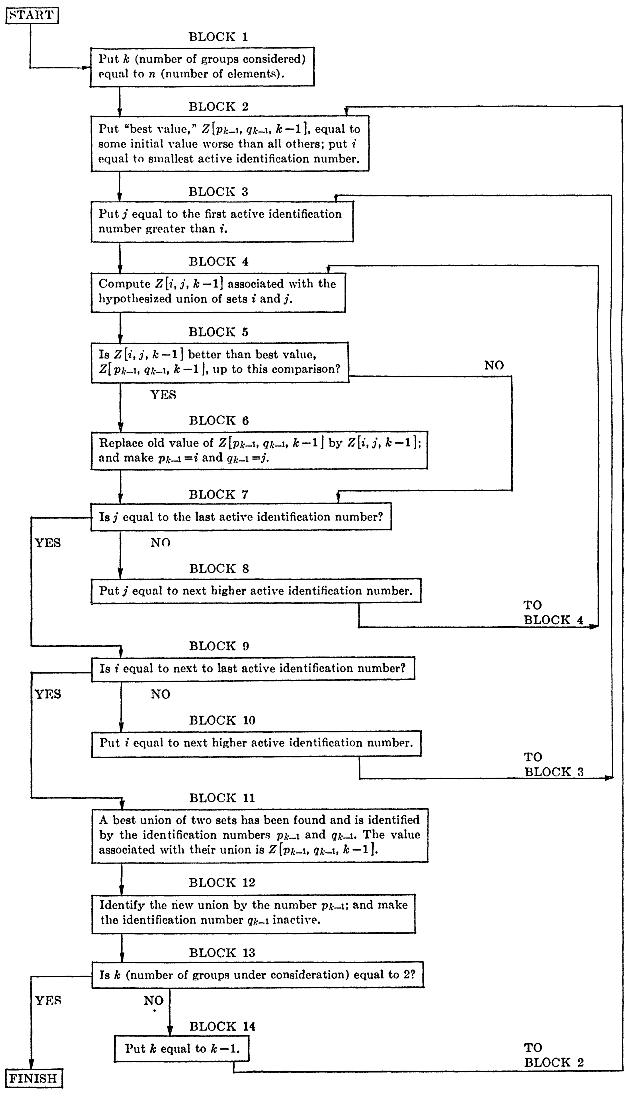
Conclusion and further Research

As demonstrated in the previous chapters, the case of only k–Ward–microaggregated regressors does not cause any difficulty as it is possible to obtain unbiased estimator for the intercept and slope coefficients easily. Furthermore, the underlying error variance can also be estimated unbiased in case the response is aggregated concomitantly. Therefore only the vector of the group–sizes for each observation is mandatory, which the data user should easily extract in case the data holder does not provide it. More challenging is the determination of the first assumption that the response is not used in the grouping process as the data user cannot examine this on its own.
If the regressors, the response and model–excluded variables are employed in the k–Ward–Algorithm, it is generally not thinkable to determine a consistent estimator. The results, described in the paragraph above, are hardly to be used here, as the aggregated values of the response and the regressors depend not only on their original values but also on model–excluded ones, and thus the error term. To specify these dependencies the idea of an latent sorting variable is assumed. Even with this assumption it is not conceiveable to obtain a consistent estimator as the covariances between the regressors itself and the response is not uniformly the same but rather depends on the considered group and its size. Thus in considering the minimal and maximal group–sizes an upper and lower bound of the covariance can be given. Nevertheless, with a more strict assumption, the group–size is a random variable with a finite support and an existing expected value, the covariances can be estimated consistently. That assumption is quite firm, especially for small data sets this is not given, as the group–sizes are not allowed to vary widely (*For example: if one observation belongs to a group with 4 others, then at least these four observation also have a value of the group–size of 5*).
More challenging is the exact specification of the latent sorting variable. A naive approach would be to sort the data set according to the groups an then assigning each observation of the sorting variable the number of the ordered one. It seems to be that it does not allow to interpret the differences in the values of the sorting variable and the explanation of the different group–sizes. In a more sophisticated approaches the values of the objective function in Ward's algorithm are used as

a sorting variable. This may work well in case no recursion step was needed, although it is not clear which values are to assign for the observation which were grouped in the first step of k–Ward–Algorithm. If a recursion step is applied it becomes complicated as the value of the objective function is always lower than the one of the last merge, as the within–group variance of the initial step (SSE), becomes now the total variance (SST). This side–effect collides with the idea of using the value of the objective function as a sorting variable.
This topic of generating a appropriate sorting variable needs further research as this is conditional for the results of the second section in the third chapter. If this proves to be beyond the bounds of possibility then a different idea to describe the dependence structure between the response, regressors and error term is needed.
Along with it goes the question of an unbiased or even bounded estimator of the error variance. However, this is only secondly of interest as it is only needed for testing the estimated parameters.
Most articles and theses focus on the aspect of generating secure and fast to compute microaggregated data sets. Only a few consider the scientific data users' needs of obtaining consistent and/or unbiased regression estimators for microaggregated data, however with the less complicated cases of fixed–size microaggregation.
Overall this thesis may be seen as fundamental research of this topic, which might give some impulses for further reserch.

Appendix A

Ward's clustering algorithm



START
BLOCK 1
Put k (number of groups considered) equal to n (number of elements).
BLOCK 2
Put "best value," $Z[p_{k-1}, q_{k-1}, k-1]$, equal to some initial value worse than all others; put i equal to smallest active identification number.
BLOCK 3
Put j equal to the first active identification number greater than i.
BLOCK 4
Compute $Z[i, j, k-1]$ associated with the hypothesized union of sets i and j.
BLOCK 5
Is $Z[i, j, k-1]$ better than best value, $Z[p_{k-1}, q_{k-1}, k-1]$, up to this comparison?
NO
YES
BLOCK 6
Replace old value of $Z[p_{k-1}, q_{k-1}, k-1]$ by $Z[i, j, k-1]$; and make $p_{k-1}=i$ and $q_{k-1}=j$.
BLOCK 7
Is j equal to the last active identification number?
YES
NO
BLOCK 8
Put j equal to next higher active identification number.
TO BLOCK 4
BLOCK 9
Is i equal to next to last active identification number?
YES
NO
BLOCK 10
Put i equal to next higher active identification number.
TO BLOCK 3
BLOCK 11
A best union of two sets has been found and is identified by the identification numbers p_{k-1} and q_{k-1}. The value associated with their union is $Z[p_{k-1}, q_{k-1}, k-1]$.
BLOCK 12
Identify the new union by the number p_{k-1}; and make the identification number q_{k-1} inactive.
BLOCK 13
Is k (number of groups under consideration) equal to 2?
YES
NO
BLOCK 14
Put k equal to $k-1$.
TO BLOCK 2
FINISH

Figure A.1: Flowchart for hierarchical grouping procedure (Ward (1963), p.242)

Appendix B

Proofs

Distant measure using difference in SSE before and after uniting groups (Page 5)

$$\begin{aligned}
d\left(G_1, G_2\right) &= SSE_{G_1 \cup\, G_2} - \left(SSE_{G_1} + SSE_{G_2}\right) \\
&= \sum_{k \in G_1 \cup G_2} \left(x_k\right)^2 - \left(n_1 + n_2\right) \overline{\left(n_1 \bar{x}_1 + n_2 \bar{x}_2\right)}^2 - \sum_{i \in G_1} \left(x_i\right)^2 + n_1 \bar{x}_1^2 - \sum_{j \in G_2} \left(x_j\right)^2 + n_2 \bar{x}_2^2
\end{aligned}$$

As G_1 and G_2 are disjoint groups, $\sum_{k \in G_1 \cup G_2} \left(x_k\right)^2 = \sum_{i \in G_1} \left(x_i\right)^2 + \sum_{j \in G_2} \left(x_j\right)^2$, thus

$$\begin{aligned}
&= -(n_1 + n_2) \frac{(n_1 \bar{x}_1 + n_2 \bar{x}_2)^2}{(n_1 + n_2)^2} + n_1 \bar{x}_1^2 + n_2 \bar{x}_2^2 \\
&= \frac{-1}{n_1 + n_2} \left(n_1^2 \bar{x}_1^2 + 2 n_1 n_2 \bar{x}_1 \bar{x}_2 + n_2^2 \bar{x}_2^2\right) + n_1 \bar{x}_1^2 + n_2 \bar{x}_2^2 \\
&= \frac{1}{n_1 + n_2} \left(-n_1^2 \bar{x}_1^2 - 2 n_1 n_2 \bar{x}_1 \bar{x}_2 - n_2^2 \bar{x}_2^2 + n_1^2 \bar{x}_1^2 + n_1 n_2 \bar{x}_1^2 + n_2^2 \bar{x}_2^2 + n_2 n_1 \bar{x}_2^2\right) \\
&= \frac{n_1 n_2}{n_1 + n_2} \left(\bar{x}_1^2 - 2 \bar{x}_1 \bar{x}_2 + barx_2^2\right) \\
&= \frac{n_1 n_2}{n_1 + n_2} \left(\bar{x}_1 - \bar{x}_2\right)^2
\end{aligned}$$

Distance measure for previously united and single group (Page 5)

For simplicity reason only the univariate case of x, y and z is considered, although the proof is straightforward for multivariate data. X, y and z are groups containing n_x, n_y and n_z elements, respectively. $n_k := n_x + n_y + n_z$

$$
\begin{aligned}
d\left((x \cup y), z\right) &= \frac{(n_x + n_y) \cdot n_z}{(n_x + n_y) + n_z} \left(\frac{n_x x + n_y y}{n_x + n_y} - z \right)^2 \\
&= \frac{(n_x + n_y) \cdot n_z}{n_x + n_y + n_z} \left(\frac{(n_x x + n_y y)^2}{(n_x + n_y)^2} - 2z \frac{n_x x + n_y y}{n_x + n_y} + z^2 \right) \\
&= \frac{1}{n_k} \left(\frac{n_z(n_x^2 x^2 + 2n_x n_y xy + n_y^2 y^2)}{n_x + n_y} - 2n_z z(n_x x + n_y y) + \frac{n_z (n_x + n_y)^2 z^2}{n_x + n_y} \right) \\
&= \frac{1}{n_k(n_x + n_y)} \big(n_x^2 n_z x^2 + 2n_x n_y n_z xy + n_y^2 n_z y^2 + (n_x + n_y)(-2n_x n_z xz - 2n_y n_z yz) \\
&\quad + n_x^2 n_z z^2 + 2n_x n_y n_z z^2 + n_y^2 n_z z^2 \big) \\
&= \frac{1}{n_k(n_x + n_y)} \big(n_x^2 n_z x^2 + 2n_x n_y n_z xy + n_y^2 n_z y^2 - 2n_x^2 n_z xz - 2n_x n_y n_z yz - 2n_x n_y n_z xz \\
&\quad - 2n_y^2 n_z yz + n_x^2 n_z z^2 + n_x n_y n_z z^2 + n_x n_y n_z z^2 + n_y^2 n_z z^2 \big) \\
&= \frac{1}{n_k(n_x + n_y)} \big(n_x^2 n_z x^2 + n_x n_y n_z x^2 - 2n_x^2 n_z xz - 2n_x n_y n_z xz + n_x^2 n_z z^2 + n_x n_y n_z z^2 \\
&\quad + n_x n_y n_z y^2 + n_y^2 n_z y^2 - 2n_x n_y n_z yz - 2n_y^2 n_z yz + n_x n_y n_z z^2 + n_y^2 n_z z^2 \\
&\quad + \left(-n_x n_y n_z x^2\right) + 2n_x n_y n_z xy + \left(-n_x n_y n_z y^2\right) \big) \\
&= \frac{1}{n_k} \left(\frac{(n_x + n_y)\left(n_x 2 n_z x^2 - 2n_x n_z xz + n_x n_z z^2\right)}{n_x + n_y} \right. \\
&\quad + \frac{(n_x + n_y)\left(n_y n_z y^2 - 2n_y n_z yz + n_y n_z z^2\right)}{n_x + n_y} \\
&\quad \left. - \frac{n_x n_y n_z}{n_x + n_y} \left(x^2 + 2xy + y^2\right) \right) \\
&= \frac{1}{n_k} \left(n_x n_z (x - z)^2 + n_y n_z (y - z)^2 - \frac{n_x n_y n_z}{n_x + n_y} (x - y)^2 \right) \\
&= \frac{1}{n_k} \left(\frac{n_x + n_z}{n_x + n_z} n_x n_z (x - z)^2 + \frac{n_y + n_z}{n_y + n_z} n_y n_z (y - z)^2 - n_z \frac{n_x n_y}{n_x + n_y} (x - y)^2 \right) \\
&= \frac{1}{n_k} \left((n_x + n_z) \frac{n_x n_z}{n_x + n_z} (x - z)^2 + (n_y + n_z) \frac{n_y n_z}{n_y + n_z} (y - z)^2 - n_z \frac{n_x n_y}{n_x + n_y} (x - y)^2 \right) \\
&= \frac{1}{n_x + n_y + n_z} \left((n_x + n_z) \cdot d(x, z) + (n_y + n_z) \cdot d(y, z) - n_z \cdot d(x, y) \right)
\end{aligned}
$$

D is idempotent (Page 10)

$DD =$

$$= \Pi^{-1}K \begin{pmatrix} 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix} \Pi \cdot \Pi^{-1}K \begin{pmatrix} 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix} \Pi$$

$$= \Pi^{-1}K \begin{pmatrix} 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix} K \begin{pmatrix} 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix} \Pi$$

$$= \Pi^{-1} \begin{pmatrix} \frac{1}{k_1} & \cdots & \frac{1}{k_1} & \cdots & 0 & \cdots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ \frac{1}{k_1} & \cdots & \frac{1}{k_1} & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & \frac{1}{k_r} & \cdots & \frac{1}{k_r} \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \cdots & 0 & \cdots & \frac{1}{k_r} & \cdots & \frac{1}{k_r} \end{pmatrix} \begin{pmatrix} \frac{1}{k_1} & \cdots & \frac{1}{k_1} & \cdots & 0 & \cdots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ \frac{1}{k_1} & \cdots & \frac{1}{k_1} & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & \frac{1}{k_r} & \cdots & \frac{1}{k_r} \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \cdots & 0 & \cdots & \frac{1}{k_r} & \cdots & \frac{1}{k_r} \end{pmatrix} \Pi$$

$$= \Pi^{-1} \begin{pmatrix} \frac{1}{k_1} & \cdots & \frac{1}{k_1} & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \frac{1}{k_1} & \cdots & \frac{1}{k_1} & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & \frac{1}{k_r} & \cdots & \frac{1}{k_r} \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & \frac{1}{k_r} & \cdots & \frac{1}{k_r} \end{pmatrix} \Pi$$

$$= \Pi^{-1} K \begin{pmatrix} 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix} \Pi$$

$$= D$$

Q is idempotent (Page 13)

$$\begin{aligned} QQ &= \left(I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\right)\left(I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\right) \\ &= I_nI_n - I_nD\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'DI_n \\ &\quad + D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'DD\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D \\ &= I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D \\ &\quad + D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D \\ &= I_n - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D = Q \end{aligned}$$

Q^* is idempotent (Page 13)

$$\begin{aligned} Q^*Q^* &= \left(D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\right)\left(D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\right) \\ &= DD - DD\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'DD + \\ &\quad + D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'DD\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D \\ &= D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D \\ &\quad + D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D \\ &= D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D = Q^* \end{aligned}$$

$Q^*\mathbf{X} = 0$ (Page 13)

$$\begin{aligned} Q^*\mathbf{X} &= \left(D - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\right)\mathbf{X} \\ &= D\mathbf{X} - D\mathbf{X}(\mathbf{X}'D\mathbf{X})^{-1}\mathbf{X}'D\mathbf{X} \\ &= D\mathbf{X} - D\mathbf{X} = 0 \end{aligned}$$

Proof of Lemma 1 (Page 15)

The proofs of parts a), b) and c) follow the same steps as in Schmid (2007) on pages 213ff so it is just referred to these explanations there.

Parts d) and e) can be proven together. The major part is analogous to the proof of Schmid (2007) second lemma (page 216f). However, for better understanding of the differing steps the complete proof is given. In order to obtain a probability limit of $\tilde{s}_{ij}$, $i, j = y, x_1, \ldots, x_p$ the inverse regressions of (3.26) and (3.27) are needed, implying inverse regressions for the aggregated data values

$$\tilde{x}_i = \alpha_i + \gamma_i \tilde{h} + \tilde{\delta}_i, \qquad i = 1, \ldots, p \tag{B.1}$$

$$\tilde{y} = \alpha_y + \gamma_y \tilde{h} + \tilde{\delta}_y \ , \tag{B.2}$$

with $\tilde{\delta}_i$ and $\tilde{\delta}_y$ containing the aggregated values of δ_i and δ_i, respectively. Those relationships in (B.1) and (B.2) imply

$$\tilde{s}_{ij} = \gamma_i \gamma_j \tilde{s}_{hh} + \tilde{s}_{\delta_i \delta_j} + \gamma_i \tilde{s}_{h\gamma_j} + \gamma_j \tilde{s}_{h\gamma_i} \ , \ i, j = y, x_1, \ldots x_p \tag{B.3}$$

'By part a) of the lemma, $\operatorname{plim}_{n\to\infty} \tilde{s}_{hh} = \sigma_{hh}$. *Moreover, from the proof of parts a), b) and c), taking* $W = \gamma_i$, [$\operatorname{plim}_{n\to\infty} \tilde{s}_{h\gamma_i}$ becomes] $\operatorname{plim}_{n\to\infty} \tilde{s}_{h\gamma_i} = \sigma_{h\gamma_i} = 0$, $i = y, x_1, \ldots x_p$.*'* [1] The probability limit of $\tilde{s}_{\delta_i\delta_j}$ is bounded by

$$\frac{1}{2k}\sigma_{\delta_i\delta_j} \leq \operatorname*{plim}_{n\to\infty} \tilde{s}_{\delta_i\delta_j} \leq \frac{1}{k}\sigma_{\delta_i\delta_j} \tag{B.4}$$

This becomes clear as the sorted variables $\delta_{[1]i}, \ldots, \delta_{[n]i}$ are i.i.d. with the same distribution as δ_i. Secondly, the empirical covariance between δ_i and δ_j can be written as

$$\operatorname{Cov}\left(\tilde{\delta}_i, \tilde{\delta}_j\right) = \frac{1}{n-1}\sum_{r=1}^{n}(\tilde{\delta}_{ir} - \bar{\tilde{\delta}}_i)(\tilde{\delta}_{jr} - \bar{\tilde{\delta}}_j). \tag{B.5}$$

Applying the computational formula for the covariance and with $\bar{\tilde{\delta}}_j = \bar{\delta}_j, \forall j = y, x_1, \ldots, x_p$, follows

$$\operatorname{Cov}\left(\tilde{\delta}_i, \tilde{\delta}_j\right) = \frac{1}{n-1}\left(\sum_{r=1}^{n}\left(\tilde{\delta}_{ir}\tilde{\delta}_{jr}\right) - n\bar{\delta}_i\bar{\delta}_j\right). \tag{B.6}$$

[1] Schmid (2007), page 216

As each $\tilde{\delta}_{ir}$ and $\tilde{\delta}_{jr}$ can be written as the sum of its original elements divided by their group–size,

$$\begin{aligned}\tilde{\delta}_{ir}\tilde{\delta}_{jr} &= \left(\frac{1}{k_r}\sum_{s=1}^{k_r}\delta_{is}\right)\left(\frac{1}{k_r}\sum_{s=1}^{k_r}\delta_{js}\right) && \text{(B.7)}\\ &= \bar{\delta}_{ig}\bar{\delta}_{jg}, && \text{(B.8)}\end{aligned}$$

with g as an identifier for the group. Because in each group g are k_g elements, the sums in (B.6) can be rewritten as the sum over all groups

$$\operatorname{Cov}\left(\tilde{\delta}_i, \tilde{\delta}_j\right) = \frac{1}{n-1}\left(\sum_{g=1}^{G}\left(k_g\bar{\delta}_{ig}\bar{\delta}_{jg}\right) - n\bar{\delta}_i\bar{\delta}_j\right). \tag{B.9}$$

With a zero-sum supplement it follows

$$= \frac{1}{n-1}\left(\sum_{g=1}^{G}\left(-\sum_{r=1}^{k_g}\delta_{igr}\delta_{jgr} + k_g\bar{\delta}_{ig}\bar{\delta}_{jg} + \sum_{r=1}^{k_g}\delta_{igr}\delta_{jgr}\right) - n\bar{\delta}_i\bar{\delta}_j\right) \tag{B.10}$$

$$= \frac{1}{n-1}\left(-\sum_{g=1}^{G}(k_g-1)\operatorname{Cov}\left(\delta_{ig}, \delta_{jg}\right) + \sum_{g=1}^{G}\sum_{r=1}^{k_g}\left(\delta_{igr}\delta_{jgr}\right) - n\bar{\delta}_i\bar{\delta}_j\right) \tag{B.11}$$

$$= \frac{1}{n-1}\left(-\sum_{g=1}^{G}(k_g-1)\operatorname{Cov}\left(\delta_{ig}, \delta_{jg}\right) + \sum_{s=1}^{n}\left(\delta_{is}\delta_{js}\right) - n\bar{\delta}_i\bar{\delta}_j\right) \tag{B.12}$$

$$= \frac{1}{n-1}\left(-\sum_{g=1}^{G}(k_g-1)\operatorname{Cov}\left(\delta_{ig}, \delta_{jg}\right) + (n-1)\operatorname{Cov}\left(\delta_i, \delta_j\right)\right) \tag{B.13}$$

$$= \operatorname{Cov}\left(\delta_i, \delta_j\right) - \frac{1}{n-1}\left(\sum_{g=1}^{G}(k_g-1)\operatorname{Cov}\left(\delta_{ig}, \delta_{jg}\right)\right). \tag{B.14}$$

As δ_{ig} is i.i.d like δ_i, (B.14) simplifies to

$$\begin{aligned}
\operatorname{Cov}\left(\tilde{\delta}_i,\tilde{\delta}_j\right) &= \operatorname{Cov}\left(\delta_i,\delta_j\right) - \frac{1}{n-1}\left(\sum_{g=1}^{G}(k_g-1)\operatorname{Cov}\left(\delta_i,\delta_j\right)\right) && \text{(B.15)}\\
&= \operatorname{Cov}\left(\delta_i,\delta_j\right) - \frac{1}{n-1}\left(\operatorname{Cov}\left(\delta_i,\delta_j\right)\sum_{g=1}^{G}(k_g-1)\right) && \text{(B.16)}\\
&= \operatorname{Cov}\left(\delta_i,\delta_j\right) - \frac{1}{n-1}\left(\operatorname{Cov}\left(\delta_i,\delta_j\right)\left(\underbrace{\sum_{g=1}^{G}k_g}_{=n} - \underbrace{\sum_{g=1}^{G}1}_{G}\right)\right) && \text{(B.17)}\\
&= \operatorname{Cov}\left(\delta_i,\delta_j\right) - \frac{n}{n-1}\operatorname{Cov}\left(\delta_i,\delta_j\right) + \frac{G}{n-1}\operatorname{Cov}\left(\delta_i,\delta_j\right) && \text{(B.18)}\\
&= \frac{G-1}{n-1}\operatorname{Cov}\left(\delta_i,\delta_j\right). && \text{(B.19)}
\end{aligned}$$

With $\tilde{s}_{\delta_i\delta_j} = \operatorname{Cov}\left(\tilde{\delta}_i,\tilde{\delta}_j\right)$ and its probability limit and $\operatorname*{plim}_{n\to\infty}\operatorname{Cov}\left(\delta_i,\delta_j\right) = \sigma_{\delta_i\delta_j}$,

$$\tilde{\sigma}_{\delta_i\delta_j} = \operatorname*{plim}_{n\to\infty}\tilde{s}_{\delta_i\delta_j} = \operatorname*{plim}_{n\to\infty}\frac{G-1}{n-1}\operatorname{Cov}\left(\delta_i,\delta_j\right) \qquad \text{(B.20)}$$

is obtained. As G increases with n in the same manner it is not possible to obtain the exact limit. However, from the definition of the k–Ward–Algorithm with its minimal and maximal group–size, a bounding is achievable. The minimal group–size is k, so the aggregated data implies $\frac{n}{k}$ groups if each group consists of k elements, and $\frac{n}{2k-1}$ groups in the maximum case of every group–size $2k-1$. The lower bound of (B.20) is then

$$\frac{1}{2k-1}\sigma_{\delta_i\delta_j} = \operatorname*{plim}_{n\to\infty}\frac{n-2k-1}{(2k-1)(n-1)}\operatorname{Cov}\left(\delta_i,\delta_j\right) \le \operatorname*{plim}_{n\to\infty}\frac{G-1}{n-1}\operatorname{Cov}\left(\delta_i,\delta_j\right) = \tilde{\sigma}_{\delta_i\delta_j} \qquad \text{(B.21)}$$

and the upper bound of (B.20) is similarly

$$\frac{1}{k}\sigma_{\delta_i\delta_j} = \operatorname*{plim}_{n\to\infty}\frac{n-k}{k(n-1)}\operatorname{Cov}\left(\delta_i,\delta_j\right) \ge \operatorname*{plim}_{n\to\infty}\frac{G-1}{n-1}\operatorname{Cov}\left(\delta_i,\delta_j\right) = \tilde{\sigma}_{\delta_i\delta_j}. \qquad \text{(B.22)}$$

Now with (B.3) and (B.4) an upper bound of σ_ij can be constructed

$$\tilde{\sigma}_{ij} = \operatorname*{plim}_{n\to\infty}\tilde{s}_{ij} \le \gamma_i\gamma_j\sigma_{hh} + \frac{1}{k}\sigma_{\delta_i\delta_j}. \qquad \text{(B.23)}$$

'*As* $\sigma_{ij} = \gamma_i\gamma_j\sigma_{hh} + \sigma_{\delta_i\delta_j}$ *and* $\gamma_i = \sigma_{ih}/\sigma_{hh}, i = y, x_1, \ldots, x_p$, [it is] *finally obtain[ed]*'[2]

$$\begin{aligned}
\tilde{\sigma}_{ij} &\le \gamma_i\gamma_j\sigma_{hh} + \frac{1}{k}\sigma_{\delta_i\delta_j}\\
&\le \frac{\sigma_{ih}\sigma_{jh}}{\sigma_{hh}} + \frac{1}{k}\left(si1_{ij} - \gamma_i\gamma_j\sigma_{hh}\right) \qquad \text{(B.24)}\\
&\le \left(1-\frac{1}{k}\right)\frac{\sigma_{ih}\sigma_{jh}}{\sigma_{hh}} + \frac{1}{k}\sigma_{ij}.
\end{aligned}$$

[2]Schmid (2007), page 216

The lower bound is constructed similarly. With both follow parts d) and e) of the lemma.

Proof of Lemma 2 (Page 16)

This proof follows mostly the previous one. The only difference is in the evaluation of $\operatorname*{plim}_{n\to\infty} \tilde{s}_{\delta_i\delta_j}$.
Let K be independent of $\delta_i,\ i = y, x_1, \dots, x_p$ as the groups rely on the data but not on their actual values. As K is a random variable, the number of groups G is random as well. Furthermore assume that K follows a discrete distribution function with an expected value $\mathbb{E}\left(K\right) = \mu_k$ and its support from k to $2k-1$. This seems plausible as before the grouping process all possible group–sizes are probable. Besides the fact that the last few group–sizes are not absolutely random, as the minimal group–size criterion and the finite number of observations has to be taken into account. Then the expected value of $\mathrm{Cov}\left(\tilde{\delta}_i, \tilde{\delta}_j\right)$ relative to K has to be evaluated.

$$\begin{aligned}
\mathbb{E}_K\left(\tilde{s}_{\delta_i\delta_j}\right) &= \mathbb{E}_K\left(\mathrm{Cov}\left(\tilde{\delta}_i, \tilde{\delta}_j\right)\right)\\
&= \mathbb{E}_K\left(\frac{G-1}{n-1}\mathrm{Cov}\left(\delta_i, \delta_j\right)\right)\\
&= \mathrm{Cov}\left(\delta_i, \delta_j\right)\mathbb{E}_K\left(\frac{G-1}{n-1}\right)\\
&= \mathrm{Cov}\left(\delta_i, \delta_j\right)\frac{\mathbb{E}_K\left(G\right)-1}{n-1}
\end{aligned} \tag{B.25}$$

The above statement relies on the expected value of G. In case of a fixed–size microaggregation the number of groups can easily be denoted by $\frac{n}{k}$. This idea is ported to variable–size microaggregation by considering the expected value of K and then the expected number of groups (G) can be expressed by a function of the expected value of K:

$$\mathbb{E}\left(G\right) = \frac{n}{\mathbb{E}\left(K\right)} \ .$$

With the above and (B.25) follows

$$\begin{aligned}
\mathbb{E}_K\left(\mathrm{Cov}\left(\tilde{\delta}_i, \tilde{\delta}_j\right)\right) &= \mathrm{Cov}\left(\delta_i, \delta_j\right)\frac{\frac{n}{\mathbb{E}(K)}-1}{n-1}\\
&= \mathrm{Cov}\left(\delta_i, \delta_j\right)\frac{n-\mathbb{E}\left(K\right)}{(n-1)\mathbb{E}\left(K\right)} \ .
\end{aligned} \tag{B.26}$$

Now concerning the probability limit, it results in:

$$
\begin{aligned}
\operatorname*{plim}_{n\to\infty} \tilde{s}_{\delta_i\delta_j} &= \operatorname*{plim}_{n\to\infty} \left(\operatorname{Cov}(\delta_i, \delta_j) \frac{n - \mathbb{E}(K)}{(n-1)\mathbb{E}(K)} \right) \\
&= \operatorname*{plim}_{n\to\infty} \operatorname{Cov}(\delta_i, \delta_j) \cdot \operatorname*{plim}_{n\to\infty} \left(\frac{n - \mathbb{E}(K)}{n-1} \cdot \frac{1}{\mathbb{E}(K)} \right) \\
&= \sigma_{\delta_i\delta_j} \cdot \frac{1}{\mathbb{E}(K)} \underbrace{\operatorname*{plim}_{n\to\infty} \frac{n - \mathbb{E}(K)}{n-1}}_{=1} \qquad \text{(B.27)} \\
&= \frac{1}{\mathbb{E}(K)} \sigma_{\delta_i\delta_j} \\
&= \frac{1}{\mu_k} \sigma_{\delta_i\delta_j}.
\end{aligned}
$$

The rest of the proof is done in analogy to the creation of the lower bound (See (B.23) and (B.24)) by substituting the term k with μ_k. The final result is then

$$
\tilde{\sigma}_{ij} = \left(1 - \frac{1}{\mu_k}\right) \frac{\sigma_{ih}\sigma_{jh}}{\sigma_{hh}} + \frac{1}{\mu_k}\sigma_{ij} \ ,
$$

from which the proof of parts e) and f) of Lemma 2 follows.

Appendix C

R– and FORTRAN77–Code

Along with this thesis a CD is shipped containing the R– and FORTRAN77–scripts. They also contain the generated graphics and saved estimators of the simulations.

R–Code

File *kward.r*

In this section the R–code for the main function *kwm* is given. The function aims at microaggregating a data.frame object by using Mateo-Sanz and Domingo-Ferrer (1998) k–Ward–Algorithm. The output is a R object of class data.frame, with the same variables and row.names; a vector giving the group belongings for each observation and a second vector giving the number of observation in each group for each observation. This method should be applied to a data set containing only continual numeric variables. This function is constructed as a container for the actual grouping and sorting function *kwm_algorithm*.

```
#######################################################
# kwm: K-Ward-Microaggregation                        #
#                                                     #
# @params k:          minimum group size.             #
# @params data:       Data set to be microaggregated, #
#                     has to be of type 'data.frame'. #
#                                                     #
# @return data:       data.frame object containing    #
#                     the microaggregated data set.   #
# @return order:      vector containing the group-    #
#                     belongings.                     #
# @return groupsize:  vector containing the groupsize #
#                     for each observation.           #
#######################################################

kwm <- function (data , k = 3) {

  # validating the input variables
  mf <- match.call()
  if (missing ( data )) stop ( "There must be a data set to be microaggregated" )
  if (!is.data.frame ( data )) stop ( cat("'", substitute( data ),
    "'must be of type 'data.frame'", sep="") )
  n <- nrow ( data )
  if ( n/k < 2 ) stop ( "data contains too less observations" )
  if ( as.integer(k)-k!=0 || k == 1 ) stop ( "k has to be an integer greater than 1" )

  # initializing variables
  # gk: vector containing the group-size for each row in the data.frame object
  # istogroup: logical vector, if the row is heading of its group (TRUE)
  #            or bounded to another (FALSE)
  # order: vector containing the group belonging of each row
  gk <- rep.int ( 1 , n)
  istogroup <- rep ( TRUE , n )
  order <- c(1:n)

  # calling the function which does the real group forming
  kwm_body <- kwm_algorithm ( data = data, k = k, n = n, gk = gk,
    istogroup = istogroup, order = order)

  # finally microaggregate the complete data set according to order and groupcount
  datam <- microaggregate(data, kwm_body$orderv, kwm_body$groupcount)

  # ensure thats it is a data.frame and update row.names as the ones at
  # this point are no longer usable
  datam <- as.data.frame(datam)
  row.names(datam) <- c(1:NROW(datam))

  # return microaggregated data.frame
  return(list(data=datam, order=kwm_body$orderv, groupsize=kwm_body$groupcount))
}
```

The function *kwm_recursive* is used as a kind of interface to *kwm_algorithm*. This was necessary as *kwm* does aggregation at the end, which is not needed for a recursion and furthermore the recursion requires the '*group belongings*' as input.

```
#######################################################
# kwm_recursive: K-Ward-Microaggregation              #
#                                                     #
# @params k:          minimum group size.             #
# @params data:       Data set to be microaggregated, #
#                     has to be of type 'data.frame'. #
# @params ordered:    vector containing group         #
#                     belongings.                     #
#                                                     #
# @return order:      vector containing the group-    #
#                     belongings.                     #
# @return groupsize:  vector containing the groupsize #
#                     for each observation.          #
######################################################

kwm_recursive <- function (data , k , ordered) {

  # initializing variables
  # n: observation/rows of data
  # gk: vector containing the group-size for each row in the data.frame object
  # istogroup: logical vector, if the row is heading of its group (TRUE)
  # or bounded to another (FALSE)
  n <- nrow ( data )
  gk <- rep.int ( 1 , n)
  istogroup <- rep ( TRUE , n )

  # calling the function which does the real group forming
  kwm_body <- kwm_algorithm ( data = data, k = k, n = n, gk = gk,
    istogroup = istogroup, order = ordered)

  # returning vectors for group belongings and element-count per group
  return( list( kwm_body$orderv, kwm_body$groupcount ))
}
```

Kwm_algorithm is the complete calculation process of k–Ward–Algorithm. It requires the shared object *kward.so* on Unix and *kward.dll* on Windows machines[1]. It uses low level function as *step1* and *dist_ward*.

```
######################################################################
# kwm_algorithm: This is the body of the k-Ward-Algorithm            #
#                                                                    #
# @params data:       data.frame object with n rows/observations.    #
# @params k:          integer giving minimum group size.             #
# @params n:          integer giving number of rows in 'data'.       #
# @params gk:         vector containing the elements in each.        #
#                     group per row (1xn).                           #
# @params istogroup:  logical vector (1xn), if the row is heading of #
#                     its group (TRUE) or bounded to another (FALSE). #
# @params order:      vector containing the group belongings (1xn).  #
#                                                                    #
# @return orderv:     Updated group belonging.                       #
# @return groupcount: Updated elements in group.                     #
######################################################################

kwm_algorithm <- function (data, k, n, gk, istogroup, order) {

  #####################
  # k-Ward First Step #
  #####################
  # creating a distance matrix between all rows with the diagonal
  # containing -1 as elements
```

[1]R Development Core Team (2009*d*,*a*); Ligges (2008)

```
dist1 <- as.matrix ( dist ( data ))
dist1 <- dist1 + diag(rep (-1, n))

# finding the maximum distance between two rows, saving the coordinates
# of these in maxdist (only the first found maximum distance is used)
maxdist <- which(dist1==max(dist1),arr.ind=T)

# grouping each k-1 nearest rows around those rows identified in the last
# step by using function 'step1'
vs1 <- step1(dist1, order, gk, istogroup, n, k, maxdist[1,1])
vs2 <- step1(vs1[[1]], vs1[[2]], vs1[[3]], vs1[[4]], n, k, maxdist[1,2])
dist1 <- vs2[[1]]
order <- vs2[[2]]
gk <- vs2[[3]]
istogroup <- vs2[[4]]

# de-allocate memory in case the distance matrix is quite large
# thus twice a call of garbage collector
rm(dist1)
gc()
gc()

# in case there a some single groups left, performing step 2 of k-Ward-Algorithm
if (min(gk) < k) {

  ######################
  # k-Ward Second Step #
  ######################
  # virtually aggregating the data to get the mean of the groups formed
  # before (important for ward's distance calculation)
  data1 <- microaggregate(data, order , gk)

  # initializing the matrix which is going to be filled with the distances
  # between the groups (only lower the diagonal)
  dist <- matrix(0, nrow=n, ncol=n)

  # actual calculating the distance matrix (only the pairs lower than the
  # diagonal are evaluated) by call to function 'dist_ward'
  for (i in 2:n) {
    if (!istogroup[i]) next
    for (j in 1:(i-1)) {
      if (!istogroup[j]) next
      dist[i,j] <- dist_ward(data1[i,], data1[j,], gk[i], gk[j])
    }
  }

  # setting the distance between the groups identified in 'k-Ward First step'
  # to a high value thus it's highly improbable that they are merged afterwards
  if (maxdist[1,1]>maxdist[1,2]) {
    dist[maxdist[1,1], maxdist[1,2]] <- 100000000000
  } else {
    dist[maxdist[1,2], maxdist[1,1]] <- 100000000000
  }

  # creating distance object to save memory and get a list
  ddist <- as.dist(dist)

  # de-allocating memory by removing virtually aggregated data set and
  # the distance matrix, afterwards call of garbage collector twice
  rm(data1, dist)
  gc()
  gc()

  # preparation to Fortran-function call
  m <- as.integer(attr(ddist, "Size"))
  len <- as.integer(m * (m - 1)/2)
  if (length(ddist) != len)
  (if (length(ddist) < len)
    stop
  else warning)("dissimilarities of improper length")

  # call of the external FORTRAN77 function kWard
  # (for details see its documentation in kward.f)
  fkwd <- .Fortran("kward", n = m, len = len, members = as.integer(gk),
        nn = integer(m), disnn = double(m), flag = as.logical(istogroup),
        diss = as.double(ddist), order = as.integer(order), minsize = as.integer(k))

  # assigning the returned variables to its original ones
  istogroup <- fkwd$flag
  order <- fkwd$order
  gk <- fkwd$members

  # again de-allocation of memory by removing the not longer
  # used distance matrix dist and calling gc() twice
  rm(ddist)
  gc()
  gc()
}

# determining if there has to be a step 3 of k-Ward-Algorithm
# by checking the group-counts of the group heads
for (i in 1:n) {

  # if no head continue loop
  if (!istogroup[i]) next

  # if groupcount is less than 2*k then k-Ward step 3
  # is not needed, so continue loop
  if (gk[i] < (2*k)) {
    next
```

```
} else {

  #####################
  # k-Ward Third Step #
  #####################
  # get the highest value of order so the group specific
  # order vector does not mess up with the original vector
  m1 <- max(order)

  # initializing variables
  # orders: same function as 'order'
  # original: vector to store the positions of the extracted
  #           row in the original data set
  # datas: data.frame/matrix containing the extracted data rows
  # r: index of the row
  orders <- numeric(gk[i])
  original <- numeric(gk[i])
  datas <- matrix(0, ncol=ncol(data), nrow=gk[i])
  r <- 1

  # extracting the elements of the group and writing
  # its values to the vectors and matrix
  for (j in 1:n) {
    if (order[j]==order[i]) {
      datas[r,] <- as.matrix(data[r,])
      orders[r] <- m1 + r
      original[r] <- j
      r <- r + 1
    } else {
      next
    }
  }

  # transform data matrix to data.frame
  datas <- as.data.frame(datas)

  # Perform recursion step by calling function 'kwm_recursive'.
  # The decision to use an external function was made as there
  # has to be another parameter given (order),
  # and the validation of input parameter is obsolete, as it
  # is checked before calling the function.
  rkwm <- kwm_recursive (data=datas, k=k, ordered=orders)

  # putting the resulted variables back to its original
  # position in the order and groupcount vector
  for (h in 1:gk[i]) {
    indo <- original[h]
    order[indo] <- rkwm[[1]][h]
    gk[indo] <- rkwm[[2]][h]
  }
```

```
      # running garbage collector
      gc()
      gc()
    }
  }

  # returning vectors for group belongings and element-count per group
  return( list( orderv = order, groupcount = gk ))
}
```

The only purpose of *step1* is to perform the first step in the k–Ward–Algorithm.

```
#######################################################################
# step1: Helpful function to first step of k-Ward-Algorithm           #
#                                                                     #
# @param distamt:     distance matrix (NxN).                          #
# @param longorder:   vector containing the group belongings (1xN).   #
# @param groupsizes:  vector containing the elements in each.         #
#                     group per row (1xN).                            #
# @param groups:      logical vector (1xN), if the row is heading of  #
#                     its group (TRUE) or bounded to another (FALSE). #
# @param obs:         integer giving number of N.                     #
# @param minsize:     integer giving the minimum group-size so        #
#                     (minsize-1) iterations needed.                  #
# @param index:       integer of the row-, column-index in the matrix #
#                     to be considered for Nearest-Neighbor-search.   #
#                                                                     #
# @return distmat:    updated distance matrix 'distmat'.              #
# @return longorder:  updated vector 'longorder'.                     #
# @return groupsizes: updated vector 'groupsizes'.                    #
# @return groups:     updated vector 'groups'.                        #
#######################################################################

step1 <- function (distmat, longorder, groupsizes, groups, obs, minsize, index) {

  # looking for k-1 nearest distances to index
  for (kk in 1:(minsize-1)) {

    # getting index with smallest distance > -1
    ind <- 0
    ind <- which(distmat[index,]>-1)
    ind <- ind[which(distmat[index,ind]==min(distmat[index,ind]))]

    # in case there are more than one with smallest distance, use first
    if(length(ind)>1) ind <- ind[1]

    # sets group and groupcount of found index same with given index
    longorder[ind] <- longorder[index]
    groupsizes[ind] <- groupsizes[index]
    # updating groupcount of all groups with given index as order
    for (o in 1:obs) {
      if (longorder[o] == longorder[index]) {
        groupsizes[o] <- groupsizes[o] + 1
      }
    }

    # removing row and column of found index from
    # distance search path by setting to -1
    distmat[ind,] <- -1
    distmat[,ind] <- -1

    # setting unbound value of found index to false
    groups[ind] <- FALSE
  }

  # returning vectors for group belongings and element-count per group
  return(list(distmat, longorder, groupsizes, groups))
}
```

Dist_ward calculates the distance between two groups, by given elements per group.

```
############################################################
# Small function to calculate Ward's distance             #
#                                                          #
# @param x:              vector containing either one      #
#                        observation or aggregated group #
#                        values (group1).                  #
# @param y:              vector containing either one      #
#                        observation or aggregated group #
#                        values (group2).                  #
# @param nx:             number of elements in group1.     #
# @param ny:             number of elements in group2.     #
#                                                          #
# @return distance_ward: distance between group1 and       #
#                        group2 according to increase      #
#                        in SSE.                           #
############################################################

dist_ward <- function (x , y , nx , ny) {

  # distance measure according to Ward's method (only needed for initial step)
  distance_ward <- ((nx*ny)/(nx+ny))*((dist(rbind(x,y))[1])^2)
  return(distance_ward)
}
```

Whereas the previous function make only sense in this context, the function *microaggregate* may be generally used for microaggregating a data.frame object by a given group belonging and group–size vector.

```
####################################################
# microaggregate: Microaggregate a data set        #
#                                                  #
# @param dataset:    A data.frame object, with     #
#                    each row an observation.      #
# @param ordering:   A vector with same length     #
#                    as rows in dataset; contains  #
#                    the group belongings.         #
# @param groupcount: A vector with same length     #
#                    as rows in dataset; contains  #
#                    the elements in the group.    #
#                                                  #
# @return x:         A data.frame object with same #
#                    order as dataset but with     #
#                    microaggregated values.       #
####################################################

microaggregate <- function (dataset, ordering, groupcount) {

  # looking if input is appropriate
  if (missing ( dataset )) stop ( "There must be a data set to be microaggregated" )
  if (!is.data.frame(dataset)) dataset <- as.data.frame(dataset)
  n <- nrow ( dataset )
  if (missing ( ordering ) || (length(ordering) != n)) stop ( cat(
    "There must be a definition of group belonging for each observation in '",
     substitute(dataset), "'\n", sep="" ))
  if (missing ( groupcount ) || (length(groupcount) != n)) stop ( cat("There must
     be a definition of elements in the group for each observation in '",
     substitute(dataset), "'\n", sep="" ))

  # initializing vector of original order of data set
  trues <- c(1:n)

  # building data.frame with original data.frame, order, groupcount and initial order
  dataset <- cbind(dataset, cbind(ordering, cbind(groupcount, trues)))

  # ordering data.frame according to order to get groups together
  dataset <- dataset[ order(dataset$ordering) ,]

  # saving new ordering in extra variable and then deleting it from the data.frame
  trues <- dataset$trues
  dataset <- dataset[, -ncol(dataset)]

  # aggregating over groups with function FUN
  dataset <- lapply(split(dataset, dataset$ordering), mean)

  # initializing a matrix with one row and n+2 columns
  x <- matrix(0, ncol=length(unlist(dataset[[1]])), nrow=1)

  # forming matrix where each group is replicated according
  # to its value in groupcount
  for (i in 1:(NROW(dataset))) {
    xu <- unlist(dataset[[i]])
    if (i==1) {
      x[1,] <- xu
      gk <- (x[1,ncol(x)])-1
    } else {
      gk <- (xu[length(xu)])
    }
    if (gk==0) next
    for (r in 1:gk) {
      x<-rbind(x, xu)
    }
  }

  # transforming matrix with variable new ordering
  # to data.frame, updating row.names
  x <- as.data.frame(cbind(x,trues), row.names=c(1:length(trues)))

  # sorting the data.frame according to the new ordering variable.
  # The data.frame has now the same order as the original one.
  x <- x[ do.call("order", list(x$trues)) ,]

  # removing the bounded help-variables from the data.frame
  x <- x[, -c((ncol(x)-2):ncol(x))]

  # returning the microaggregated data.frame
  return(x)
}
```

File *kwardmicroaggregation.r*

This file sources the code of *kward.r* and automatically loads the shared object created by *kward.f* if existing.

File *simulation.r*

The following code was used to created the data for the simulations. As the generation of the 900 microaggregated data sets in the first simulation and 390 in the second require a lot of computation time, the results of these are given in the files *simulation1.Rdata* and *simulation2.Rdata*. Re–running this code will lead to different values of the estimators as the data sets were generated randomly, but the conclusions remain the same.

File *graphics.r*

This routine creates the graphics of estimators, based on the simulations.

File *validation.r*

The purpose of this script is to validate the programming and compiling of the code as it compares two user–generate data.frames with predefined ones.

FORTRAN77–Code

The following function is based on the 'hclust' function used in the R-package stats[2] in source–file *hclust.f.* Its authors and the year of contribution are

- F. Murtagh, ESA/ESO/STECF, Garching, February 1986
- Ross Ihaka, Dec 1996 (Modifications for R)
- Fritz Leisch, Jun 2000 (Modifications for R)
- Martin Maechler, Apr 2001 (all vars declared) .

The idea of Nearest–Neighbor–Search in order to identify a minimum in a distance matrix is left unchanged, hence this kind of search algorithm is computing time optimal. Its computing time is of $O(n^2)$ whereas a naive search in each step over all elements in the matrix would lead to $O(n^3)$. Domingo-Ferrer and Mateo-Sanz (2002) stated the same result for the here used stored matrix approach[3]. The used algorithm is based on *Algorithm D* proposed by Murtagh[4]

The difference between the algorithm and the one in *hclust* is the updating of the distance, as the distance between two groups, each containing k or more elements has to be set to the highest value, to ensure that they never get merged. Another major change is the implementation of a exiting rule: The algorithm is exited if the minimum of the group–size–vector is k.
The following pages give the outline of the code.

[2]R Development Core Team (2008)
[3]Domingo-Ferrer and Mateo-Sanz (2002), page 193f
[4]Murtagh (1985), pages 71ff, 86

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++C
C                                                            C
C  WARDS HIERARCHICAL CLUSTERING method adapted for kWARD    C
C                                                            C
C  Parameters:                                               C
C                                                            C
C  N                  the number of elements being clustered. C
C  DISS(LEN)          dissimilarities in lower half diagonal  C
C                     storage; LEN = N.N-1/2.                 C
C  MEMBR, NN, DISNN   vectors of length N, used to store      C
C                     cluster cardinalities, current nearest  C
C                     neighbour, and the dissimilarity assoc. C
C                     with the latter.                        C
C                     (MEMBR must be initialized by R to the  C
C                     default of  rep(1, N)  )                C
C  FLAG               boolean indicator of agglomerable obj./ C
C                     clusters.                               C
C  ORDER              vector of length N, used to store the   C
C                     cluster numbers.                        C
C  MINSIZE            minimal cardinality for the result      C
C                     clusters.                               C
C                                                            C
C------------------------------------------------------------C
      SUBROUTINE KWARD(N,LEN,MEMBR,NN,DISNN,
     X                 FLAG,DISS,ORDER,MINSIZE)
c Args
      INTEGER N, LEN, MINSIZE
      INTEGER NN(N), MEMBR(N), ORDER(N)
      LOGICAL FLAG(N)
      DOUBLE PRECISION DISS(LEN), DISNN(N)
c Var
      INTEGER IM, JJ, JM, I, J, IND, I2, J2, K, IND1,
     X            IND2, IND3, KI, KJ, KK, IO, JO
      DOUBLE PRECISION INF, DMIN, D12, GMIN
c External function
      INTEGER IOFFST
c
      DATA INF/1.D+300/
c
c     unnecessary initialization of im jj jm to keep g77 -Wall happy
c
      IM = 0
      JJ = 0
      JM = 0
C
C  Carry out an agglomeration - first create list of NNs
C  Note NN and DISNN are the nearest neighbour and its distance
C  TO THE RIGHT of I.
C
      DO 20 I=1,N-1
```

```
         IF (.NOT.FLAG(I)) GOTO 20
         DMIN=INF
         DO 10 J=I+1,N
            IF (.NOT.FLAG(J)) GOTO 10
            IND=IOFFST(N,I,J)
            IF (DISS(IND).GE.DMIN) GOTO 10
               DMIN=DISS(IND)
               JM=J
  10        CONTINUE
         NN(I)=JM
         DISNN(I)=DMIN
  20     CONTINUE
C
  100 CONTINUE
C     Next, determine least diss. using list of NNs
      DMIN=INF
      DO 30 I=1,N-1
         IF (.NOT.FLAG(I)) GOTO 30
         IF (DISNN(I).GE.DMIN) GOTO 30
         DMIN=DISNN(I)
         IM=I
         JM=NN(I)
  30     CONTINUE
C
C  This allows an agglomeration to be carried out.
C
      I2=MIN0(IM,JM)
      J2=MAX0(IM,JM)
C
C  Update dissimilarities from new cluster.
C
      DMIN=INF
      KI=MEMBR(I2)
      KJ=MEMBR(J2)
      IO=ORDER(I2)
      JO=ORDER(J2)
      DO 40 K=1,N
         KK=MEMBR(K)
         IF (ORDER(K).EQ.IO) THEN
            MEMBR(K)=KK+KJ
         ELSEIF (ORDER(K).EQ.JO) THEN
            MEMBR(K)=KK+KI
            ORDER(K)=IO
            FLAG(K)=.FALSE.
         ELSE
            MEMBR(K)=KK
         ENDIF
         IF (.NOT.FLAG(K)) GOTO 40
         IF (K.EQ.I2) GOTO 40
         IF (I2.LT.K) THEN
```

```
            IND1=IOFFST(N,I2,K)
         ELSE
            IND1=IOFFST(N,K,I2)
         ENDIF
         IF (J2.LT.K) THEN
            IND2=IOFFST(N,J2,K)
          ELSE
            IND2=IOFFST(N,K,J2)
         ENDIF
C        Setting distance to INF in case both group-sizes are greater or equal to minsize
         IF ((KI.GE.MINSIZE).AND.(KK.GE.MINSIZE)) THEN
            DISS(IND1)=INF
         ELSE
            IND3=IOFFST(N,I2,J2)
            D12=DISS(IND3)
            DISS(IND1)=(KI+KK)*DISS(IND1)+
     X        (KJ+KK)*DISS(IND2) - KK*D12
            DISS(IND1)=DISS(IND1) / (KI+KJ+KK)
         ENDIF
  40     CONTINUE
C
C  Update list of NNs
C
      DO 50 I=1,N-1
         IF (.NOT.FLAG(I)) GOTO 50
C        (Redetermine NN of I:)
         DMIN=INF
         DO 60 J=I+1,N
            IF (.NOT.FLAG(J)) GOTO 60
            IND=IOFFST(N,I,J)
            IF (DISS(IND).GE.DMIN) GOTO 60
            DMIN=DISS(IND)
            JJ=J
  60        CONTINUE
         NN(I)=JJ
         DISNN(I)=DMIN
  50     CONTINUE
C
C  Find the minimal group-size in order array to determine whether
C  a new step has to be carried out
C
      GMIN=MEMBR(1)
      DO 70 I=1,N
         IF (MEMBR(I).LT.GMIN) THEN
            GMIN=MEMBR(I)
         ENDIF
  70     CONTINUE
C
C  Repeat previous steps until minimal group-size is minsize.
C
```

```
      IF (GMIN.LT.MINSIZE) GOTO 100
C
      RETURN
      END
C
      INTEGER FUNCTION IOFFST(N,I,J)
C  Map row I and column J of upper half diagonal symmetric matrix
C  onto vector.
      INTEGER N,I,J
      IOFFST=J+(I-1)*N-(I*(I+1))/2
      RETURN
      END
```

Bibliography

Domingo-Ferrer, J. and J. M. Mateo-Sanz (2002), 'Practical data-oriented microaggregation for statistical disclosure control', *IEEE Transactions on Knowledge and Data Engineering* **14**(1), 189–201.

Fahrmeir, L., A. Hamerle and G. Tutz (1996), *Multivariate statistische Verfahren*, 2nd edn, Berlin: de Gruyter.

Fayyoumi, E. and B. J. Oommen (2006), On optimizing the k-Ward microaggregation technique for secure statistical databases, *in* 'Information Security and Privacy, Proceedings', Vol. 4058 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin, pp. 324–335.

Feige, E. L. and H. W. Watts (1972), 'An investigation of consequences of partial aggregation of micro-economic data', *Econometrica* **40**(2), 343–360.

Lance, G. N. and W. T. Williams (1967), 'A general theory of classificatory sorting strategies: 1. hierarchical systems', *Computer Journal* **9**(4), 373–380.

Law on Statistics for Federal Purposes of Germany (2007).
URL: *http://www.bundesrecht.juris.de/bstatg_1987/index.html*

Lechner, S. and W. Pohlmeier (2003), 'Schätzung ökonometrischer Modelle auf der Grundlage anonymisierter Daten'.
URL: *http://www.ub.uni-konstanz.de/kops/volltexte/2003/1007*

Li, Y., S. Zhu, L. Wang and S. Jajodia (2002), A privacy-enhanced microaggregation method, *in* 'Foundations of Information and Knowledge Systems', Vol. 2284 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin, pp. 148–159.

Ligges, U. (2008), *Programmieren mit R*, Statistik und ihre Anwendungen, 3rd rev edn, Berlin: Springer-Verlag.

Mateo-Sanz, J. M. and J. Domingo-Ferrer (1998), 'A comparative study of microaggregation methods', *Qüestiio* **22**(3), 511–526.

Murtagh, F. (1985), *Multidimensional clustering algorithms.*, Compstat Lectures 4, Würzburg: Physica-Verlag.

R Development Core Team (2008), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org*

R Development Core Team (2009*a*), *R Installation and Administration*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org*

R Development Core Team (2009*b*), *R Internals*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org*

R Development Core Team (2009*c*), *R Language Definition*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org*

R Development Core Team (2009*d*), *Writing R Extensions*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org*

Schmid, M. (2007), *Estimation of a linear regression with microaggregated data.*, 1st edn, München: Verlag Dr. Hut.

Schmid, M. and H. Schneeweiss (2005), 'The effect of microaggregation procedures on the estimation of linear models: A simulation study', *Jahrbücher für Nationalökonomie und Statistik* **225**(5), 529–543.

Schmid, M., H. Schneeweiss and H. Küchenhoff (2007), 'Estimation of a linear regression under microaggregation with the response variable as a sorting variable', *Statistica Neerlandica* **61**(4), 407–431.

Ward, J. H. (1963), 'Hierarchical grouping to optimize an objective function', *Journal of the American Statistical Association* **58**(301), 236–&.

Affidavit

I, Paul Fink, hereby declare that this bachelor–thesis in question was written single–handed and no further as the denounced resources and sources were employed.

Munich, 2nd October 2009

(Paul Fink)

Eidesstattliche Erklärung

Hiermit versichere ich, Paul Fink, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 02.10.2009

(Paul Fink)